Maximum Likelihood Estimation

HW0 due today (ungraded but for pts)

Observe $X_1, X_2, \ldots, X_n$ drawn IID from $f(x;\theta)$ for some "true" $\theta = \theta_*$

Likelihood function $L_n(\theta) = \prod_{i=1}^{n} f(X_i;\theta)$

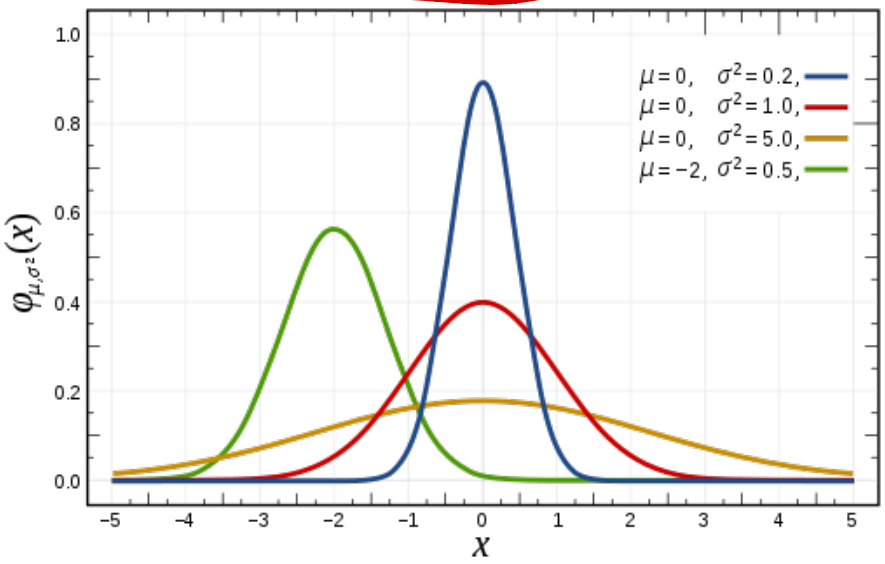
Log-Likelihood function $l_n(\theta) = \log(L_n(\theta)) = \sum_{i=1}^{n} \log(f(X_i;\theta))$

Maximum Likelihood Estimator (MLE) $\widehat{\theta}_{MLE} = \arg\max_{\theta} L_n(\theta)$

What about continuous variables?

- *Client*: What if I am measuring a **continuous variable**?
- ***You*: Let me tell you about Gaussians…**

$$P(x \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

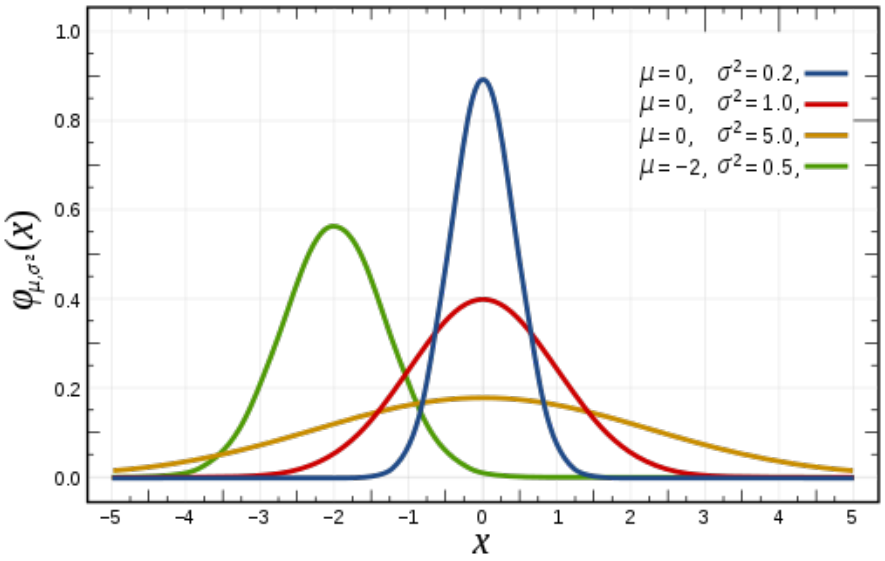


What about continuous variables?

- *Client*: What if I am measuring a **continuous variable**?
- ***You*: Let me tell you about Gaussians…**

$$P(x \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

PDF of Gaussian

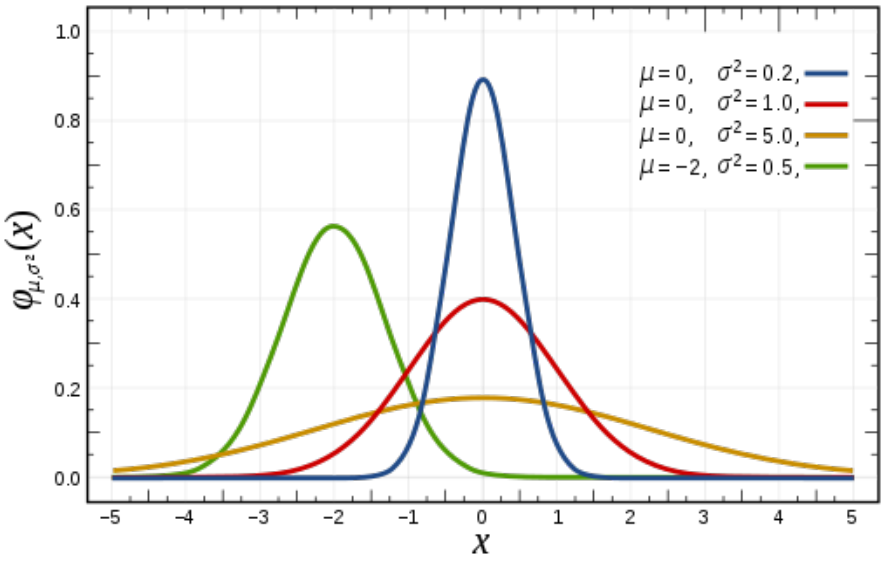


What about continuous variables?

- *Client*: What if I am measuring a **continuous variable**?
- ***You*: Let me tell you about Gaussians…**

$$P(x \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

```
# PDF of Gaussian
# (good candidate for formula sheet)
```



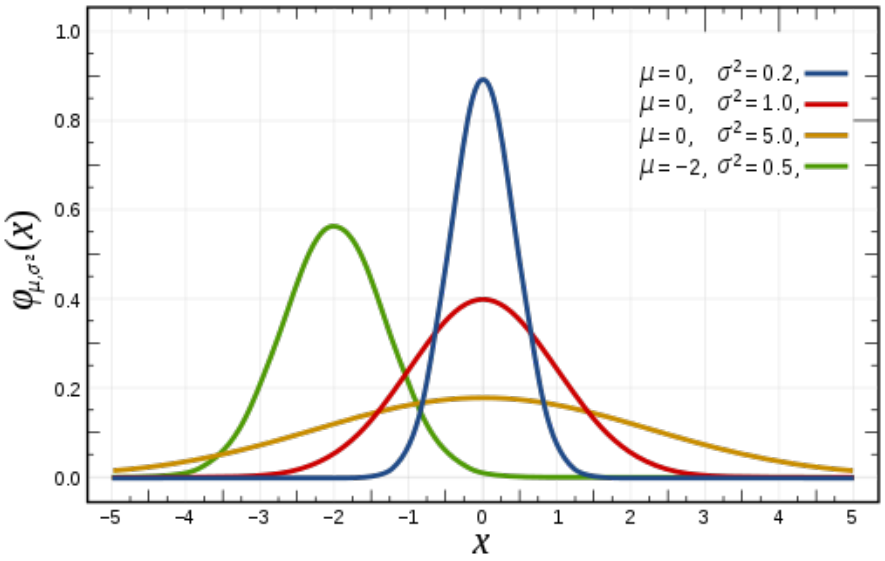
What about continuous variables?

- *Client*: What if I am measuring a **continuous variable**?
- ***You*: Let me tell you about Gaussians…**

$$P(x \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

PDF of Gaussian

(good candidate for formula sheet)



Given a set of i.i.d. samples from a Gaussian, fit what parameters?

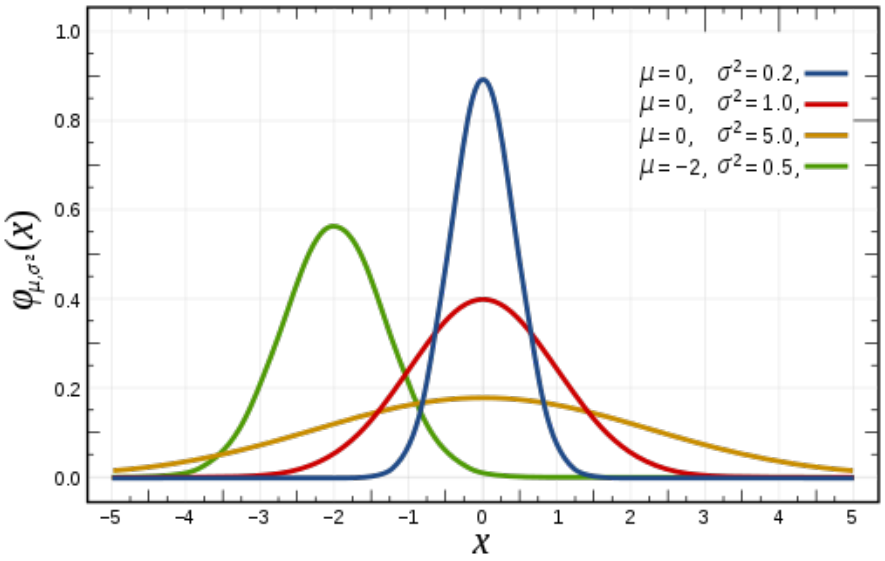
What about continuous variables?

- *Client*: What if I am measuring a **continuous variable**?
- ***You*: Let me tell you about Gaussians…**

$$P(x \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

PDF of Gaussian

(good candidate for formula sheet)



Given a set of i.i.d. samples from a Gaussian, fit what parameters?

$$\theta = [\mu, \sigma]$$

MLE for Gaussian

- Prob. of i.i.d. samples $D=\{x_1,\ldots,x_n\}$ (e.g., temperature):

$$P(\mathcal{D}|\mu,\sigma) = P(x_1,\ldots,x_n|\mu,\sigma)$$

$$= \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^n \prod_{i=1}^{n} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

- Log-likelihood of data:

$$\log P(\mathcal{D}|\mu,\sigma) = -n\log(\sigma\sqrt{2\pi}) - \sum_{i=1}^{n}\frac{(x_i-\mu)^2}{2\sigma^2}$$

- What is $\widehat{\theta}_{MLE}$ for $\theta = (\mu,\sigma^2)$? Draw a picture!

MLE for Gaussian

- Prob. of i.i.d. samples $D=\{x_1,\ldots,x_n\}$ (e.g., temperature):

$$P(\mathcal{D}|\mu,\sigma) = P(x_1,\ldots,x_n|\mu,\sigma)$$

$$= \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^n \prod_{i=1}^{n} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

Wait, why can I just multiply all samples together?

- Log-likelihood of data:

$$\log P(\mathcal{D}|\mu,\sigma) = -n\log(\sigma\sqrt{2\pi}) - \sum_{i=1}^{n}\frac{(x_i-\mu)^2}{2\sigma^2}$$

- What is $\widehat{\theta}_{MLE}$ for $\theta = (\mu,\sigma^2)$? Draw a picture!

MLE for Gaussian

- Prob. of i.i.d. samples $D=\{x_1,\ldots,x_n\}$ (e.g., temperature):

$$P(\mathcal{D}|\mu,\sigma) = P(x_1,\ldots,x_n|\mu,\sigma)$$
$$= \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^n \prod_{i=1}^{n} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

Wait, why can I just multiply all samples together?

→ By i.i.d. assumption

- Log-likelihood of data:

$$\log P(\mathcal{D}|\mu,\sigma) = -n\log(\sigma\sqrt{2\pi}) - \sum_{i=1}^{n}\frac{(x_i-\mu)^2}{2\sigma^2}$$

- What is $\widehat{\theta}_{MLE}$ for $\theta=(\mu,\sigma^2)$? Draw a picture!

MLE for Gaussian

- Prob. of i.i.d. samples $D=\{x_1,\ldots,x_n\}$ (e.g., temperature):

$$P(\mathcal{D}|\mu,\sigma) = P(x_1,\ldots,x_n|\mu,\sigma)$$

Likelihood:

$$= \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^n \prod_{i=1}^{n} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

Wait, why can I just multiply all samples together?

→ By i.i.d. assumption

- Log-likelihood of data:

$$\log P(\mathcal{D}|\mu,\sigma) = -n\log(\sigma\sqrt{2\pi}) - \sum_{i=1}^{n}\frac{(x_i-\mu)^2}{2\sigma^2}$$

- What is $\widehat{\theta}_{MLE}$ for $\theta = (\mu,\sigma^2)$? Draw a picture!

MLE for Gaussian

- Prob. of i.i.d. samples $D=\{x_1,\ldots,x_n\}$ (e.g., temperature):

$$P(\mathcal{D}|\mu,\sigma) = P(x_1,\ldots,x_n|\mu,\sigma)$$

Likelihood:

$$= \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^n \prod_{i=1}^{n} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

Wait, why can I just multiply all samples together?

→ By i.i.d. assumption

- Log-likelihood of data:

LL:

$$\log P(\mathcal{D}|\mu,\sigma) = -n\log(\sigma\sqrt{2\pi}) - \sum_{i=1}^{n}\frac{(x_i-\mu)^2}{2\sigma^2}$$

- What is $\widehat{\theta}_{MLE}$ for $\theta = (\mu,\sigma^2)$? Draw a picture!

MLE for Gaussian

Generate $\mathcal{D} = \{x_1, ..., x_n\}$, where

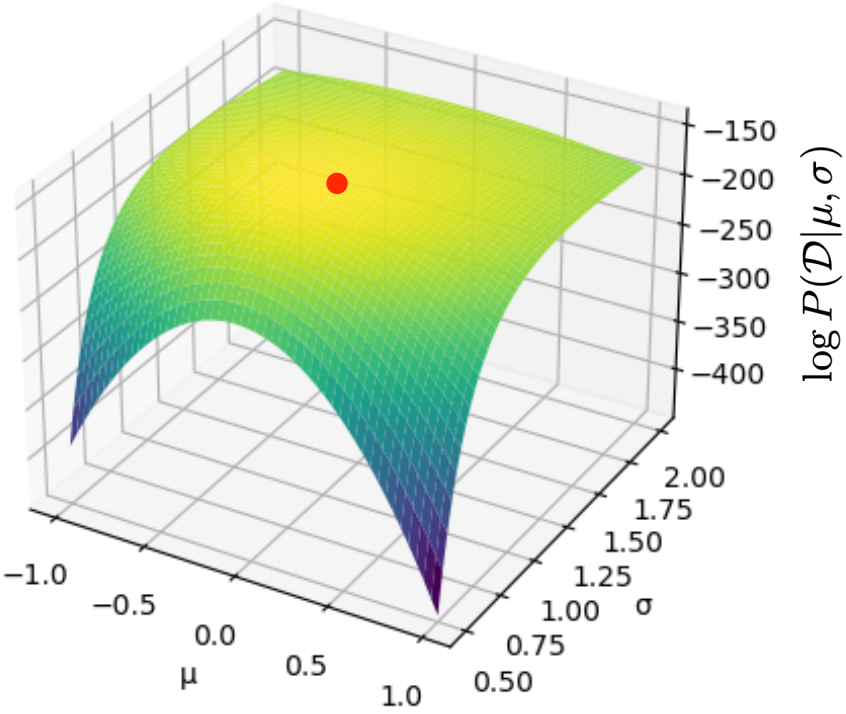
$$n = 100$$

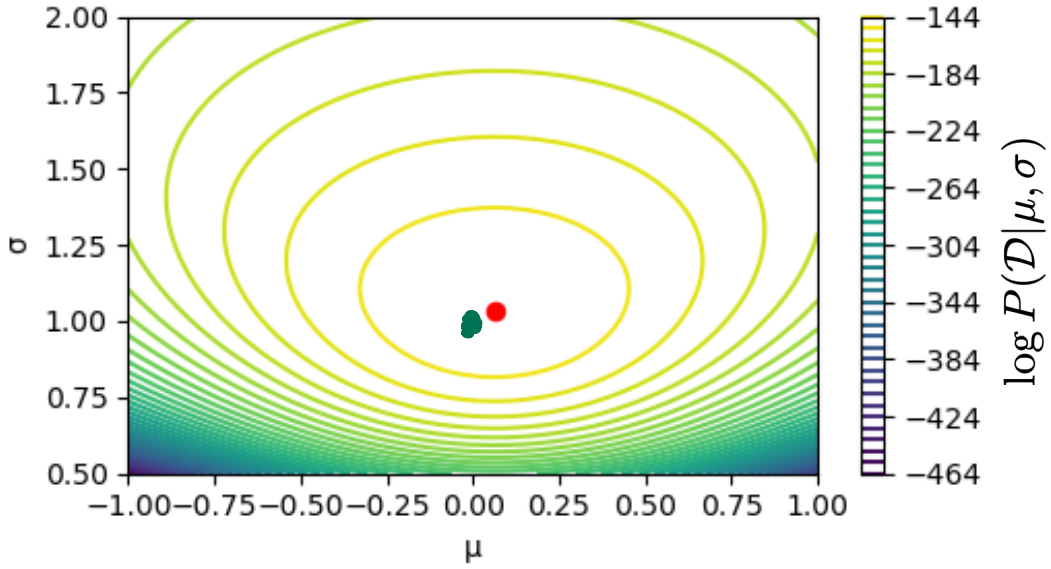
$$x_i \sim \mathcal{N}(\mu, \sigma^2)$$

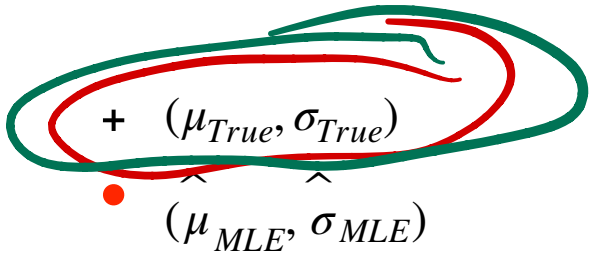
$$\mu = 0$$

$$\sigma^2 = 1$$

$$\log P(\mathcal{D}|\mu, \sigma) = -n \log(\sigma\sqrt{2\pi}) - \sum_{i=1}^{n} \frac{(x_i - \mu)^2}{2\sigma^2}$$







Your second learning algorithm: MLE for mean of a Gaussian

- What's MLE for mean? Set **partial derivative** to zero.

$$\frac{\partial}{\partial \mu} \log P(\mathcal{D} \mid \mu, \sigma) = \frac{\partial}{\partial \mu} \left[-n \log(\sigma\sqrt{2\pi}) - \sum_{i=1}^{n} \frac{(x_i - \mu)^2}{2\sigma^2} \right]$$

Your second learning algorithm: MLE for mean of a Gaussian

- What's MLE for mean? Set **partial derivative** to zero.

$$\frac{\partial}{\partial \mu} \log P(\mathcal{D} \mid \mu, \sigma) = \frac{\partial}{\partial \mu} \left[-n \log(\sigma\sqrt{2\pi}) - \sum_{i=1}^{n} \frac{(x_i - \mu)^2}{2\sigma^2} \right]$$

Your second learning algorithm: MLE for mean of a Gaussian

- What's MLE for mean? Set **partial derivative** to zero.

$$\frac{\partial}{\partial \mu} \log P(\mathcal{D} \mid \mu, \sigma) = \frac{\partial}{\partial \mu} \left[-n \log(\sigma\sqrt{2\pi}) - \sum_{i=1}^{n} \frac{(x_i - \mu)^2}{2\sigma^2} \right]$$

$$= -\sum_{i=1}^{n} \frac{-2(x_i - \mu)}{2\sigma^2}$$

Your second learning algorithm: MLE for mean of a Gaussian

- What's MLE for mean? Set **partial derivative** to zero.

$$\frac{\partial}{\partial \mu} \log P(\mathcal{D} \mid \mu, \sigma) = \frac{\partial}{\partial \mu} \left[-n \log(\sigma\sqrt{2\pi}) - \sum_{i=1}^{n} \frac{(x_i - \mu)^2}{2\sigma^2} \right]$$

$$= -\sum_{i=1}^{n} \frac{-2(x_i - \mu)}{2\sigma^2}$$

Your second learning algorithm: MLE for mean of a Gaussian

- What's MLE for mean? Set **partial derivative** to zero.

$$\frac{\partial}{\partial \mu} \log P(\mathcal{D} \mid \mu, \sigma) = \frac{\partial}{\partial \mu} \left[-n \log(\sigma\sqrt{2\pi}) - \sum_{i=1}^{n} \frac{(x_i - \mu)^2}{2\sigma^2} \right]$$

$$= \sum_{i=1}^{n} \frac{2(x_i - \mu)}{2\sigma^2}$$

Your second learning algorithm: MLE for mean of a Gaussian

- What's MLE for mean? Set **partial derivative** to zero.

$$\frac{\partial}{\partial \mu} \log P(\mathcal{D} \mid \mu, \sigma) = \frac{\partial}{\partial \mu} \left[-n \log(\sigma\sqrt{2\pi}) - \sum_{i=1}^{n} \frac{(x_i - \mu)^2}{2\sigma^2} \right]$$

$$= \sum_{i=1}^{n} \frac{2(x_i - \mu)}{2\sigma^2}$$

$$= \frac{-n\mu + \sum_{i=1}^{n} x_i}{\sigma^2} = 0$$

Your second learning algorithm: MLE for mean of a Gaussian

- What's MLE for mean? Set **partial derivative** to zero.

$$\frac{\partial}{\partial \mu} \log P(\mathcal{D} \mid \mu, \sigma) = \frac{\partial}{\partial \mu} \left[-n \log(\sigma\sqrt{2\pi}) - \sum_{i=1}^{n} \frac{(x_i - \mu)^2}{2\sigma^2} \right]$$

$$= \sum_{i=1}^{n} \frac{2(x_i - \mu)}{2\sigma^2}$$

$$= \frac{-n\mu + \sum_{i=1}^{n} x_i}{\sigma^2} = 0$$

$$\hat{\mu}_{\text{MLE}} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

Your second learning algorithm: MLE for mean of a Gaussian

- What's MLE for mean? Set **partial derivative** to zero.

$$\frac{\partial}{\partial \mu} \log P(\mathcal{D} \mid \mu, \sigma) = \frac{\partial}{\partial \mu} \left[-n \log(\sigma\sqrt{2\pi}) - \sum_{i=1}^{n} \frac{(x_i - \mu)^2}{2\sigma^2} \right]$$

$$= \sum_{i=1}^{n} \frac{2(x_i - \mu)}{2\sigma^2}$$

$$= \frac{-n\mu + \sum_{i=1}^{n} x_i}{\sigma^2} = 0$$

$$\hat{\mu}_{\text{MLE}} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

Your second learning algorithm: MLE for mean of a Gaussian

- What's MLE for mean? Set **partial derivative** to zero.

$$\frac{\partial}{\partial \mu} \log P(\mathcal{D} \mid \mu, \sigma) = \frac{\partial}{\partial \mu} \left[-n \log(\sigma\sqrt{2\pi}) - \sum_{i=1}^{n} \frac{(x_i - \mu)^2}{2\sigma^2} \right]$$

$$= \sum_{i=1}^{n} \frac{2(x_i - \mu)}{2\sigma^2}$$

$$= \frac{-n\mu + \sum_{i=1}^{n} x_i}{\sigma^2} = 0$$

$$\hat{\mu}_{\text{MLE}} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

empirical mean!

MLE for variance

- Again, set partial derivative to zero:

$$\frac{\partial}{\partial \sigma} \log P(\mathcal{D} \mid \mu, \sigma) = \frac{\partial}{\partial \sigma} \left[-n \log(\sigma\sqrt{2\pi}) - \sum_{i=1}^{n} \frac{(x_i - \mu)^2}{2\sigma^2} \right]$$

MLE for variance

- Again, set partial derivative to zero:

$$\frac{\partial}{\partial \sigma} \log P(\mathcal{D} \mid \mu, \sigma) = \frac{\partial}{\partial \sigma} \left[-n \log(\sigma\sqrt{2\pi}) - \sum_{i=1}^{n} \frac{(x_i - \mu)^2}{2\sigma^2} \right]$$

MLE for variance

reminders for formulas:

$$\frac{d}{dx}\log x = \frac{1}{x}$$

- Again, set partial derivative to zero:

$$\frac{\partial}{\partial \sigma} \log P(\mathcal{D} \mid \mu, \sigma) = \frac{\partial}{\partial \sigma} \left[-n \log(\sigma\sqrt{2\pi}) - \sum_{i=1}^{n} \frac{(x_i - \mu)^2}{2\sigma^2} \right]$$

MLE for variance

reminders for formulas:

$$\frac{d}{dx}\log x = \frac{1}{x} \qquad \frac{d}{dx}\frac{1}{x^2} = \frac{d}{dx}x^{-2} = -2x^{-3}$$

- Again, set partial derivative to zero:

$$\frac{\partial}{\partial \sigma} \log P(\mathcal{D} \mid \mu, \sigma) = \frac{\partial}{\partial \sigma}\left[-n \log(\sigma\sqrt{2\pi}) - \sum_{i=1}^{n} \frac{(x_i - \mu)^2}{2\sigma^2}\right]$$

MLE for variance

reminders for formulas:

$$\frac{d}{dx}\log x = \frac{1}{x} \qquad \frac{d}{dx}\frac{1}{x^2} = \frac{d}{dx}x^{-2} = -2x^{-3}$$

- Again, set partial derivative to zero:

$$\frac{\partial}{\partial \sigma}\log P(\mathcal{D} \mid \mu, \sigma) = \frac{\partial}{\partial \sigma}\left[-n\log(\sigma\sqrt{2\pi}) - \sum_{i=1}^{n}\frac{(x_i-\mu)^2}{2\sigma^2}\right]$$

$$= \frac{-n}{\sigma} + \sum_{i=1}^{n}\frac{2(x_i-\mu)^2}{2\sigma^3}$$

MLE for variance

reminders for formulas:

$$\frac{d}{dx}\log x = \frac{1}{x} \qquad \frac{d}{dx}\frac{1}{x^2} = \frac{d}{dx}x^{-2} = -2x^{-3}$$

- Again, set partial derivative to zero:

$$\frac{\partial}{\partial \sigma}\log P(\mathcal{D} \mid \mu, \sigma) = \frac{\partial}{\partial \sigma}\left[-n\log(\sigma\sqrt{2\pi}) - \sum_{i=1}^{n}\frac{(x_i-\mu)^2}{2\sigma^2}\right]$$

$$= \frac{-n}{\sigma} + \sum_{i=1}^{n}\frac{2(x_i-\mu)^2}{2\sigma^3}$$

MLE for variance

reminders for formulas:

$$\frac{d}{dx}\log x = \frac{1}{x} \qquad \frac{d}{dx}\frac{1}{x^2} = \frac{d}{dx}x^{-2} = -2x^{-3}$$

- Again, set partial derivative to zero:

$$\frac{\partial}{\partial\sigma}\log P(\mathcal{D} \mid \mu, \sigma) = \frac{\partial}{\partial\sigma}\left[-n\log(\sigma\sqrt{2\pi}) - \sum_{i=1}^{n}\frac{(x_i-\mu)^2}{2\sigma^2}\right]$$

$$= \frac{-n}{\sigma} + \sum_{i=1}^{n}\frac{2(x_i-\mu)^2}{2\sigma^3}$$

$$= \frac{-n\sigma^2 + \sum_{i=1}^{n}(x_i-\mu)^2}{\sigma^3}$$

MLE for variance

reminders for formulas:

$$\frac{d}{dx}\log x = \frac{1}{x} \qquad \frac{d}{dx}\frac{1}{x^2} = \frac{d}{dx}x^{-2} = -2x^{-3}$$

- Again, set partial derivative to zero:

$$\frac{\partial}{\partial\sigma}\log P(\mathcal{D} \mid \mu, \sigma) = \frac{\partial}{\partial\sigma}\left[-n\log(\sigma\sqrt{2\pi}) - \sum_{i=1}^{n}\frac{(x_i-\mu)^2}{2\sigma^2}\right]$$

$$= \frac{-n}{\sigma} + \sum_{i=1}^{n}\frac{2(x_i-\mu)^2}{2\sigma^3}$$

$$= \frac{-n\sigma^2 + \sum_{i=1}^{n}(x_i-\mu)^2}{\sigma^3} = 0$$

MLE for variance

reminders for formulas:

$$\frac{d}{dx}\log x = \frac{1}{x} \qquad \frac{d}{dx}\frac{1}{x^2} = \frac{d}{dx}x^{-2} = -2x^{-3}$$

- Again, set partial derivative to zero:

$$\frac{\partial}{\partial\sigma}\log P(\mathcal{D} \mid \mu, \sigma) = \frac{\partial}{\partial\sigma}\left[-n\log(\sigma\sqrt{2\pi}) - \sum_{i=1}^{n}\frac{(x_i-\mu)^2}{2\sigma^2}\right]$$

$$= \frac{-n}{\sigma} + \sum_{i=1}^{n}\frac{2(x_i-\mu)^2}{2\sigma^3}$$

$$= \frac{-n\sigma^2 + \sum_{i=1}^{n}(x_i-\mu)^2}{\sigma^3} = 0$$

MLE for variance

reminders for formulas:

$$\frac{d}{dx}\log x = \frac{1}{x} \qquad \frac{d}{dx}\frac{1}{x^2} = \frac{d}{dx}x^{-2} = -2x^{-3}$$

- Again, set partial derivative to zero:

$$\frac{\partial}{\partial\sigma}\log P(\mathcal{D} \mid \mu, \sigma) = \frac{\partial}{\partial\sigma}\left[-n\log(\sigma\sqrt{2\pi}) - \sum_{i=1}^{n}\frac{(x_i-\mu)^2}{2\sigma^2}\right]$$

$$= \frac{-n}{\sigma} + \sum_{i=1}^{n}\frac{2(x_i-\mu)^2}{2\sigma^3}$$

$$= \frac{-n\sigma^2 + \sum_{i=1}^{n}(x_i-\mu)^2}{\sigma^3} = 0$$

$$= \sigma^2_{\text{MLE}} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu}_{\text{MLE}})^2$$

MLE for variance

reminders for formulas:

$$\frac{d}{dx}\log x = \frac{1}{x} \qquad \frac{d}{dx}\frac{1}{x^2} = \frac{d}{dx}x^{-2} = -2x^{-3}$$

- Again, set partial derivative to zero:

$$\frac{\partial}{\partial\sigma}\log P(\mathcal{D} \mid \mu, \sigma) = \frac{\partial}{\partial\sigma}\left[-n\log(\sigma\sqrt{2\pi}) - \sum_{i=1}^{n}\frac{(x_i-\mu)^2}{2\sigma^2}\right]$$

$$= \frac{-n}{\sigma} + \sum_{i=1}^{n}\frac{2(x_i-\mu)^2}{2\sigma^3}$$

$$= \frac{-n\sigma^2 + \sum_{i=1}^{n}(x_i-\mu)^2}{\sigma^3} = 0$$

$$= \sigma^2_{\text{MLE}} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu}_{\text{MLE}})^2$$

sub in $\hat{\mu}_{\text{MLE}}$

MLE for variance

reminders for formulas:

$$\frac{d}{dx}\log x = \frac{1}{x} \qquad \frac{d}{dx}\frac{1}{x^2} = \frac{d}{dx}x^{-2} = -2x^{-3}$$

- Again, set partial derivative to zero:

$$\frac{\partial}{\partial\sigma}\log P(\mathcal{D} \mid \mu, \sigma) = \frac{\partial}{\partial\sigma}\left[-n\log(\sigma\sqrt{2\pi}) - \sum_{i=1}^{n}\frac{(x_i-\mu)^2}{2\sigma^2}\right]$$

$$= \frac{-n}{\sigma} + \sum_{i=1}^{n}\frac{2(x_i-\mu)^2}{2\sigma^3}$$

$$= \frac{-n\sigma^2 + \sum_{i=1}^{n}(x_i-\mu)^2}{\sigma^3} = 0$$

$$= \sigma^2_{\text{MLE}} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu}_{\text{MLE}})^2$$

sub in $\hat{\mu}_{\text{MLE}}$

Learning Gaussian parameters

- MLE:

$$\widehat{\mu}_{MLE} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\widehat{\sigma^2}_{MLE} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \widehat{\mu}_{MLE})^2$$

- MLE for the variance of a Gaussian is **biased**

$$\mathbb{E}[\widehat{\sigma^2}_{MLE}] \neq \sigma^2$$

- Unbiased variance estimator:

$$\widehat{\sigma^2}_{unbiased} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \widehat{\mu}_{MLE})^2$$

Maximum Likelihood Estimation

Observe $X_1, X_2, \ldots, X_n$ drawn IID from $f(x;\theta)$ for some "true" $\theta = \theta_*$

Likelihood function $L_n(\theta) = \prod_{i=1}^{n} f(X_i;\theta)$

Log-Likelihood function $l_n(\theta) = \log(L_n(\theta)) = \sum_{i=1}^{n} \log(f(X_i;\theta))$

Maximum Likelihood Estimator (MLE) $\widehat{\theta}_{MLE} = \arg\max_{\theta} L_n(\theta)$

Properties (under benign regularity conditions—smoothness, identifiability, etc.):

- Asymptotically consistent and normal: $\frac{\widehat{\theta}_{MLE} - \theta_*}{\widehat{se}} \sim \mathcal{N}(0,1)$
- Asymptotic Optimality, minimum variance (see Cramer-Rao lower bound)

The MLE is a "recipe" that begins with a *model* for data $f(x;\theta)$

Linear Regression



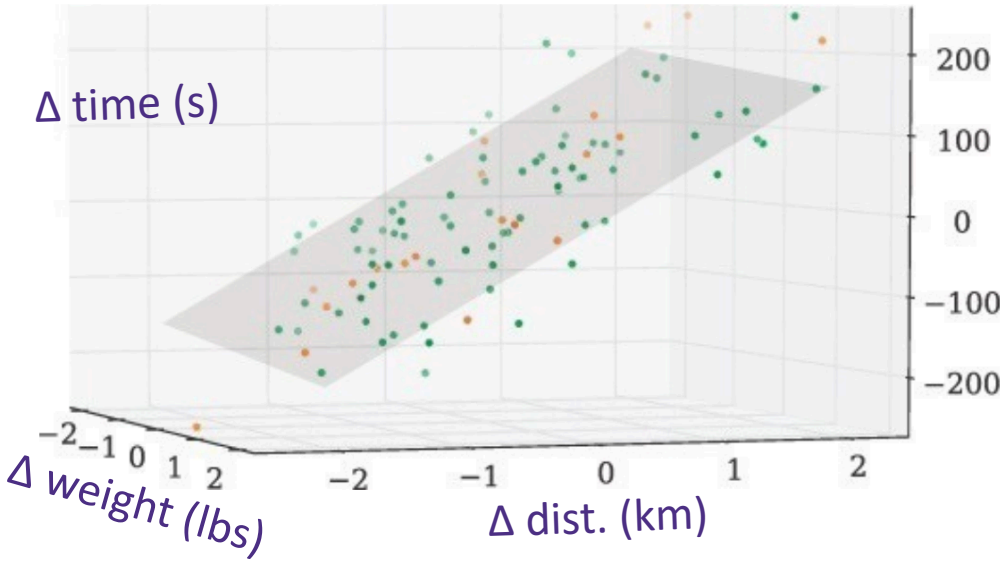
The regression problem, d-dimensions



Centered potato delivery logs, predict:

***y* = Δ delivery time (s)** *from*

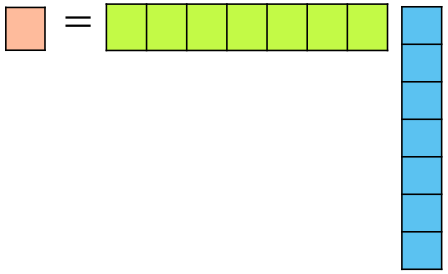
***x* = {distance, traffic, potato weight, etc.}**



Training Data: $\{(x_i, y_i)\}_{i=1}^{n}$ $x_i \in \mathbb{R}^d$ $y_i \in \mathbb{R}$

Hypothesis/Model: linear

$$y_i = x_i^T w + \epsilon_i \qquad \epsilon_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2)$$



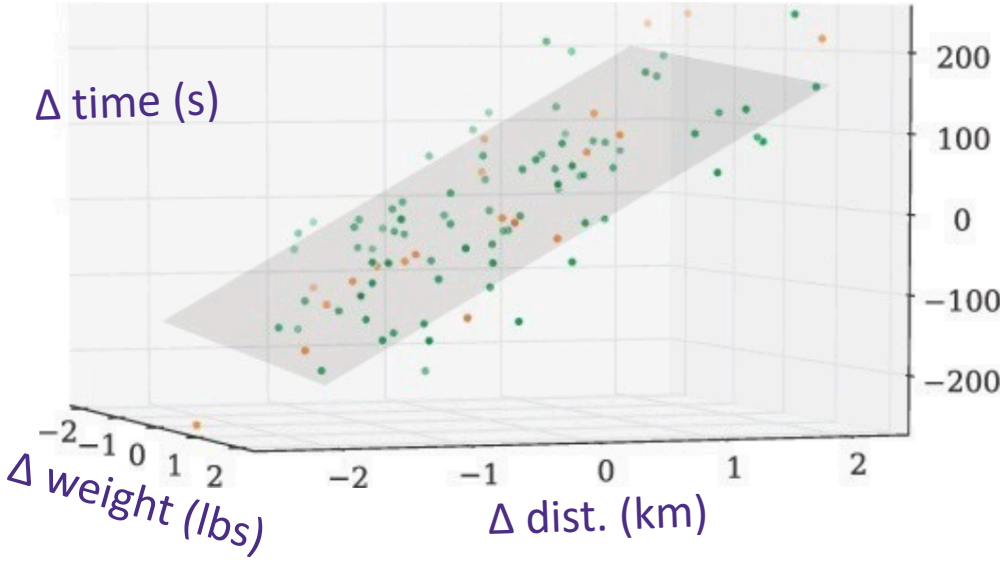
The regression problem, d-dimensions

Centered potato delivery logs, predict:

***y* = Δ delivery time (s)** *from*

***x* = {distance, traffic, potato weight, etc.}** $x \in \mathbb{R}^d$



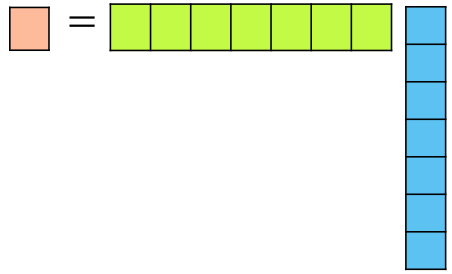


Training Data:

$$\{(x_i, y_i)\}_{i=1}^{n} \qquad x_i \in \mathbb{R}^d \quad y_i \in \mathbb{R}$$

Hypothesis/Model: linear

$$y_i = x_i^T w + \epsilon_i \qquad \epsilon_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2)$$



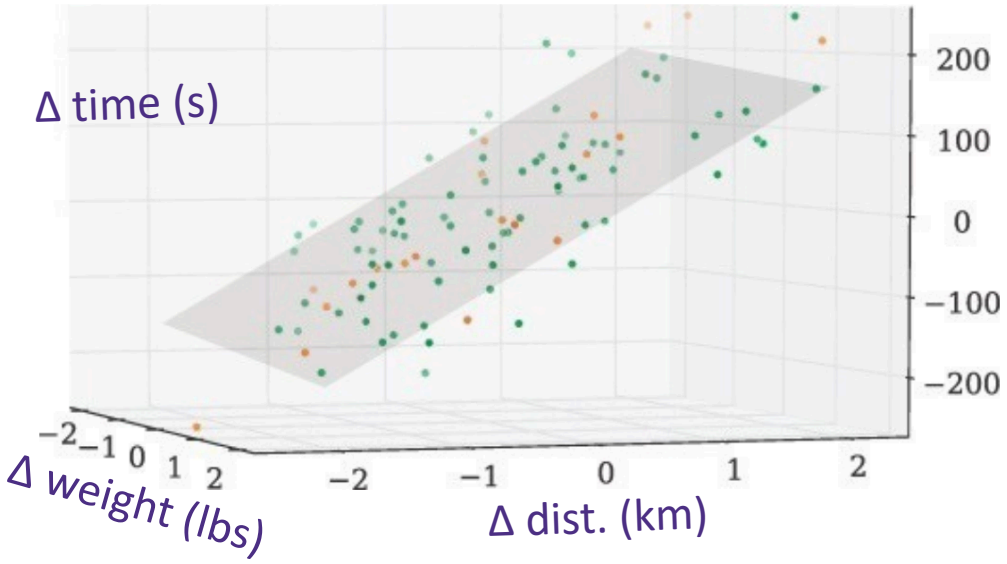
The regression problem, d-dimensions



Centered potato delivery logs, predict:

***y* = Δ delivery time (s)** *from*

***x* = {distance, traffic, potato weight, etc.}** $x \in \mathbb{R}^d$

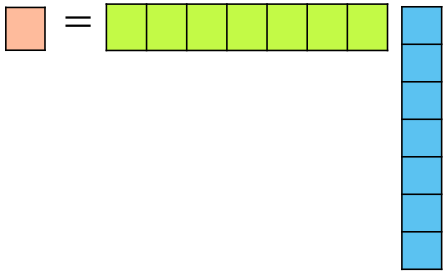


Training Data: $\{(x_i, y_i)\}_{i=1}^n$ $x_i \in \mathbb{R}^d$ $y_i \in \mathbb{R}$

Hypothesis/Model: linear

$$y_i = x_i^T w + \epsilon_i \qquad \epsilon_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2)$$

$x \in \mathbb{R}^d$



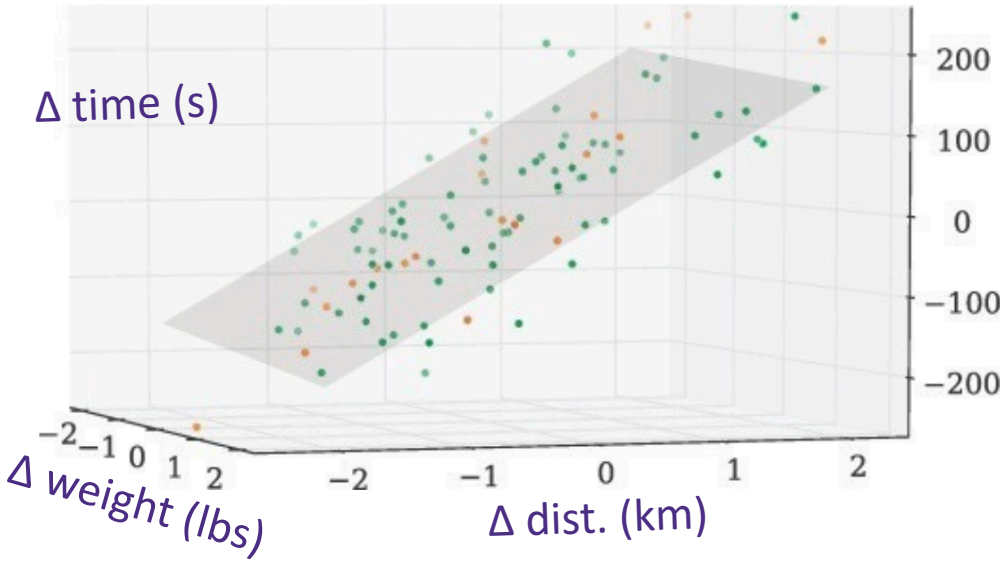
The regression problem, d-dimensions



Centered potato delivery logs, predict:

***y* = Δ delivery time (s)** *from*

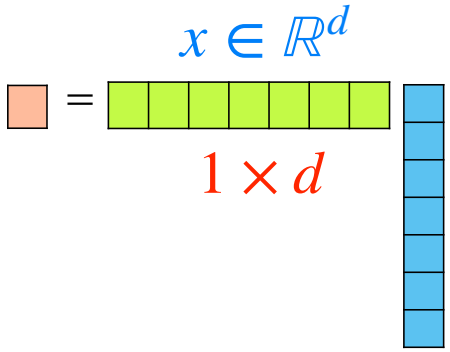
***x* = {distance, traffic, potato weight, etc.}** $x \in \mathbb{R}^d$



Training Data: $\{(x_i, y_i)\}_{i=1}^n$ $x_i \in \mathbb{R}^d$ $y_i \in \mathbb{R}$

Hypothesis/Model: linear

$$y_i = x_i^T w + \epsilon_i \qquad \epsilon_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2)$$



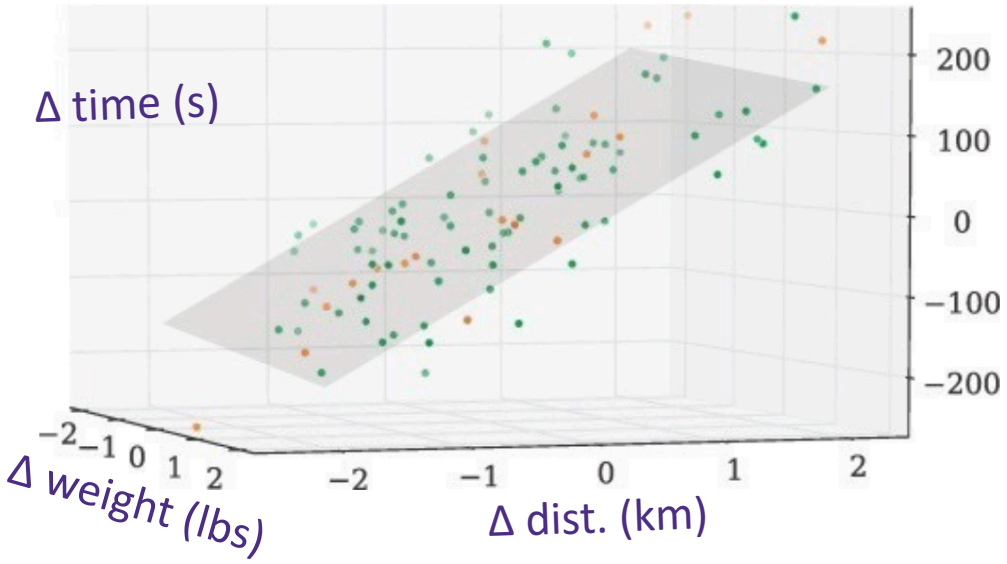
The regression problem, d-dimensions



Centered potato delivery logs, predict:

***y* = Δ delivery time (s)** *from*

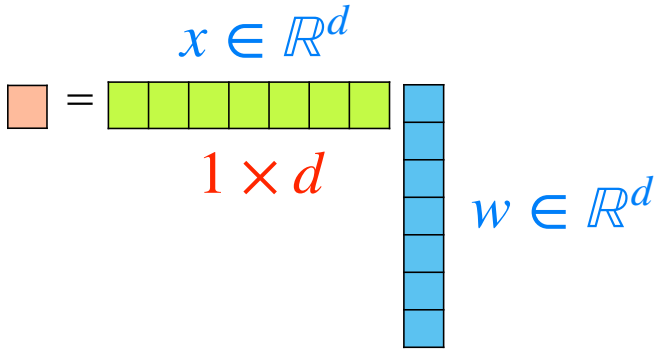
***x* = {distance, traffic, potato weight, etc.}** $x \in \mathbb{R}^d$



Training Data: $\{(x_i, y_i)\}_{i=1}^n$ $x_i \in \mathbb{R}^d$ $y_i \in \mathbb{R}$

Hypothesis/Model: linear

$$y_i = x_i^T w + \epsilon_i \qquad \epsilon_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2)$$



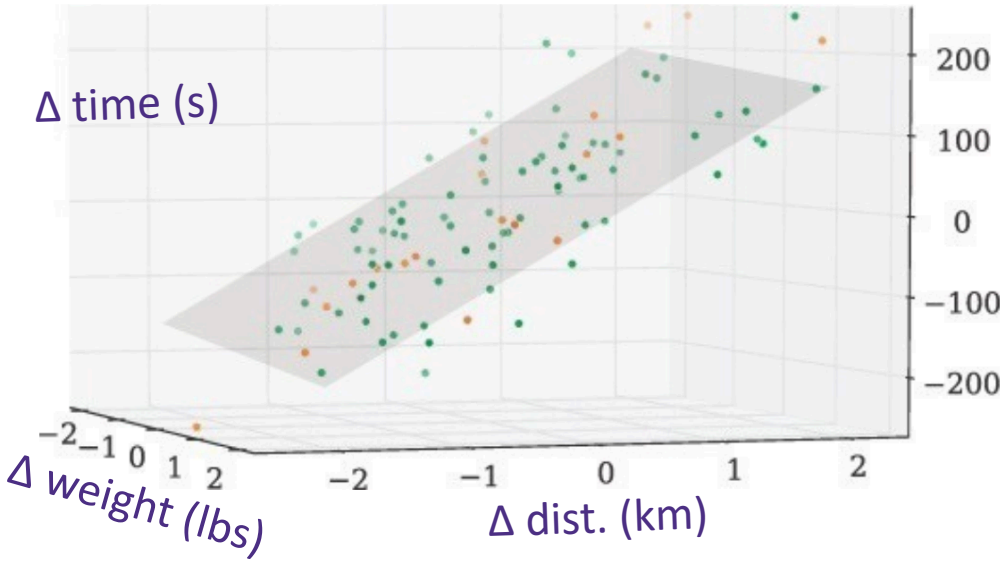
The regression problem, d-dimensions



Centered potato delivery logs, predict:

***y* = Δ delivery time (s)** *from*

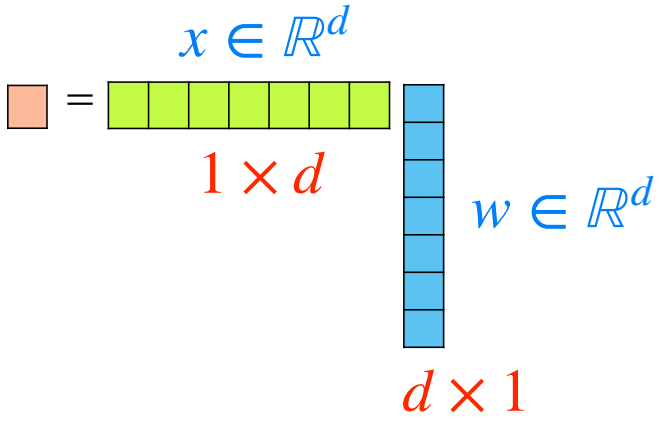
***x* = {distance, traffic, potato weight, etc.}** $x \in \mathbb{R}^d$



Training Data: $\{(x_i, y_i)\}_{i=1}^n$ $x_i \in \mathbb{R}^d$ $y_i \in \mathbb{R}$

Hypothesis/Model: linear

$$y_i = x_i^T w + \epsilon_i \qquad \epsilon_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2)$$



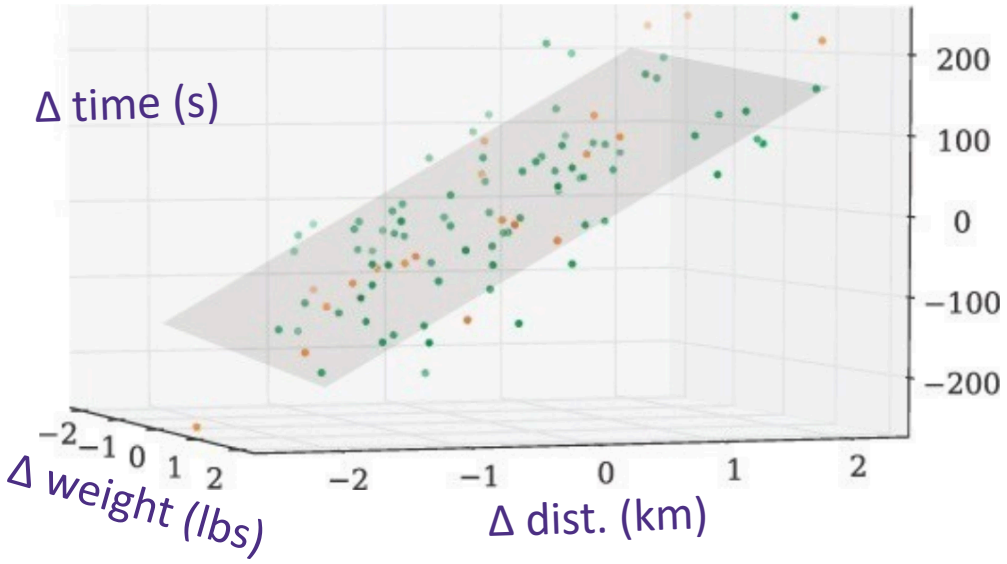
The regression problem, d-dimensions



Centered potato delivery logs, predict:

***y* = Δ delivery time (s)** *from*

***x* = {distance, traffic, potato weight, etc.}** $x \in \mathbb{R}^d$



Training Data: $\{(x_i, y_i)\}_{i=1}^n$ $x_i \in \mathbb{R}^d$ $y_i \in \mathbb{R}$

Hypothesis/Model: linear

$$y_i = x_i^T w + \epsilon_i \qquad \epsilon_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2)$$

$x \in \mathbb{R}^d$

$y \in \mathbb{R}$ *scalar*

$1 \times d$

$w \in \mathbb{R}^d$

$d \times 1$

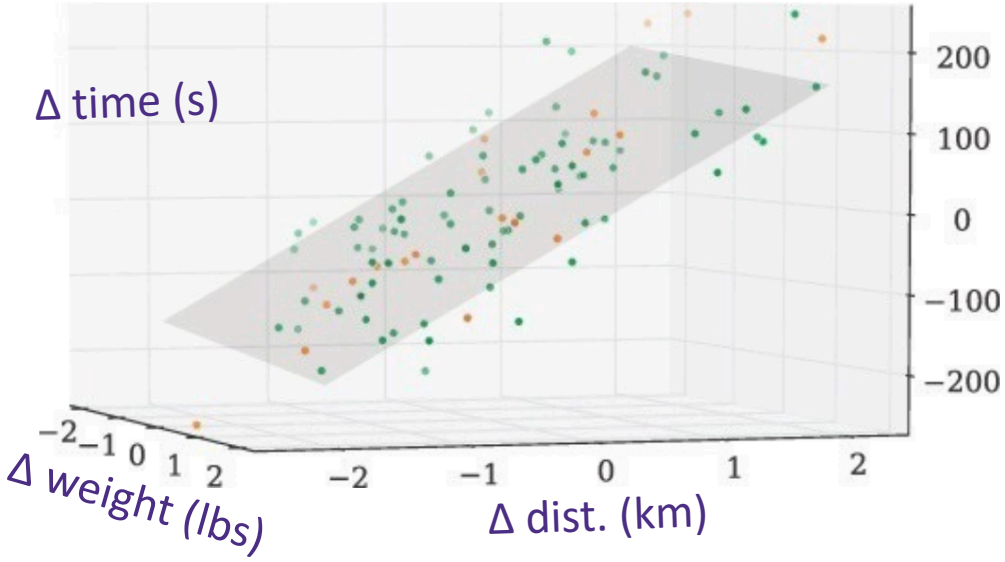
The regression problem, d-dim

Centered potato delivery logs, predict:

***y* = Δ delivery time (s)** *from*

***x* = {distance, traffic, potato weight, etc.}**





Training Data: $\{(x_i, y_i)\}_{i=1}^{n}$ $x_i \in \mathbb{R}^d$ $y_i \in \mathbb{R}$

Hypothesis/Model: linear

$$y_i = x_i^T w + \epsilon_i$$

$$\epsilon_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2)$$

$$p(y|x, w, \sigma) =$$

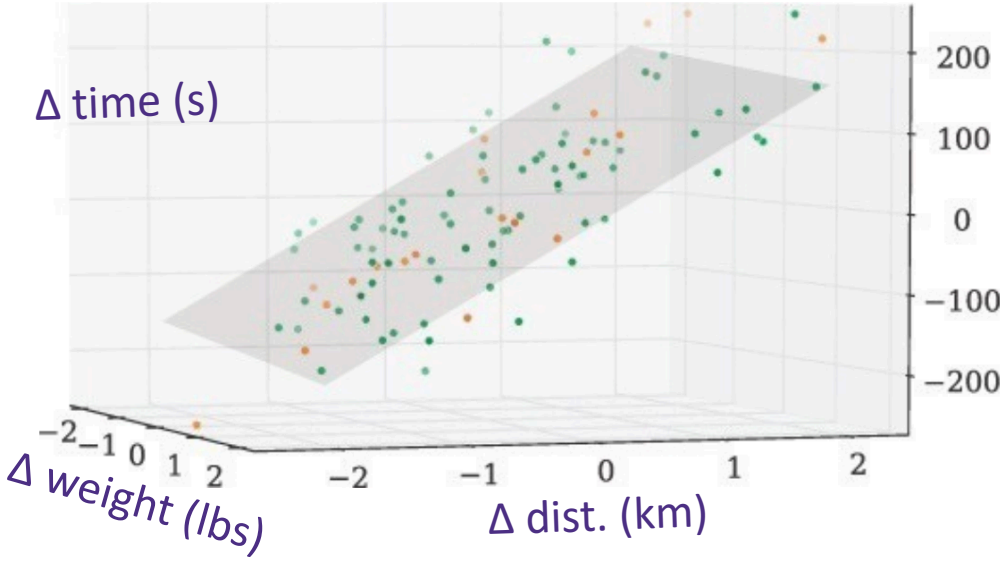
The regression problem, d-dim

Centered potato delivery logs, predict:

***y* = Δ delivery time (s)** *from*

***x* = {distance, traffic, potato weight, etc.}**





Training Data: $\{(x_i, y_i)\}_{i=1}^n$ $x_i \in \mathbb{R}^d$ $y_i \in \mathbb{R}$

Hypothesis/Model: linear

$$y_i = x_i^T w + \epsilon_i \qquad \epsilon_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2)$$

$$p(y|x, w, \sigma) = \mathcal{N}(x^T w, \sigma^2)$$

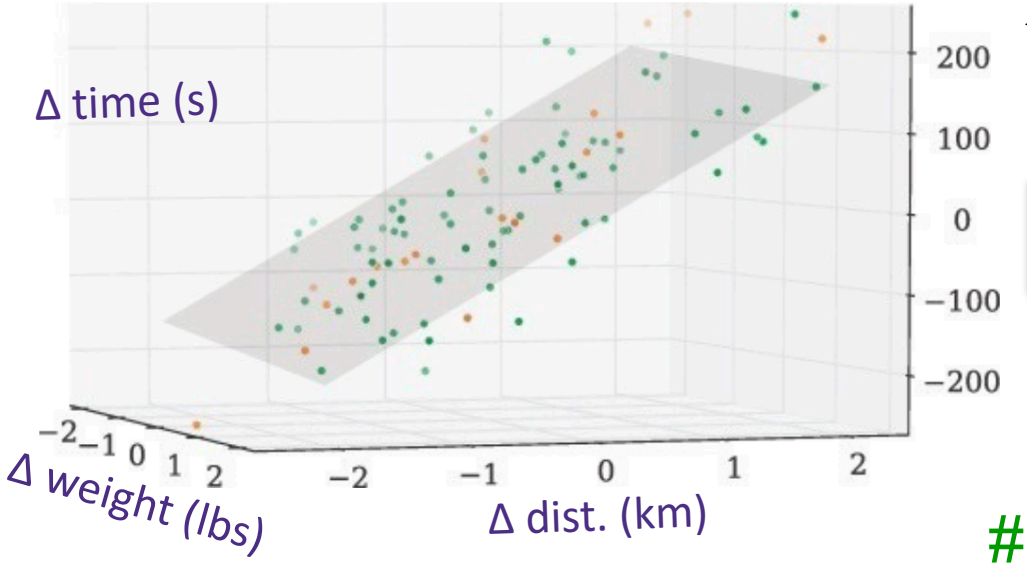
The regression problem, d-dim

Centered potato delivery logs, predict:

***y* = Δ delivery time (s)** *from*

***x* = {distance, traffic, potato weight, etc.}**





Training Data: $\{(x_i, y_i)\}_{i=1}^{n}$ $x_i \in \mathbb{R}^d$ $y_i \in \mathbb{R}$

Hypothesis/Model: linear

$$y_i = x_i^T w + \epsilon_i \qquad \epsilon_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2)$$

$$p(y|x, w, \sigma) = \mathcal{N}(x^T w, \sigma^2)$$

go back to formula sheet, retrieve PDF of Gaussian…

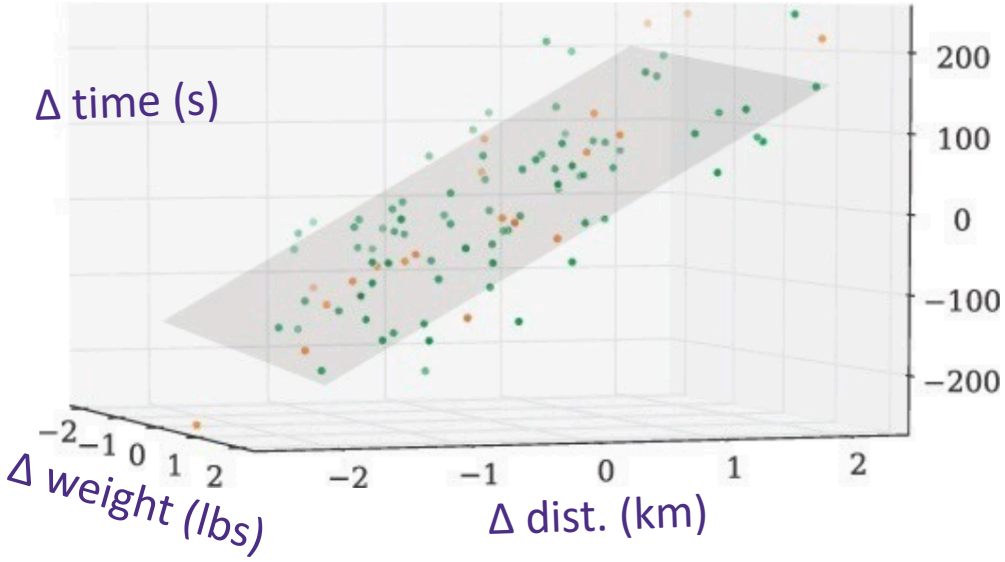
The regression problem, d-dim

Centered potato delivery logs, predict:

***y* = Δ delivery time (s)** *from*

***x* = {distance, traffic, potato weight, etc.}**





Training Data: $\{(x_i, y_i)\}_{i=1}^n$ $x_i \in \mathbb{R}^d$ $y_i \in \mathbb{R}$

Hypothesis/Model: linear

$$y_i = x_i^T w + \epsilon_i \qquad \epsilon_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2)$$

$$p(y|x, w, \sigma) = \mathcal{N}(x^T w, \sigma^2)$$

$$p(y|x, w, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y - x^\top w)^2 / 2\sigma^2}$$

What are we optimizing for then?

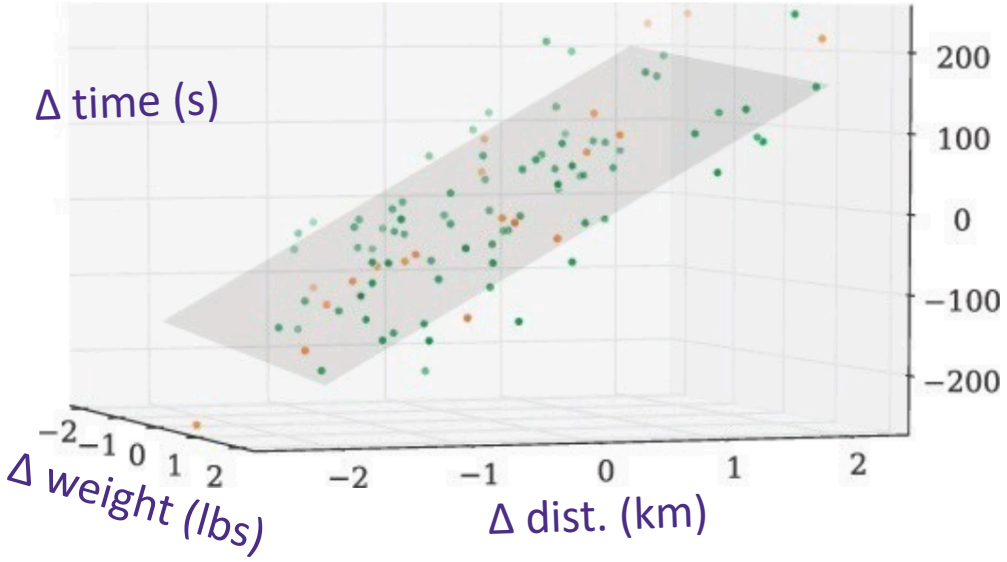
We discussed minimizing squared error/finding the best linear fit

Loss: least squares solution

$$\min_w \sum_{i=1}^{n} (y_i - x_i w)^2$$

But we could also do max likelihood!

$$p(y|x, w, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y - x^\top w)^2 / 2\sigma^2}$$



Maximizing log-likelihood for linear regression

Training Data: $\{(x_i, y_i)\}_{i=1}^n$ $x_i \in \mathbb{R}^d$ $y_i \in \mathbb{R}$

$$p(y|x, w, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y - x^\top w)^2/2\sigma^2}$$

Likelihood: $$P(\mathcal{D}|w, \sigma) = \prod_{i=1}^n p(y_i|x_i, w, \sigma) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y_i - x_i^\top w)^2/2\sigma^2}$$

Maximizing log-likelihood for linear regression

Training Data: $\{(x_i, y_i)\}_{i=1}^n$ $x_i \in \mathbb{R}^d$ $y_i \in \mathbb{R}$

$$p(y|x, w, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y - x^\top w)^2/2\sigma^2}$$

Likelihood: $$P(\mathcal{D}|w, \sigma) = \prod_{i=1}^n p(y_i|x_i, w, \sigma) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y_i - x_i^\top w)^2/2\sigma^2}$$

Yay, we have a new likelihood! Now, how do we find MLE?

Maximizing log-likelihood, step 1

Training Data: $\{(x_i, y_i)\}_{i=1}^n$ $x_i \in \mathbb{R}^d$ $y_i \in \mathbb{R}$

$$p(y|x, w, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y - x^\top w)^2/2\sigma^2}$$

Likelihood: $$P(\mathcal{D}|w, \sigma) = \prod_{i=1}^n p(y_i|x_i, w, \sigma) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y_i - x_i^\top w)^2/2\sigma^2}$$

$$\log P(\mathcal{D}|w, \sigma) = \log \left(\prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y_i - x_i^\top w)^2/2\sigma^2} \right)$$ # take log

Maximizing log-likelihood, step 1

Training Data: $\{(x_i, y_i)\}_{i=1}^n$ $x_i \in \mathbb{R}^d$ $y_i \in \mathbb{R}$

$$p(y|x, w, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y - x^\top w)^2/2\sigma^2}$$

Likelihood: $$P(\mathcal{D}|w, \sigma) = \prod_{i=1}^n p(y_i|x_i, w, \sigma) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y_i - x_i^\top w)^2/2\sigma^2}$$

Maximize (wrt w): $$\log P(\mathcal{D}|w, \sigma) = \log \left(\prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y_i - x_i^\top w)^2/2\sigma^2} \right)$$ # take log

Maximizing log-likelihood, step 1

Training Data: $\{(x_i, y_i)\}_{i=1}^n$ $x_i \in \mathbb{R}^d$ $y_i \in \mathbb{R}$

$$p(y|x, w, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y - x^\top w)^2/2\sigma^2}$$

Likelihood: $$P(\mathcal{D}|w, \sigma) = \prod_{i=1}^n p(y_i|x_i, w, \sigma) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y_i - x_i^\top w)^2/2\sigma^2}$$

Maximize (wrt w): $$\log P(\mathcal{D}|w, \sigma) = \log \left(\prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y_i - x_i^\top w)^2/2\sigma^2} \right)$$ # take log

first manipulate log likelihood to make it easier to work with

Maximizing log-likelihood, step 1

Training Data: $\{(x_i, y_i)\}_{i=1}^n$ $x_i \in \mathbb{R}^d$ $y_i \in \mathbb{R}$

$$p(y|x, w, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y - x^\top w)^2/2\sigma^2}$$

Likelihood: $$P(\mathcal{D}|w, \sigma) = \prod_{i=1}^n p(y_i|x_i, w, \sigma) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y_i - x_i^\top w)^2/2\sigma^2}$$

Maximize (wrt w): $$\log P(\mathcal{D}|w, \sigma) = \log \left(\prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y_i - x_i^\top w)^2/2\sigma^2} \right)$$ # take log

first manipulate log likelihood to make it easier to work with

what is log of product?

Maximizing log-likelihood, step 1

Training Data: $\{(x_i, y_i)\}_{i=1}^n$ $x_i \in \mathbb{R}^d$ $y_i \in \mathbb{R}$

$$p(y|x, w, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y - x^\top w)^2/2\sigma^2}$$

Likelihood: $$P(\mathcal{D}|w, \sigma) = \prod_{i=1}^n p(y_i|x_i, w, \sigma) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y_i - x_i^\top w)^2/2\sigma^2}$$

Maximize (wrt w): $$\log P(\mathcal{D}|w, \sigma) = \log \left(\prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y_i - x_i^\top w)^2/2\sigma^2} \right)$$ # take log

first manipulate log likelihood to make it easier to work with

what is log of product?

$\log AB =$

Maximizing log-likelihood, step 1

Training Data: $\{(x_i, y_i)\}_{i=1}^n$ $x_i \in \mathbb{R}^d$ $y_i \in \mathbb{R}$

$$p(y|x, w, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y - x^\top w)^2/2\sigma^2}$$

Likelihood: $$P(\mathcal{D}|w, \sigma) = \prod_{i=1}^n p(y_i|x_i, w, \sigma) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y_i - x_i^\top w)^2/2\sigma^2}$$

Maximize (wrt w): $$\log P(\mathcal{D}|w, \sigma) = \log \left(\prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y_i - x_i^\top w)^2/2\sigma^2} \right)$$ # take log

first manipulate log likelihood to make it easier to work with

what is log of product?

$$\log AB = \log A + \log B$$

Maximizing log-likelihood, step 1

Training Data: $\{(x_i, y_i)\}_{i=1}^n$ $x_i \in \mathbb{R}^d$ $y_i \in \mathbb{R}$

$$p(y|x, w, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y - x^\top w)^2/2\sigma^2}$$

Likelihood: $$P(\mathcal{D}|w, \sigma) = \prod_{i=1}^n p(y_i|x_i, w, \sigma) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y_i - x_i^\top w)^2/2\sigma^2}$$

Maximize (wrt w): $$\log P(\mathcal{D}|w, \sigma) = \log \left(\prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y_i - x_i^\top w)^2/2\sigma^2} \right)$$ # take log

first manipulate log likelihood to make it easier to work with

$$= \sum_{i=1}^n \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(- \frac{(y_i - x_i^T w)^2}{2\sigma^2} \right) \right]$$

what is log of product?

$$\log AB = \log A + \log B$$

Maximizing log-likelihood, step 2

Maximize (wrt w): $\log P(\mathcal{D}|w,\sigma) = \log\left(\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y_i - x_i^\top w)^2/2\sigma^2}\right)$ # take log

first manipulate log likelihood to make it easier to work with

$$= \sum_{i=1}^{n} \log\left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - x_i^T w)^2}{2\sigma^2}\right)\right]$$

Maximizing log-likelihood, step 2

Maximize (wrt w): $\log P(\mathcal{D}|w,\sigma) = \log\left(\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y_i - x_i^\top w)^2/2\sigma^2}\right)$ # take log

first manipulate log likelihood to make it easier to work with

$$= \sum_{i=1}^{n} \log\left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - x_i^T w)^2}{2\sigma^2}\right)\right]$$

pull out terms that don't depend on i

Maximizing log-likelihood, step 2

Maximize (wrt w): $\log P(\mathcal{D}|w,\sigma) = \log\left(\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y_i - x_i^\top w)^2/2\sigma^2}\right)$ # take log

first manipulate log likelihood to make it easier to work with

$$= \sum_{i=1}^{n} \log\left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - x_i^T w)^2}{2\sigma^2}\right)\right]$$

$$= n\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \sum_{i=1}^{n} \frac{(y_i - x_i^T w)^2}{2\sigma^2}$$

pull out terms that don't depend on i

Maximizing log-likelihood, step 2

Maximize (wrt w): $\log P(\mathcal{D}|w,\sigma) = \log\left(\prod_{i=1}^{n}\frac{1}{\sqrt{2\pi\sigma^2}}e^{-(y_i-x_i^\top w)^2/2\sigma^2}\right)$ # take log

first manipulate log likelihood to make it easier to work with

$$= \sum_{i=1}^{n}\log\left[\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(y_i - x_i^T w)^2}{2\sigma^2}\right)\right]$$

$$= n\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \sum_{i=1}^{n}\frac{(y_i - x_i^T w)^2}{2\sigma^2}$$

pull out terms that don't depend on i

we're going to max wrt w, so

Maximizing log-likelihood, step 2

Maximize (wrt w): $\log P(\mathcal{D}|w,\sigma) = \log\left(\prod_{i=1}^{n}\frac{1}{\sqrt{2\pi\sigma^2}}e^{-(y_i-x_i^\top w)^2/2\sigma^2}\right)$ # take log

first manipulate log likelihood to make it easier to work with

$$= \sum_{i=1}^{n}\log\left[\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(y_i - x_i^T w)^2}{2\sigma^2}\right)\right]$$

$$= n\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \sum_{i=1}^{n}\frac{(y_i - x_i^T w)^2}{2\sigma^2}$$

pull out terms that don't depend on i

we're going to max wrt w, so

get rid of terms that don't depend on w

Maximizing log-likelihood, step 2

Maximize (wrt w): $\log P(\mathcal{D}|w,\sigma) = \log\left(\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y_i - x_i^\top w)^2/2\sigma^2}\right)$ # take log

first manipulate log likelihood to make it easier to work with

$$= \sum_{i=1}^{n} \log\left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - x_i^T w)^2}{2\sigma^2}\right)\right]$$

$$= n\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \sum_{i=1}^{n} \frac{(y_i - x_i^T w)^2}{2\sigma^2}$$

pull out terms that don't depend on i

we're going to max wrt w, so

get rid of terms that don't depend on w

Maximizing log-likelihood, step 2

Maximize (wrt w): $\log P(\mathcal{D}|w,\sigma) = \log\left(\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y_i - x_i^\top w)^2/2\sigma^2}\right)$ # take log

first manipulate log likelihood to make it easier to work with

$$= \sum_{i=1}^{n} \log\left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - x_i^T w)^2}{2\sigma^2}\right)\right]$$

$$= n\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \sum_{i=1}^{n} \frac{(y_i - x_i^T w)^2}{2\sigma^2}$$

pull out terms that don't depend on i

we're going to max wrt w, so

get rid of terms that don't depend on w

$$- \sum_{i=1}^{n} \frac{(y_i - x_i^T w)^2}{2\sigma^2}$$

Maximizing log-likelihood, step 2

Maximize (wrt w): $\log P(\mathcal{D}|w,\sigma) = \log\left(\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y_i - x_i^\top w)^2/2\sigma^2}\right)$ # take log

first manipulate log likelihood to make it easier to work with

$$= \sum_{i=1}^{n} \log\left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - x_i^T w)^2}{2\sigma^2}\right)\right]$$

$$= n\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \sum_{i=1}^{n} \frac{(y_i - x_i^T w)^2}{2\sigma^2}$$

pull out terms that don't depend on i

we're going to max wrt w, so
get rid of terms that don't depend on w

$$- \sum_{i=1}^{n} \frac{(y_i - x_i^T w)^2}{2\sigma^2}$$

We found our simplified LL!

Maximizing log-likelihood, step 2

Maximize (wrt w): $\log P(\mathcal{D}|w,\sigma) = \log\left(\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y_i - x_i^\top w)^2/2\sigma^2}\right)$ # take log

first manipulate log likelihood to make it easier to work with

$$= \sum_{i=1}^{n} \log\left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - x_i^T w)^2}{2\sigma^2}\right)\right]$$

$$= n\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \sum_{i=1}^{n} \frac{(y_i - x_i^T w)^2}{2\sigma^2}$$

pull out terms that don't depend on i

we're going to max wrt w, so
get rid of terms that don't depend on w

$$-\sum_{i=1}^{n} \frac{(y_i - x_i^T w)^2}{2\sigma^2}$$

We found our simplified LL!

Now wat do?

Maximizing log-likelihood, step 2

Maximize (wrt w): $\log P(\mathcal{D}|w,\sigma) = \log\left(\prod_{i=1}^{n}\frac{1}{\sqrt{2\pi\sigma^2}}e^{-(y_i-x_i^\top w)^2/2\sigma^2}\right)$ # take log

first manipulate log likelihood to make it easier to work with

$$= \sum_{i=1}^{n}\log\left[\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(y_i - x_i^T w)^2}{2\sigma^2}\right)\right]$$

$$= n\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \sum_{i=1}^{n}\frac{(y_i - x_i^T w)^2}{2\sigma^2}$$

pull out terms that don't depend on i

we're going to max wrt w, so

get rid of terms that don't depend on w

$$\arg\max_{w} \; -\sum_{i=1}^{n}\frac{(y_i - x_i^T w)^2}{2\sigma^2}$$

We found our simplified LL!

Now wat do?

Maximizing log-likelihood, step 2

Maximize (wrt w): $\log P(\mathcal{D}|w,\sigma) = \log\left(\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y_i - x_i^\top w)^2/2\sigma^2}\right)$ # take log

first manipulate log likelihood to make it easier to work with

$$= \sum_{i=1}^{n} \log\left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - x_i^T w)^2}{2\sigma^2}\right)\right]$$

$$= n\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \sum_{i=1}^{n} \frac{(y_i - x_i^T w)^2}{2\sigma^2}$$

pull out terms that don't depend on i

we're going to max wrt w, so
get rid of terms that don't depend on w

$$\arg\max_w \; -\sum_{i=1}^{n} \frac{(y_i - x_i^T w)^2}{2\sigma^2}$$

We found our simplified LL!

Now wat do?

Maximizing a negative?

Maximizing log-likelihood, optimization

Training Data: $\{(x_i, y_i)\}_{i=1}^n$ $\quad x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$

$$p(y|x, w, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y - x^\top w)^2/2\sigma^2}$$

Likelihood: $$P(\mathcal{D}|w, \sigma) = \prod_{i=1}^n p(y_i|x_i, w, \sigma) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y_i - x_i^\top w)^2/2\sigma^2}$$

Maximize (wrt w): $$\log P(\mathcal{D}|w, \sigma) = \log \left(\prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y_i - x_i^\top w)^2/2\sigma^2} \right)$$

$$\widehat{w}_{MLE} = \arg\min_w \sum_{i=1}^n (y_i - x_i^\top w)^2$$

Maximizing log-likelihood → minimizing mean squared error (MSE)

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

Maximizing log-likelihood → minimizing mean squared error (MSE)

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

- Wow, minimizing squared error seems like a reasonable thing to do to fit a predictor!

Maximizing log-likelihood → minimizing mean squared error (MSE)

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

- Wow, minimizing squared error seems like a reasonable thing to do to fit a predictor!
  - When noise is Gaussian, maximizing likelihood of our linear model is equivalent to minimizing the sum of squared errors. Simply a result of derivations

Maximizing log-likelihood → minimizing mean squared error (MSE)

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

- Wow, minimizing squared error seems like a reasonable thing to do to fit a predictor!
  - When noise is Gaussian, maximizing likelihood of our linear model is equivalent to minimizing the sum of squared errors. Simply a result of derivations
  - However, if we had chosen a different noise distribution, we'd get a different estimator. E.g. Laplacian noise -> minimize absolute error

Maximizing log-likelihood → minimizing mean squared error (MSE)

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

- Wow, minimizing squared error seems like a reasonable thing to do to fit a predictor!
  - When noise is Gaussian, maximizing likelihood of our linear model is equivalent to minimizing the sum of squared errors. Simply a result of derivations
  - However, if we had chosen a different noise distribution, we'd get a different estimator. E.g. Laplacian noise -> minimize absolute error
  - Which one is better depends on assumptions about our data. Gaussian has no long tails, mean squared error incurs high penalty for outliers

Maximizing log-likelihood → minimizing mean squared error (MSE)

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

- Wow, minimizing squared error seems like a reasonable thing to do to fit a predictor!
  - When noise is Gaussian, maximizing likelihood of our linear model is equivalent to minimizing the sum of squared errors. Simply a result of derivations
  - However, if we had chosen a different noise distribution, we'd get a different estimator. E.g. Laplacian noise -> minimize absolute error
  - Which one is better depends on assumptions about our data. Gaussian has no long tails, mean squared error incurs high penalty for outliers
  - **TL;DR: Model of data <— math—> what to optimize**

Maximizing log-likelihood wrt w

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

Maximizing log-likelihood wrt w

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

So how do we actually optimize this / fit our model?

Maximizing log-likelihood wrt w

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

So how do we actually optimize this / fit our model?

Set gradient=0, solve for w

Maximizing log-likelihood wrt w

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

So how do we actually optimize this / fit our model?

Set gradient=0, solve for w

Recall: $w \in \mathbb{R}^d$

Maximizing log-likelihood wrt w

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

So how do we actually optimize this / fit our model?

Set gradient=0, solve for w

Recall: $w \in \mathbb{R}^d$

$$\nabla_w f(x) = \begin{bmatrix} \frac{\partial}{\partial w_0} f(x) \\ \frac{\partial}{\partial w_1} f(x) \\ \vdots \end{bmatrix}$$

Maximizing log-likelihood wrt w

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

So how do we actually optimize this / fit our model?

Set gradient=0, solve for w

Recall: $w \in \mathbb{R}^d$

$$\nabla_w f(x) = \begin{bmatrix} \frac{\partial}{\partial w_0} f(x) \\ \frac{\partial}{\partial w_1} f(x) \\ \vdots \end{bmatrix}$$

Gradient wrt w is a vector of partial derivatives

Maximizing log-likelihood wrt w

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

So how do we actually optimize this / fit our model?

Set gradient=0, solve for w

Recall: $w \in \mathbb{R}^d$

$$\nabla_w f(x) = \begin{bmatrix} \frac{\partial}{\partial w_0} f(x) \\ \frac{\partial}{\partial w_1} f(x) \\ \vdots \end{bmatrix}$$

$$\nabla_w \sum_{i=1}^{n} (y_i - x_i^T w)^2$$

Gradient wrt w is a vector of partial derivatives

Maximizing log-likelihood wrt w

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

So how do we actually optimize this / fit our model?

Set gradient=0, solve for w

Recall: $w \in \mathbb{R}^d$

$$\nabla_w f(x) = \begin{bmatrix} \frac{\partial}{\partial w_0} f(x) \\ \frac{\partial}{\partial w_1} f(x) \\ \vdots \end{bmatrix}$$

$$\nabla_w \sum_{i=1}^{n} (y_i - x_i^T w)^2$$

Gradient wrt w is a vector of partial derivatives

Chain rule

Maximizing log-likelihood wrt w

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

So how do we actually optimize this / fit our model?

Set gradient=0, solve for w

Recall: $w \in \mathbb{R}^d$

$$\nabla_w f(x) = \begin{bmatrix} \frac{\partial}{\partial w_0} f(x) \\ \frac{\partial}{\partial w_1} f(x) \\ \vdots \end{bmatrix}$$

$$\nabla_w \sum_{i=1}^{n} (y_i - x_i^T w)^2$$

Gradient wrt w is a vector of partial derivatives

Chain rule $\nabla_w f(x)^2 = 2f(x) \cdot \nabla_w f(x)$

Maximizing log-likelihood wrt w

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

So how do we actually optimize this / fit our model?

Set gradient=0, solve for w

Recall: $w \in \mathbb{R}^d$

$$\nabla_w f(x) = \begin{bmatrix} \frac{\partial}{\partial w_0} f(x) \\ \frac{\partial}{\partial w_1} f(x) \\ \vdots \end{bmatrix}$$

$$\nabla_w \sum_{i=1}^{n} (y_i - x_i^T w)^2$$

Gradient wrt w is a vector of partial derivatives

$$= \sum_{i=1}^{n} 2(y_i - x_i^T w)^1 \nabla_w (y_i - x_i^T w)$$

Chain rule $\nabla_w f(x)^2 = 2f(x) \cdot \nabla_w f(x)$

Maximizing log-likelihood wrt w

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

So how do we actually optimize this / fit our model?

Set gradient=0, solve for w

Recall: $w \in \mathbb{R}^d$

$$\nabla_w f(x) = \begin{bmatrix} \frac{\partial}{\partial w_0} f(x) \\ \frac{\partial}{\partial w_1} f(x) \\ \vdots \end{bmatrix}$$

$$\nabla_w \sum_{i=1}^{n} (y_i - x_i^T w)^2$$

Gradient wrt w is a vector of partial derivatives

$$= \sum_{i=1}^{n} 2(y_i - x_i^T w)^1 \nabla_w (y_i - x_i^T w)$$

Chain rule $\nabla_w f(x)^2 = 2f(x) \cdot \nabla_w f(x)$

“Matrix Cookbook” identities

Maximizing log-likelihood wrt w

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

So how do we actually optimize this / fit our model?

Set gradient=0, solve for w

Recall: $w \in \mathbb{R}^d$

$$\nabla_w f(x) = \begin{bmatrix} \frac{\partial}{\partial w_0} f(x) \\ \frac{\partial}{\partial w_1} f(x) \\ \vdots \end{bmatrix}$$

$$\nabla_w \sum_{i=1}^{n} (y_i - x_i^T w)^2$$

Gradient wrt w is a vector of partial derivatives

$$= \sum_{i=1}^{n} 2(y_i - x_i^T w)^1 \nabla_w (y_i - x_i^T w)$$

Chain rule $\nabla_w f(x)^2 = 2f(x) \cdot \nabla_w f(x)$

"Matrix Cookbook" identities

$$\nabla_w x^T w =$$

Maximizing log-likelihood wrt w

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

So how do we actually optimize this / fit our model?

Set gradient=0, solve for w

Recall: $w \in \mathbb{R}^d$

$$\nabla_w f(x) = \begin{bmatrix} \frac{\partial}{\partial w_0} f(x) \\ \frac{\partial}{\partial w_1} f(x) \\ \vdots \end{bmatrix}$$

$$\nabla_w \sum_{i=1}^{n} (y_i - x_i^T w)^2$$

Gradient wrt w is a vector of partial derivatives

$$= \sum_{i=1}^{n} 2(y_i - x_i^T w)^1 \nabla_w (y_i - x_i^T w)$$

Chain rule $\nabla_w f(x)^2 = 2f(x) \cdot \nabla_w f(x)$

"Matrix Cookbook" identities

$$\nabla_w x^T w = x$$

Maximizing log-likelihood wrt w

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

So how do we actually optimize this / fit our model?

Set gradient=0, solve for w

Recall: $w \in \mathbb{R}^d$

$$\nabla_w f(x) = \begin{bmatrix} \frac{\partial}{\partial w_0} f(x) \\ \frac{\partial}{\partial w_1} f(x) \\ \vdots \end{bmatrix}$$

Gradient wrt w is a vector of partial derivatives

$$\nabla_w \sum_{i=1}^{n} (y_i - x_i^T w)^2$$

$$= \sum_{i=1}^{n} 2(y_i - x_i^T w)^1 \nabla_w (y_i - x_i^T w)$$

Chain rule $\nabla_w f(x)^2 = 2f(x) \cdot \nabla_w f(x)$

$$= \sum_{i=1}^{n} 2(y_i - x_i^T w)(-x_i)$$

"Matrix Cookbook" identities

$$\nabla_w x^T w = x$$

Maximizing log-likelihood, setting gradient = 0

Set gradient=0, solve for w

$$\nabla_w \sum_{i=1}^{n} (y_i - x_i^T w)^2$$

Maximizing log-likelihood, setting gradient = 0

Set gradient=0, solve for w

$$\nabla_w \sum_{i=1}^{n} (y_i - x_i^T w)^2 = \sum_{i=1}^{n} 2(y_i - x_i^T w)(-x_i)$$

Maximizing log-likelihood, setting gradient = 0

Set gradient=0, solve for w

$$\nabla_w \sum_{i=1}^{n} (y_i - x_i^T w)^2 = \sum_{i=1}^{n} 2(y_i - x_i^T w)(-x_i) = 0$$

Maximizing log-likelihood, setting gradient = 0

Set gradient=0, solve for w

$$\nabla_w \sum_{i=1}^{n} (y_i - x_i^T w)^2 = \sum_{i=1}^{n} 2(y_i - x_i^T w)(-x_i) = 0$$

check dimensions!

Maximizing log-likelihood, setting gradient = 0

Set gradient=0, solve for w

$$\nabla_w \sum_{i=1}^{n} (y_i - x_i^T w)^2 = \sum_{i=1}^{n} 2(y_i - x_i^T w)(-x_i) = 0$$

$d \times 1$

check dimensions!

Maximizing log-likelihood, setting gradient = 0

Set gradient=0, solve for w

$$\nabla_w \sum_{i=1}^{n} (y_i - x_i^T w)^2 = \sum_{i=1}^{n} \underset{1\times 1}{2(y_i - x_i^T w)} \underset{d\times 1}{(-x_i)} = 0$$

check dimensions!

Maximizing log-likelihood, setting gradient = 0

Set gradient=0, solve for w

$$\nabla_w \sum_{i=1}^{n} (y_i - x_i^T w)^2 = \sum_{i=1}^{n} \underset{1\times 1}{2(y_i - x_i^T w)} \underset{1\times 1}{(} \underset{d\times 1}{-x_i)} = 0$$

check dimensions!

Maximizing log-likelihood, setting gradient = 0

Set gradient=0, solve for w

$$\nabla_w \sum_{i=1}^{n} (y_i - x_i^T w)^2 = \sum_{i=1}^{n} \underbrace{2(y_i - x_i^T w)(-x_i)}_{\substack{1\times 1 \quad 1\times 1 \quad d\times 1 \\ d\times 1}} = 0$$

check dimensions!

Maximizing log-likelihood, setting gradient = 0

Set gradient=0, solve for w

$$\nabla_w \sum_{i=1}^{n} (y_i - x_i^T w)^2 = \sum_{i=1}^{n} \underbrace{2(y_i - x_i^T w)(-x_i)}_{\substack{1\times 1 \quad 1\times 1 \quad d\times 1 \\ d\times 1}} = 0 \qquad \begin{matrix} \\ d\times 1 \end{matrix}$$

check dimensions!

Maximizing log-likelihood, setting gradient = 0

Set gradient=0, solve for w

$$\nabla_w \sum_{i=1}^{n} (y_i - x_i^T w)^2 = \sum_{i=1}^{n} \underbrace{2(y_i - x_i^T w)(-x_i)}_{\substack{1\times 1 \quad 1\times 1 \quad d\times 1 \\ d\times 1}} = \underset{d\times 1}{0}$$

check dimensions!

$\vec{0} \in \mathbb{R}^d$

Maximizing log-likelihood, setting gradient = 0

Set gradient=0, solve for w

$$\nabla_w \sum_{i=1}^{n} (y_i - x_i^T w)^2 = \sum_{i=1}^{n} \underbrace{2(y_i - x_i^T w)(-x_i)}_{d \times 1} = 0$$

$1 \times 1 \quad 1 \times 1 \quad d \times 1 \qquad d \times 1$

check dimensions!

$\vec{0} \in \mathbb{R}^d$

$$= 2 \sum_{i=1}^{n} [-y_i x_i + x_i^T w x_i] = 0$$

Maximizing log-likelihood, setting gradient = 0

Set gradient=0, solve for w

$$\nabla_w \sum_{i=1}^{n} (y_i - x_i^T w)^2 = \sum_{i=1}^{n} \underbrace{2(y_i - x_i^T w)(-x_i)}_{d \times 1} = 0$$

$1 \times 1 \quad 1 \times 1 \quad d \times 1 \qquad d \times 1$

check dimensions!

$\vec{0} \in \mathbb{R}^d$

$$= 2 \sum_{i=1}^{n} [-y_i x_i + x_i^T w x_i] = 0$$

Now for some tricks

Maximizing log-likelihood, setting gradient = 0

Set gradient=0, solve for w

$$\nabla_w \sum_{i=1}^{n} (y_i - x_i^T w)^2 = \sum_{i=1}^{n} \underbrace{2(y_i - x_i^T w)(-x_i)}_{d \times 1} = 0$$

$1 \times 1 \quad 1 \times 1 \quad d \times 1 \qquad d \times 1$

check dimensions!

$\vec{0} \in \mathbb{R}^d$

$$= 2 \sum_{i=1}^{n} [-y_i x_i + x_i^T w x_i] = 0$$

Now for some tricks

$x^T w = a$ # scalar

Maximizing log-likelihood, setting gradient = 0

Set gradient=0, solve for w

$$\nabla_w \sum_{i=1}^{n} (y_i - x_i^T w)^2 = \sum_{i=1}^{n} \underbrace{2(y_i - x_i^T w)(-x_i)}_{d \times 1} = 0$$

$1 \times 1 \quad 1 \times 1 \quad d \times 1 \qquad d \times 1$

check dimensions!

$\vec{0} \in \mathbb{R}^d$

$$= 2 \sum_{i=1}^{n} [-y_i x_i + x_i^T w x_i] = 0$$

Now for some tricks

$x^T w = a$ # scalar

$a\vec{v} = \vec{v}a$

Maximizing log-likelihood, setting gradient = 0

Set gradient=0, solve for w

$$\nabla_w \sum_{i=1}^{n} (y_i - x_i^T w)^2 = \sum_{i=1}^{n} \underbrace{2}_{1\times 1}\underbrace{(y_i - x_i^T w)}_{1\times 1}\underbrace{(-x_i)}_{d\times 1} = \underbrace{0}_{d\times 1}$$

$d \times 1$

check dimensions!

$\vec{0} \in \mathbb{R}^d$

$$= 2\sum_{i=1}^{n} [-y_i x_i + x_i^T w x_i] = 0$$

Now for some tricks

$$= 2\sum_{i=1}^{n} [-x_i y_i + x_i x_i^T w] = 0$$

$x^T w = a$ # scalar

$a\vec{v} = \vec{v}a$

Maximizing log-likelihood, setting gradient = 0

Set gradient=0, solve for w

$$\nabla_w \sum_{i=1}^{n} (y_i - x_i^T w)^2 = \sum_{i=1}^{n} \underbrace{2(y_i - x_i^T w)(-x_i)}_{d \times 1} = 0$$

$1 \times 1 \quad 1 \times 1 \quad d \times 1 \qquad d \times 1$

check dimensions!

$\vec{0} \in \mathbb{R}^d$

$$= 2 \sum_{i=1}^{n} [-y_i x_i + x_i^T w x_i] = 0$$

Now for some tricks

$x^T w = a$ # scalar

$a\vec{v} = \vec{v}a$

$$= 2 \sum_{i=1}^{n} [-x_i y_i + x_i x_i^T w] = 0$$

$$\sum_{i=1}^{n} x_i y_i = \left(\sum_{i=1}^{n} x_i x_i^T\right) w$$

Maximizing log-likelihood, setting gradient = 0

Set gradient=0, solve for w

$$\nabla_w \sum_{i=1}^{n} (y_i - x_i^T w)^2 = \sum_{i=1}^{n} \underbrace{2(y_i - x_i^T w)(-x_i)}_{d \times 1} = 0$$

$1 \times 1 \quad 1 \times 1 \quad d \times 1 \qquad d \times 1$

check dimensions!

$\vec{0} \in \mathbb{R}^d$

$$= 2 \sum_{i=1}^{n} [-y_i x_i + x_i^T w x_i] = 0$$

Now for some tricks

$$= 2 \sum_{i=1}^{n} [-x_i y_i + x_i x_i^T w] = 0$$

$x^T w = a$ # scalar

$a\vec{v} = \vec{v}a$

$$\sum_{i=1}^{n} x_i y_i = (\sum_{i=1}^{n} x_i x_i^T) w$$

How can we isolate w?

Maximizing log-likelihood, setting gradient = 0

Set gradient=0, solve for w

$$\nabla_w \sum_{i=1}^{n} (y_i - x_i^T w)^2 = \sum_{i=1}^{n} \underbrace{2(y_i - x_i^T w)(-x_i)}_{d \times 1} = 0$$

$1 \times 1 \quad 1 \times 1 \quad d \times 1 \qquad d \times 1$

check dimensions!

$\vec{0} \in \mathbb{R}^d$

$$= 2\sum_{i=1}^{n} [-y_i x_i + x_i^T w x_i] = 0$$

Now for some tricks

$$= 2\sum_{i=1}^{n} [-x_i y_i + x_i x_i^T w] = 0$$

$x^T w = a$ # scalar

$a\vec{v} = \vec{v}a$

$$\sum_{i=1}^{n} x_i y_i = \left(\sum_{i=1}^{n} x_i x_i^T\right) w$$

How can we isolate w?

Maximizing log-likelihood, setting gradient = 0

Set gradient=0, solve for w

$$\nabla_w \sum_{i=1}^{n} (y_i - x_i^T w)^2 = \sum_{i=1}^{n} \underbrace{2(y_i - x_i^T w)(-x_i)}_{d \times 1} = 0$$

$1 \times 1 \quad 1 \times 1 \quad d \times 1 \qquad d \times 1$

check dimensions!

$\vec{0} \in \mathbb{R}^d$

$$= 2 \sum_{i=1}^{n} [-y_i x_i + x_i^T w x_i] = 0$$

Now for some tricks

$$= 2 \sum_{i=1}^{n} [-x_i y_i + x_i x_i^T w] = 0$$

$x^T w = a$ # scalar

$a\vec{v} = \vec{v}a$

$$\sum_{i=1}^{n} x_i y_i = \underbrace{(\sum_{i=1}^{n} x_i x_i^T)}_{d \times d} w$$

How can we isolate w?

Maximizing log-likelihood, setting gradient = 0

Set gradient=0, solve for w

$$\nabla_w \sum_{i=1}^{n} (y_i - x_i^T w)^2 = \sum_{i=1}^{n} \underbrace{2(y_i - x_i^T w)(-x_i)}_{d \times 1} = 0$$

$1 \times 1 \quad 1 \times 1 \quad d \times 1 \qquad d \times 1$

check dimensions!

$\vec{0} \in \mathbb{R}^d$

$$= 2\sum_{i=1}^{n} [-y_i x_i + x_i^T w x_i] = 0$$

Now for some tricks

$$= 2\sum_{i=1}^{n} [-x_i y_i + x_i x_i^T w] = 0$$

$x^T w = a$ # scalar

$a\vec{v} = \vec{v}a$

Reminder

$x^T x$ = inner product or dot product 1x1

xx^T = outer product dxd

$$\sum_{i=1}^{n} x_i y_i = \underbrace{(\sum_{i=1}^{n} x_i x_i^T)}_{d \times d} w$$

How can we isolate w?

Maximizing log-likelihood, setting gradient = 0

Set gradient=0, solve for w

$$\nabla_w \sum_{i=1}^{n} (y_i - x_i^T w)^2 = \sum_{i=1}^{n} \underbrace{2(y_i - x_i^T w)(-x_i)}_{d \times 1} = 0$$

$1 \times 1 \quad 1 \times 1 \quad d \times 1 \qquad d \times 1$

check dimensions!

$\vec{0} \in \mathbb{R}^d$

$$= 2\sum_{i=1}^{n} [-y_i x_i + x_i^T w x_i] = 0$$

Now for some tricks

$$= 2\sum_{i=1}^{n} [-x_i y_i + x_i x_i^T w] = 0$$

$x^T w = a$ # scalar

$a\vec{v} = \vec{v}a$

Reminder

$x^T x$ = inner product or dot product 1x1

xx^T = outer product dxd

$$\sum_{i=1}^{n} x_i y_i = \underbrace{(\sum_{i=1}^{n} x_i x_i^T)}_{d \times d} w$$

A matrix

How can we isolate w?

Maximizing log-likelihood, setting gradient = 0

Set gradient=0, solve for w

$$\nabla_w \sum_{i=1}^{n} (y_i - x_i^T w)^2 = \sum_{i=1}^{n} \underbrace{2(y_i - x_i^T w)(-x_i)}_{d \times 1} = 0$$

$1 \times 1 \quad 1 \times 1 \quad d \times 1 \qquad d \times 1$

check dimensions!

$\vec{0} \in \mathbb{R}^d$

$$= 2\sum_{i=1}^{n} [-y_i x_i + x_i^T w x_i] = 0$$

Now for some tricks

$$= 2\sum_{i=1}^{n} [-x_i y_i + x_i x_i^T w] = 0$$

$x^T w = a$ # scalar

$a\vec{v} = \vec{v}a$

Reminder

$x^T x$ = inner product or dot product 1x1

xx^T = outer product dxd

$$\sum_{i=1}^{n} x_i y_i = \underbrace{(\sum_{i=1}^{n} x_i x_i^T)}_{d \times d} w$$

A matrix

How can we isolate w?

Take the inverse!

Maximizing log-likelihood, setting gradient = 0

Set gradient=0, solve for w

$$\nabla_w \sum_{i=1}^{n} (y_i - x_i^T w)^2 = \sum_{i=1}^{n} \underbrace{2(y_i - x_i^T w)(-x_i)}_{d \times 1} = 0$$

$1 \times 1 \quad 1 \times 1 \quad d \times 1 \qquad d \times 1$

check dimensions!

$\vec{0} \in \mathbb{R}^d$

$$= 2\sum_{i=1}^{n} [-y_i x_i + x_i^T w x_i] = 0$$

Now for some tricks

$$= 2\sum_{i=1}^{n} [-x_i y_i + x_i x_i^T w] = 0$$

$x^T w = a$ # scalar

$a\vec{v} = \vec{v}a$

Reminder

$x^T x$ = inner product or dot product 1x1

xx^T = outer product dxd

$$\sum_{i=1}^{n} x_i y_i = \boxed{\left(\sum_{i=1}^{n} x_i x_i^T\right)^{-1}} w$$

$d \times d$

A matrix

How can we isolate w?

Take the inverse!

Maximizing log-likelihood, solving for w

Set gradient=0, solve for w

$$\sum_{i=1}^{n} x_i y_i = \left(\sum_{i=1}^{n} x_i x_i^T\right)^{-1} w$$

$d \times d$

A matrix

How can we isolate w?

Take the inverse!

Maximizing log-likelihood, solving for w

Set gradient=0, solve for w

$$\sum_{i=1}^{n} x_i y_i = \left(\sum_{i=1}^{n} x_i x_i^T\right)^{-1} w$$

$d \times d$

A matrix

How can we isolate w?

Take the inverse!

$AA^{-1} = I$

Maximizing log-likelihood, solving for w

Set gradient=0, solve for w

$$\sum_{i=1}^{n} x_i y_i = \left(\sum_{i=1}^{n} x_i x_i^T\right)^{-1} w$$

$d \times d$

A matrix

$$\left(\sum_{i=1}^{n} x_i x_i^T\right)^{-1} \sum_{i=1}^{n} x_i y_i = \hat{w}_{\text{MLE}}$$

How can we isolate w?

Take the inverse!

$AA^{-1} = I$

Maximizing log-likelihood, solving for w

Set gradient=0, solve for w

$$\sum_{i=1}^{n} x_i y_i = \left(\sum_{i=1}^{n} x_i x_i^T\right)^{-1} w$$

$d \times d$

A matrix

$$\left(\sum_{i=1}^{n} x_i x_i^T\right)^{-1} \sum_{i=1}^{n} x_i y_i = \hat{w}_{\text{MLE}}$$

How can we isolate w?

Take the inverse!

$AA^{-1} = I$

Maximizing log-likelihood, solving for w

Set gradient=0, solve for w

$$\sum_{i=1}^{n} x_i y_i = \left(\sum_{i=1}^{n} x_i x_i^T\right)^{-1} w$$

$d \times d$

A matrix

$$\left(\sum_{i=1}^{n} x_i x_i^T\right)^{-1} \sum_{i=1}^{n} x_i y_i = \hat{w}_{\text{MLE}}$$

How can we isolate w?

Take the inverse!

$AA^{-1} = I$

The equation for fitting our model parameters w to our data

Maximizing log-likelihood, solving for w

Set gradient=0, solve for w

$$\sum_{i=1}^{n} x_i y_i = \left(\sum_{i=1}^{n} x_i x_i^T\right)^{-1} w$$

$d \times d$

A matrix

How can we isolate w?

Take the inverse!

$AA^{-1} = I$

$$\left(\sum_{i=1}^{n} x_i x_i^T\right)^{-1} \sum_{i=1}^{n} x_i y_i = \hat{w}_{\text{MLE}}$$

The equation for fitting our model parameters w to our data

- What constraints does taking the inverse place on our data?
  - Matrix must be invertible
  - n >= d

Maximizing log-likelihood, solving for w

Set gradient=0, solve for w

$$\sum_{i=1}^{n} x_i y_i = \left(\sum_{i=1}^{n} x_i x_i^T\right)^{-1} w$$

$d \times d$

A matrix

How can we isolate w?

Take the inverse!

$AA^{-1} = I$

$$\left(\sum_{i=1}^{n} x_i x_i^T\right)^{-1} \sum_{i=1}^{n} x_i y_i = \hat{w}_{\text{MLE}}$$

The equation for fitting our model parameters w to our data

- What constraints does taking the inverse place on our data?
  - Matrix must be invertible
  - n >= d
- For this model we can analytically derive how to find the optimal weights. Will not always be true for more complex models

Maximizing log-likelihood, final form

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

Set gradient=0, solve for w

$$\widehat{w}_{MLE} = \left(\sum_{i=1}^{n} x_i x_i^\top \right)^{-1} \sum_{i=1}^{n} x_i y_i$$

The regression problem in matrix notation

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}$$

We ❤️ matrix notation!

d : # of features

n : # of examples/datapoints

The regression problem in matrix notation

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}$$

We ❤️ matrix notation!

d : # of features

n : # of examples/datapoints

$x \in \mathbb{R}^d$

The regression problem in matrix notation

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

We ❤️ matrix notation!

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix} \begin{matrix} d \times 1 \\ d \times 1 \\ \\ d \times 1 \end{matrix}$$

d : # of features

n : # of examples/datapoints

$x \in \mathbb{R}^d$

The regression problem in matrix notation, translating

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

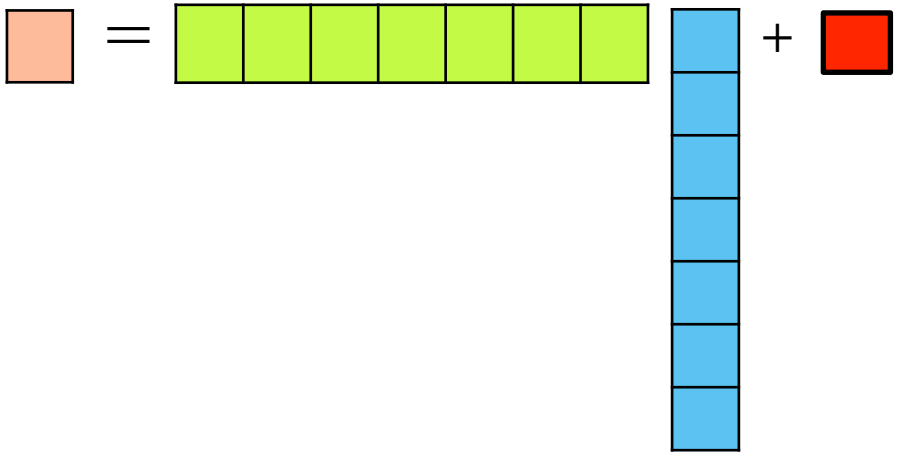
$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix} \begin{matrix} d \times 1 \\ d \times 1 \\ \\ d \times 1 \end{matrix}$$

d : # of features

n : # of examples/datapoints

$x \in \mathbb{R}^d$

Previously… $y_i = x_i^T w + \epsilon_i$



The regression problem in matrix notation, translating

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

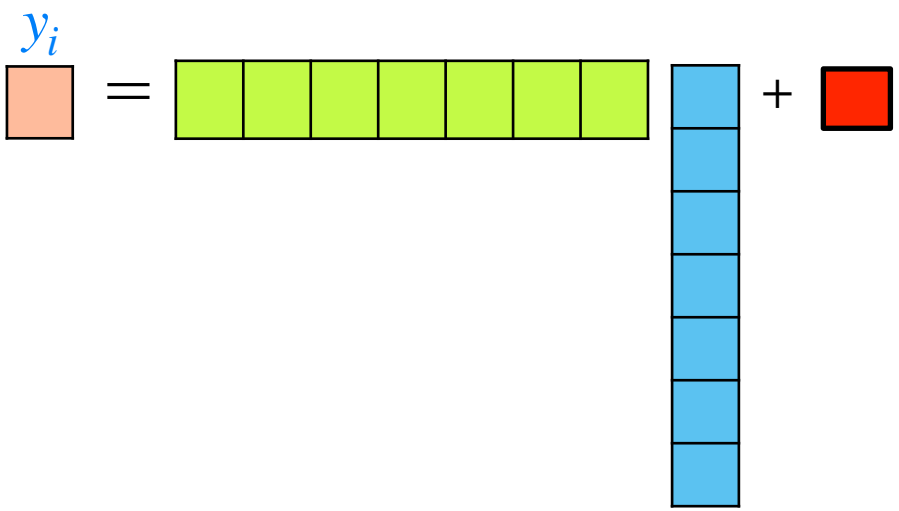
$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix} \begin{matrix} d \times 1 \\ d \times 1 \\ \\ d \times 1 \end{matrix}$$

d : # of features

n : # of examples/datapoints

$x \in \mathbb{R}^d$

Previously… $y_i = x_i^T w + \epsilon_i$



The regression problem in matrix notation, translating

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

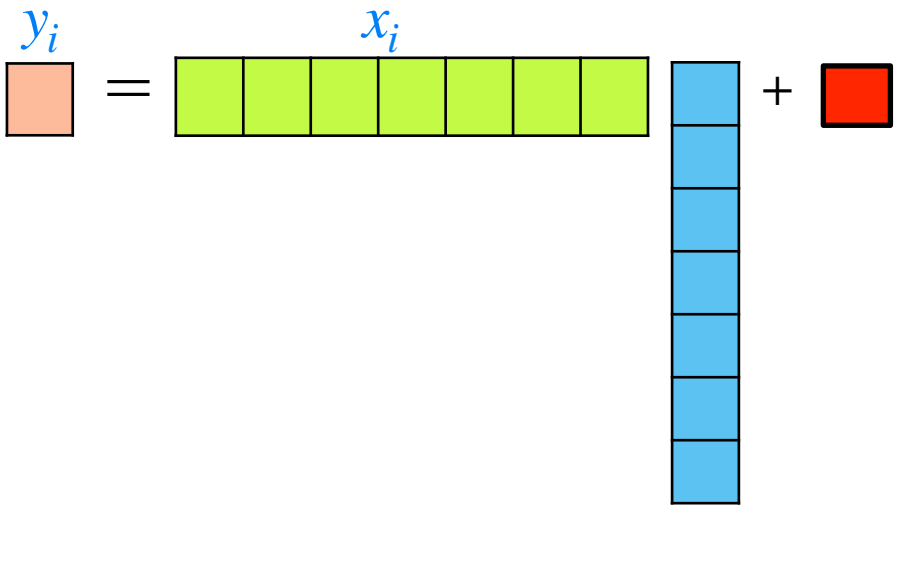
$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix} \begin{matrix} d \times 1 \\ d \times 1 \\ \\ d \times 1 \end{matrix}$$

d : # of features

n : # of examples/datapoints

$x \in \mathbb{R}^d$

Previously… $y_i = x_i^T w + \epsilon_i$



The regression problem in matrix notation, translating

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

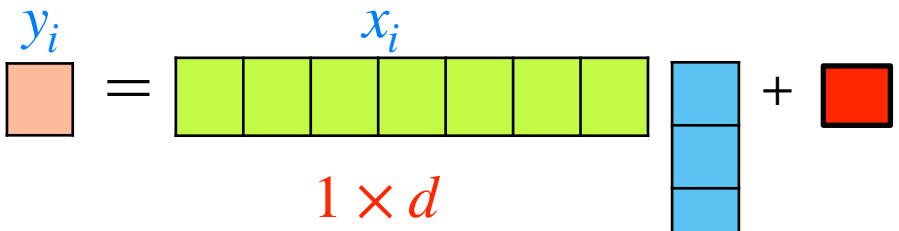
$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix} \begin{matrix} d \times 1 \\ d \times 1 \\ \\ d \times 1 \end{matrix}$$

d : # of features

n : # of examples/datapoints

$x \in \mathbb{R}^d$

Previously… $y_i = x_i^T w + \epsilon_i$



The regression problem in matrix notation, translating

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

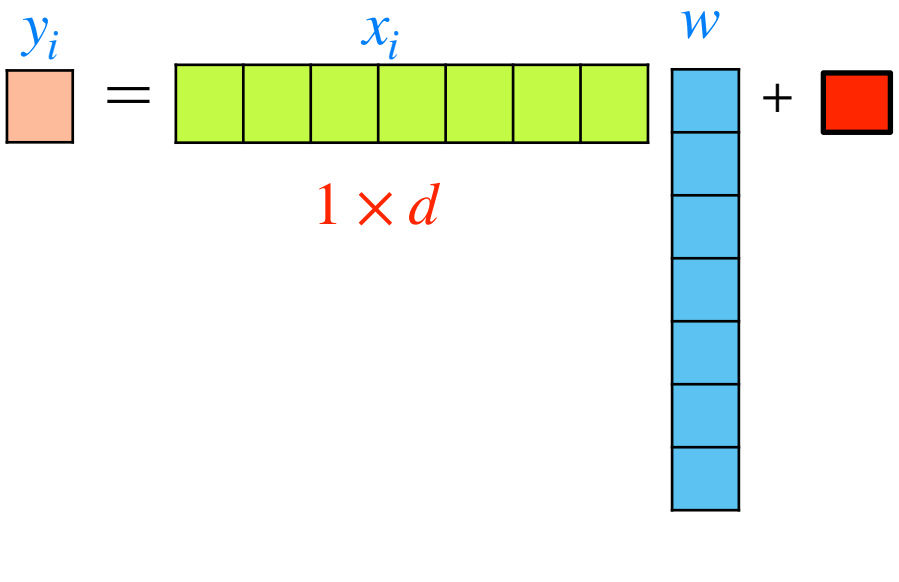
$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix} \begin{matrix} d \times 1 \\ d \times 1 \\ \\ d \times 1 \end{matrix}$$

d : # of features

n : # of examples/datapoints

$x \in \mathbb{R}^d$

Previously… $y_i = x_i^T w + \epsilon_i$



The regression problem in matrix notation, translating

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

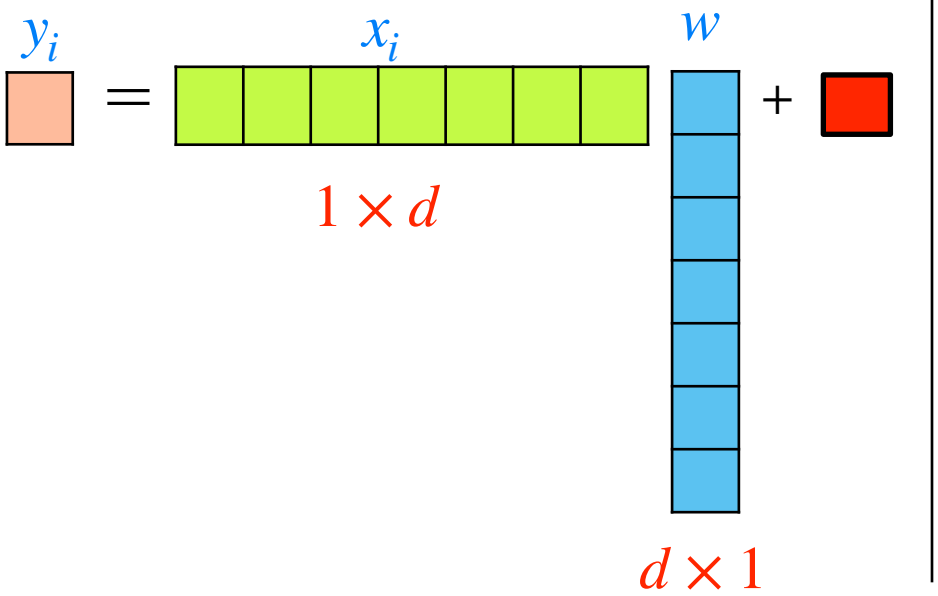
$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix} \begin{matrix} d \times 1 \\ d \times 1 \\ \\ d \times 1 \end{matrix}$$

d : # of features

n : # of examples/datapoints

$x \in \mathbb{R}^d$

Previously… $y_i = x_i^T w + \epsilon_i$



The regression problem in matrix notation, translating

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

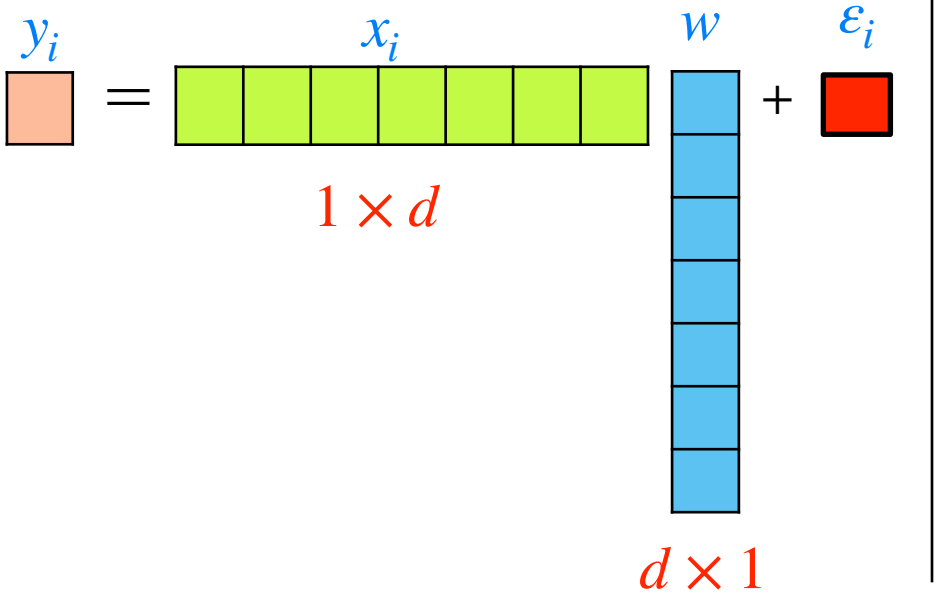
$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix} \begin{matrix} d \times 1 \\ d \times 1 \\ \\ d \times 1 \end{matrix}$$

d : # of features

n : # of examples/datapoints

$x \in \mathbb{R}^d$

Previously… $y_i = x_i^T w + \epsilon_i$



The regression problem in matrix notation, translated

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

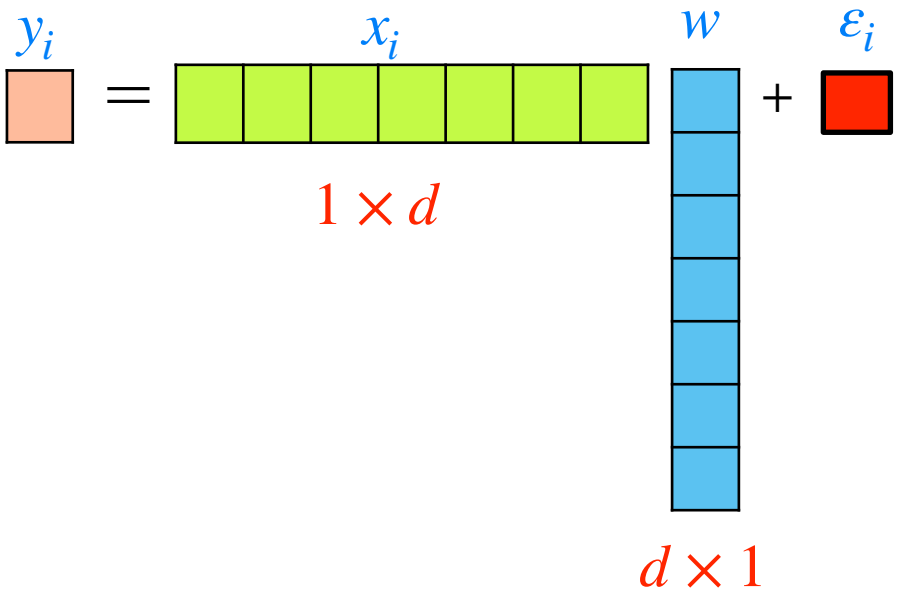
$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}$$

$\mathbf{X} \in \mathbb{R}^{n \times d}$

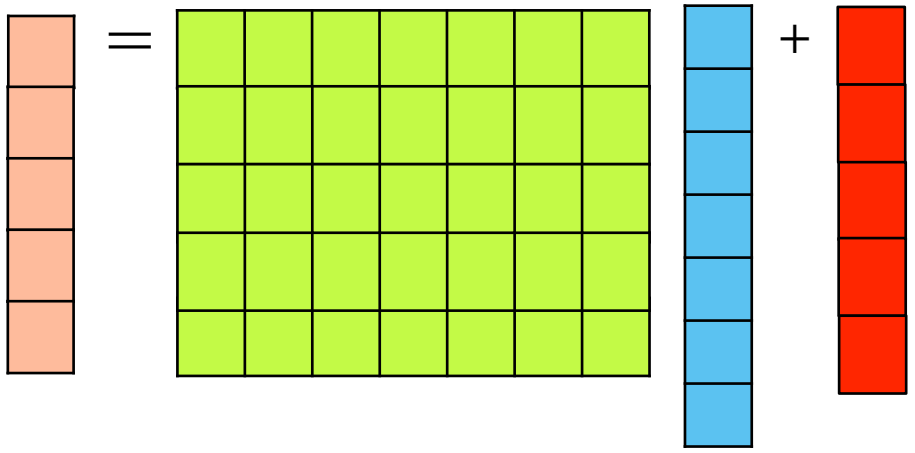
d : # of features

n : # of examples/datapoints

Previously… $y_i = x_i^T w + \epsilon_i$



Now: $\mathbf{y} = \mathbf{X}w + \epsilon$



The regression problem in matrix notation, translated

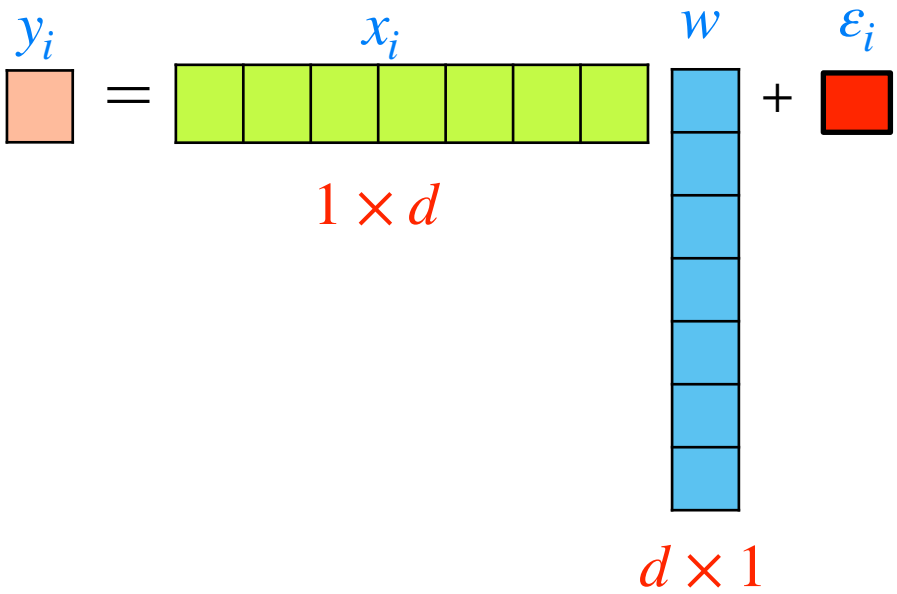
$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}$$

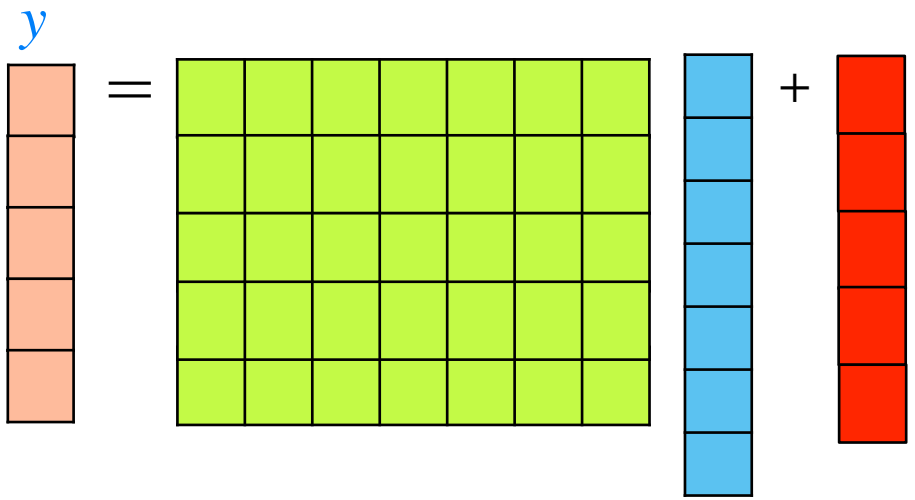
d : # of features

n : # of examples/datapoints

Previously… $y_i = x_i^T w + \epsilon_i$



Now: $\mathbf{y} = \mathbf{X}w + \epsilon$



The regression problem in matrix notation, translated

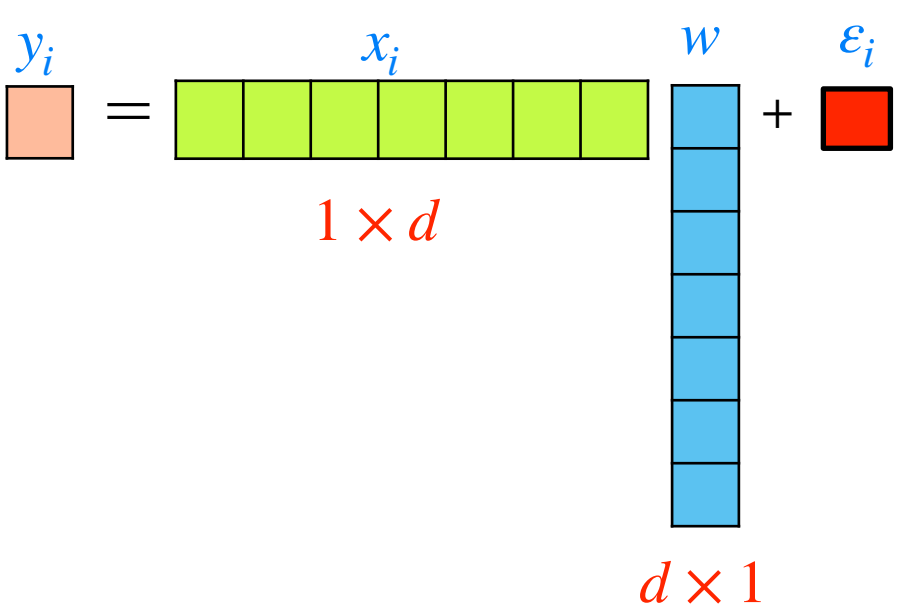
$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}$$

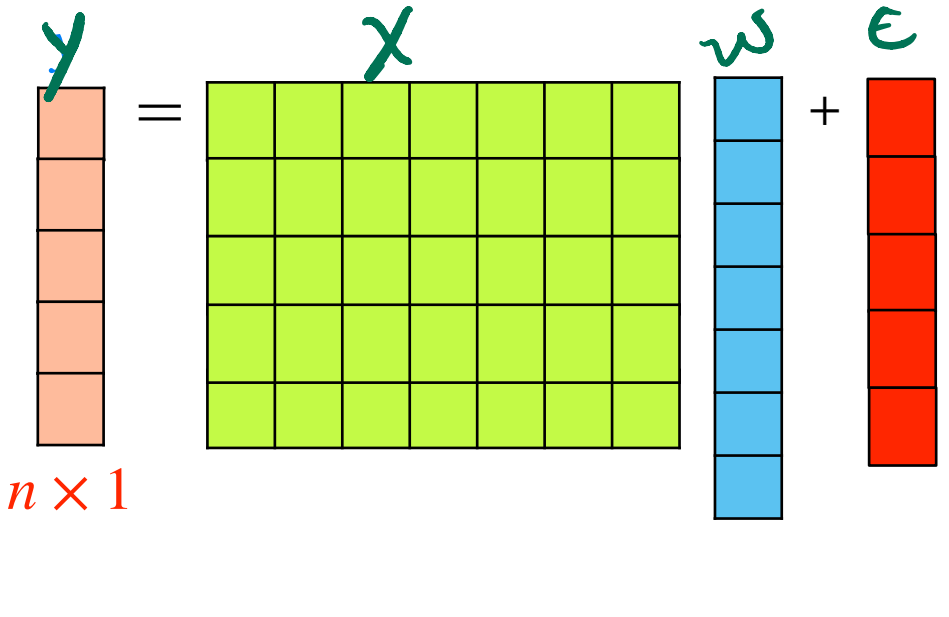
d : # of features

n : # of examples/datapoints

Previously… $y_i = x_i^T w + \epsilon_i$



Now: $\mathbf{y} = \mathbf{X}w + \epsilon$



The regression problem in matrix notation, translated

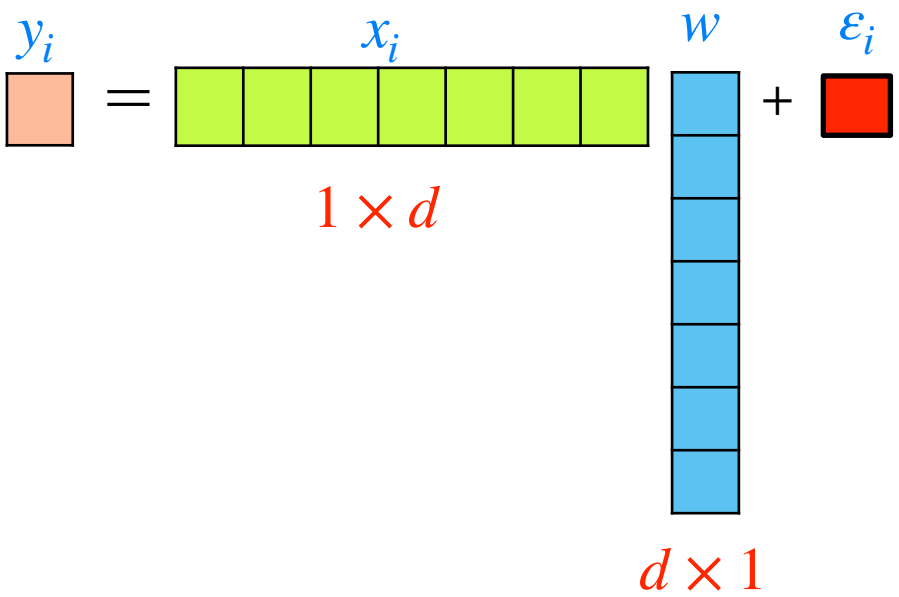
$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}$$

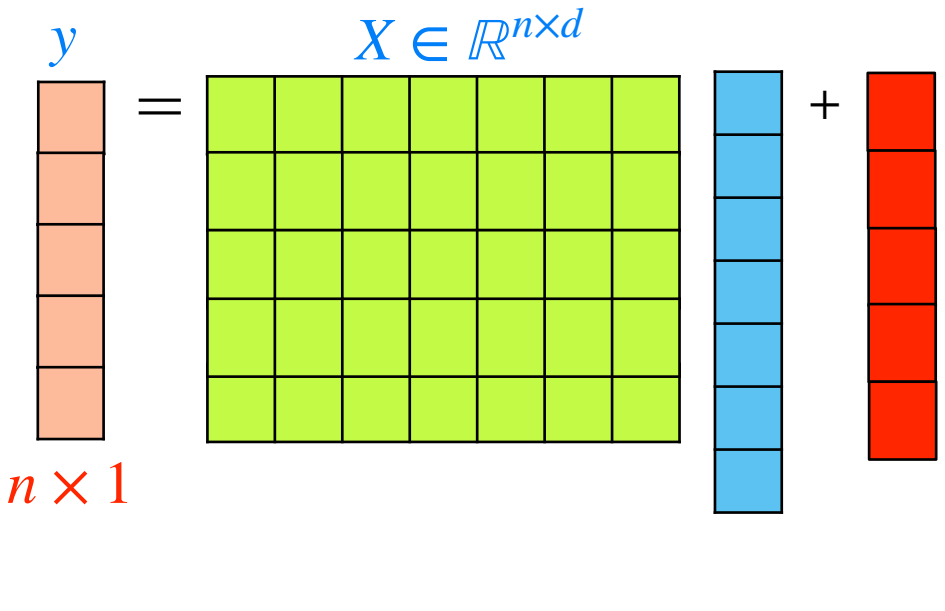
d : # of features

n : # of examples/datapoints

Previously… $y_i = x_i^T w + \epsilon_i$



Now: $\mathbf{y} = \mathbf{X}w + \epsilon$



The regression problem in matrix notation, translated

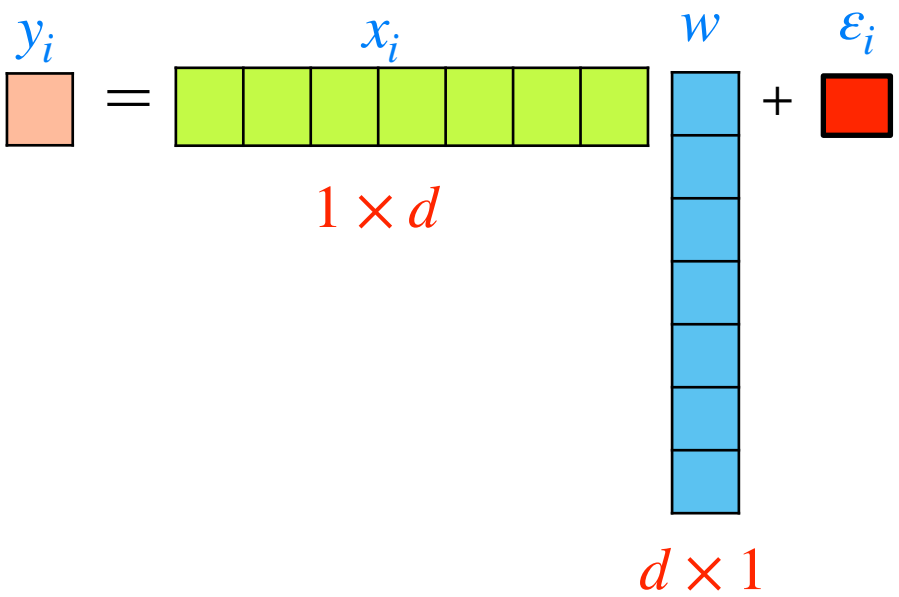
$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}$$

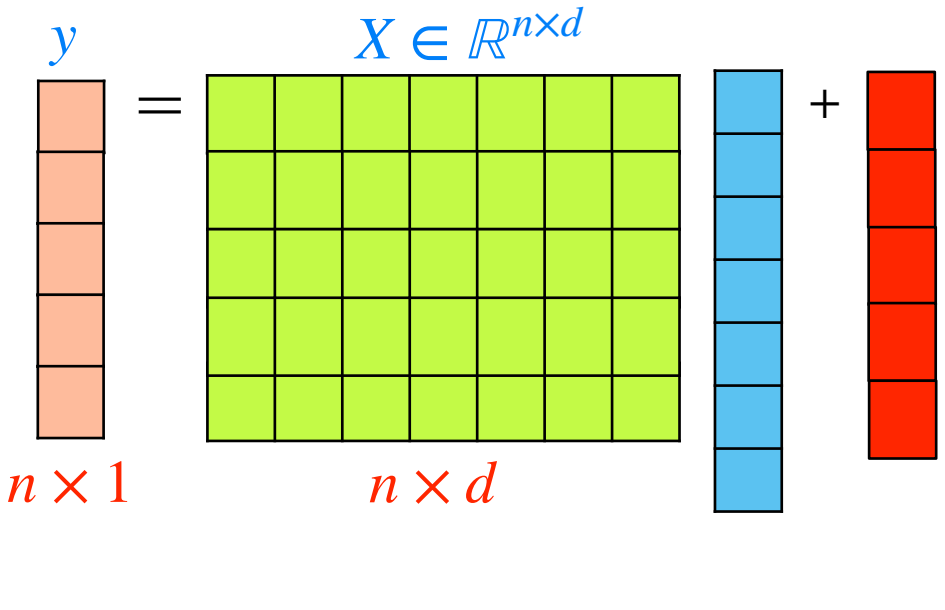
d : # of features

n : # of examples/datapoints

Previously… $y_i = x_i^T w + \epsilon_i$



Now: $\mathbf{y} = \mathbf{X}w + \epsilon$



The regression problem in matrix notation, translated

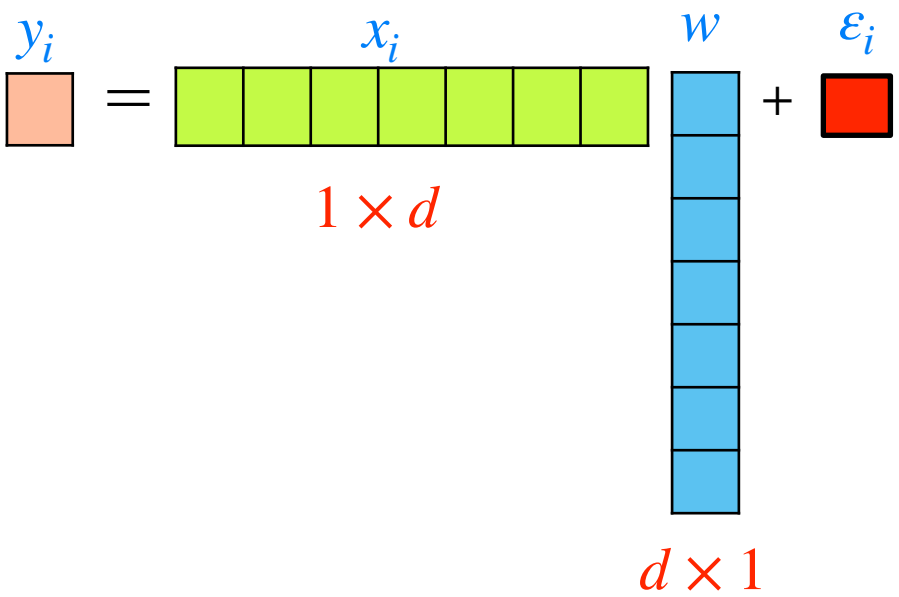
$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}$$

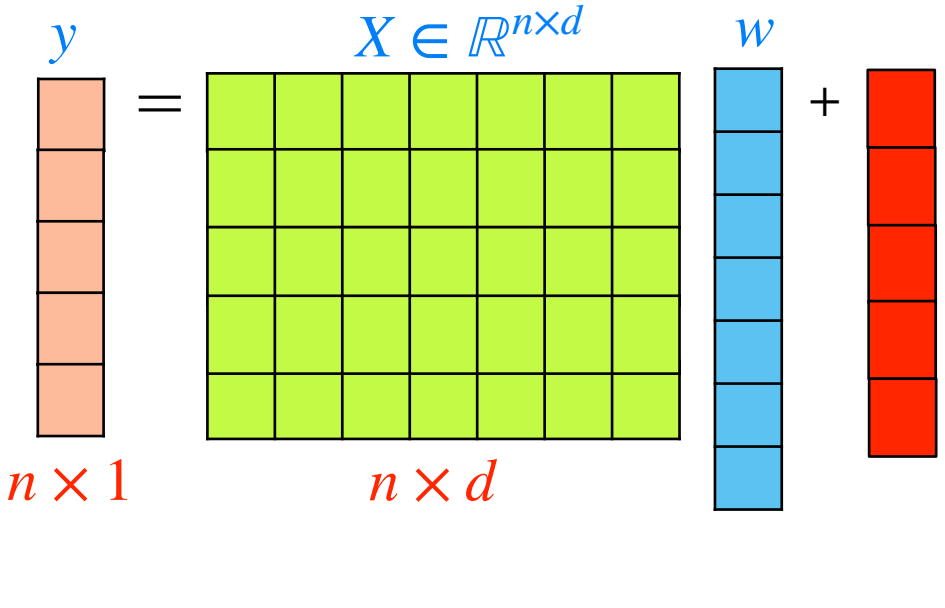
d : # of features

n : # of examples/datapoints

Previously… $y_i = x_i^T w + \epsilon_i$



Now: $\mathbf{y} = \mathbf{X}w + \epsilon$



The regression problem in matrix notation, translated

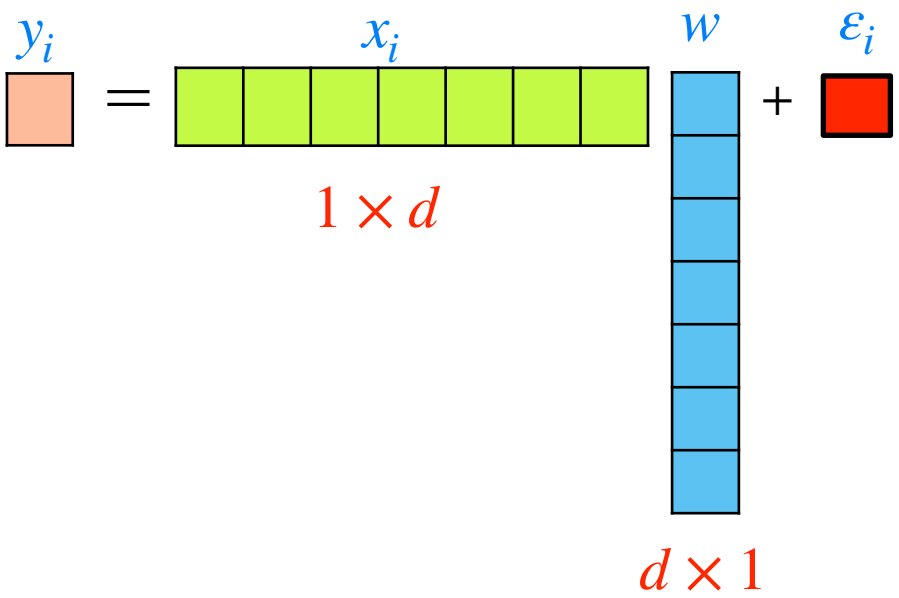
$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}$$

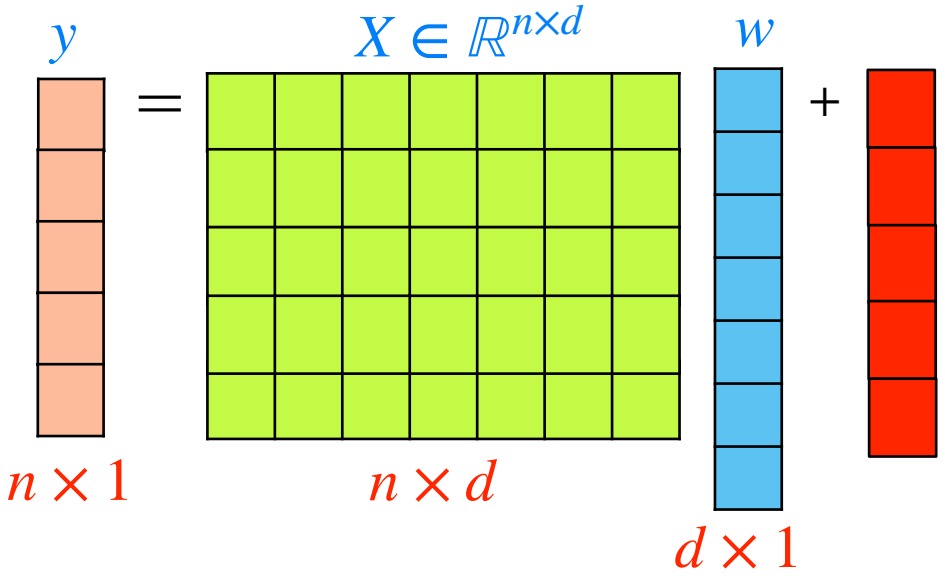
d : # of features

n : # of examples/datapoints

Previously… $y_i = x_i^T w + \epsilon_i$



Now: $\mathbf{y} = \mathbf{X}w + \epsilon$



The regression problem in matrix notation, translated

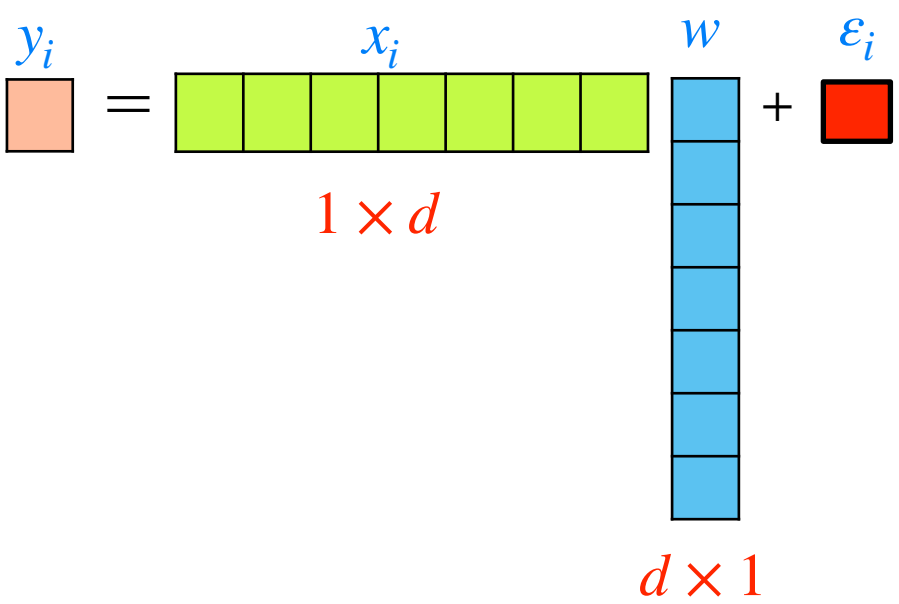
$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}$$

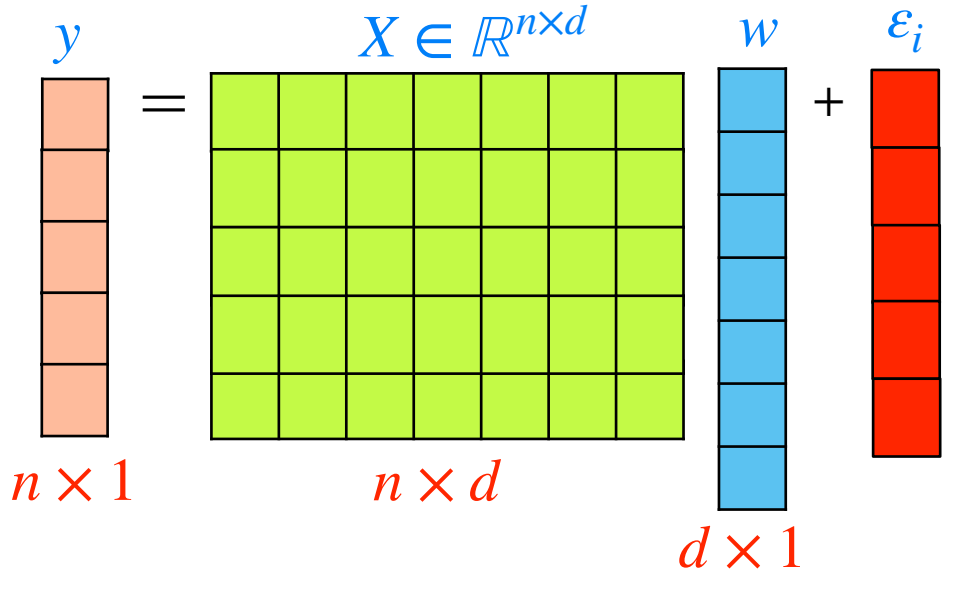
d : # of features

n : # of examples/datapoints

Previously… $y_i = x_i^T w + \epsilon_i$



Now: $\mathbf{y} = \mathbf{X}w + \epsilon$



The regression problem in matrix notation, translated

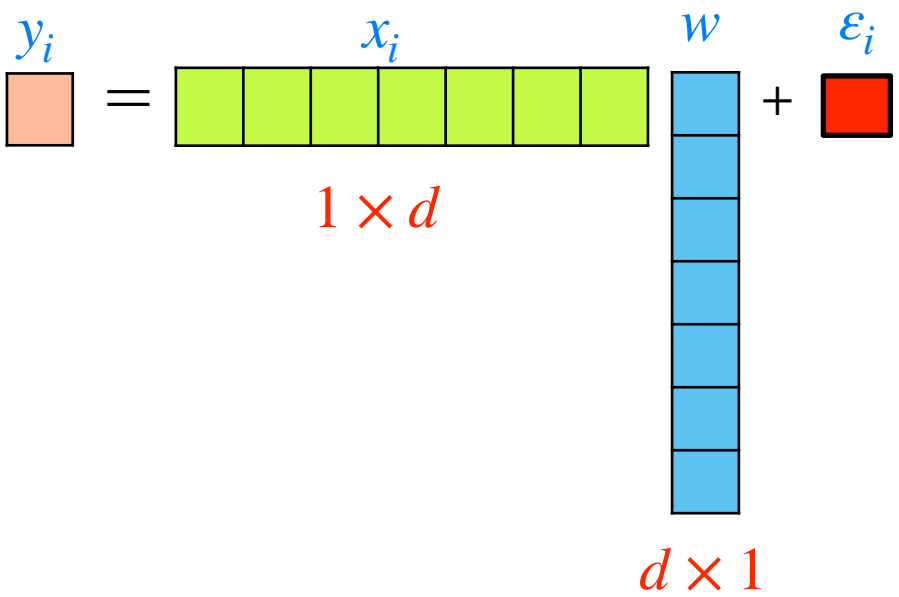
$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}$$

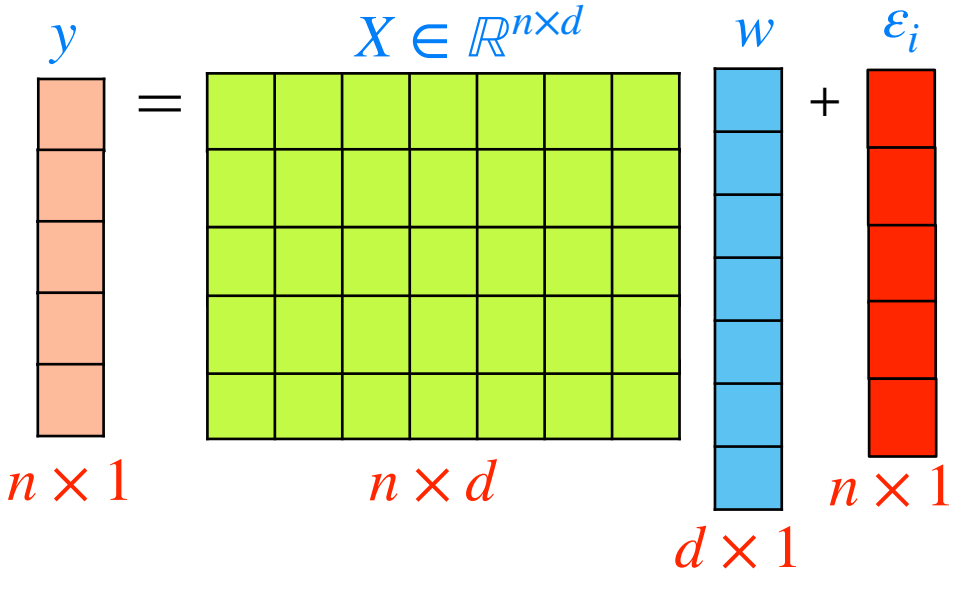
d : # of features

n : # of examples/datapoints

Previously… $y_i = x_i^T w + \epsilon_i$



Now: $\mathbf{y} = \mathbf{X}w + \epsilon$



Least squares in matrix notation

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}$$

d : # of features

n : # of examples/datapoints

Now: $\mathbf{y} = \mathbf{X}w + \epsilon$

Least squares in matrix notation

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

How to make matrix-y?

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}$$

d : # of features

n : # of examples/datapoints

Now: $\mathbf{y} = \mathbf{X}w + \epsilon$

Least squares in matrix notation

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

How to make matrix-y?

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}$$

d : # of features

n : # of examples/datapoints

Now: $\mathbf{y} = \mathbf{X}w + \epsilon$

More identities: ℓ_2 norm: $\|z\|_2 = \sqrt{\sum_{i=1}^{n} z_i^2} = \sqrt{z^\top z}$

Least squares in matrix notation

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

How to make matrix-y?

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}$$

d : # of features

n : # of examples/datapoints

Now: $\mathbf{y} = \mathbf{X}w + \epsilon$

More identities: ℓ_2 norm: $\|z\|_2 = \sqrt{\sum_{i=1}^{n} z_i^2} = \sqrt{z^\top z}$

Let $z_i = (y_i - x_i^T w)$

Least squares in matrix notation

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

How to make matrix-y?

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}$$

d : # of features

n : # of examples/datapoints

Now: $\mathbf{y} = \mathbf{X}w + \epsilon$

More identities: ℓ_2 norm: $\|z\|_2 = \sqrt{\sum_{i=1}^{n} z_i^2} = \sqrt{z^\top z}$

Let $z_i = (y_i - x_i^T w)$ Then

Least squares in matrix notation

$$\widehat{w}_{MLE} = \arg\min_w \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

How to make matrix-y?

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}$$

d : # of features

n : # of examples/datapoints

Now: $\mathbf{y} = \mathbf{X}w + \epsilon$

More identities: ℓ_2 norm: $\|z\|_2 = \sqrt{\sum_{i=1}^n z_i^2} = \sqrt{z^\top z}$

Let $z_i = (y_i - x_i^T w)$ Then

$$\widehat{w}_{LS} = \arg\min_w ||\mathbf{y} - \mathbf{X}w||_2^2$$

$$\hat{w} = \arg\min_w \sum_i (y_i - x_i^T w)^2$$
$$= \arg\min_w \sum_i z_i^2$$
$$= \arg\min_w ||z||_2^2$$

Least squares in matrix notation

$$\widehat{w}_{MLE} = \arg\min_w \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

How to make matrix-y?

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}$$

d : # of features

n : # of examples/datapoints

Now: $\mathbf{y} = \mathbf{X}w + \epsilon$

More identities: ℓ_2 norm: $\|z\|_2 = \sqrt{\sum_{i=1}^n z_i^2} = \sqrt{z^\top z}$

Let $z_i = (y_i - x_i^T w)$ Then

$$\widehat{w}_{LS} = \arg\min_w ||\mathbf{y} - \mathbf{X}w||_2^2$$

$$= \arg\min_w (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

Solving least squares in matrix notation

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

Solving least squares in matrix notation

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

As before, first simplify the function to make it easier to work with

Solving least squares in matrix notation

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

As before, first simplify the function to make it easier to work with

$$= \arg\min_{w} \; y^T y - y^T Xw - (Xw)^T y + (Xw)^T (Xw)$$

Solving least squares in matrix notation

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

As before, first simplify the function to make it easier to work with

$$= \arg\min_{w} \; y^T y - y^T X w - (Xw)^T y + (Xw)^T (Xw)$$

More identities:

Solving least squares in matrix notation

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

As before, first simplify the function to make it easier to work with

$$= \arg\min_{w} \; y^T y - y^T X w - (Xw)^T y + (Xw)^T (Xw)$$

More identities:

$$(ABC)^T = C^T B^T A^T$$

Solving least squares in matrix notation

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

As before, first simplify the function to make it easier to work with

$$= \arg\min_{w} \; y^T y - y^T X w - (Xw)^T y + (Xw)^T (Xw)$$

$$= \arg\min_{w} \; y^T y - y^T X w - w^T X^T y + w^T X^T X w$$

More identities:

$$(ABC)^T = C^T B^T A^T$$

Solving least squares in matrix notation

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

As before, first simplify the function to make it easier to work with

$$= \arg\min_{w} \; y^T y - y^T X w - (Xw)^T y + (Xw)^T (Xw)$$

$$= \arg\min_{w} \; y^T y - y^T X w - w^T X^T y + w^T X^T X w$$

More identities:

$$(ABC)^T = C^T B^T A^T$$

Minimizing wrt w, so?

Solving least squares in matrix notation

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

As before, first simplify the function to make it easier to work with

$$= \arg\min_{w} \; y^T y - y^T Xw - (Xw)^T y + (Xw)^T (Xw)$$

$$= \arg\min_{w} \; y^T y - y^T Xw - w^T X^T y + w^T X^T Xw$$

More identities:

$$(ABC)^T = C^T B^T A^T$$

Minimizing wrt w, so?

Solving least squares in matrix notation

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

As before, first simplify the function to make it easier to work with

$$= \arg\min_{w} \; y^T y - y^T X w - (Xw)^T y + (Xw)^T (Xw)$$

$$= \arg\min_{w} \; y^T y - y^T X w - w^T X^T y + w^T X^T X w$$

More identities:

$$(ABC)^T = C^T B^T A^T$$

Minimizing wrt w, so?

Check dimensions

Solving least squares in matrix notation

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

As before, first simplify the function to make it easier to work with

$$= \arg\min_{w} \; y^T y - y^T Xw - (Xw)^T y + (Xw)^T (Xw)$$

$$= \arg\min_{w} \; y^T y - y^T Xw - w^T X^T y + w^T X^T Xw$$

More identities:

$$(ABC)^T = C^T B^T A^T$$

Minimizing wrt w, so?

Check dimensions

Solving least squares in matrix notation

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

As before, first simplify the function to make it easier to work with

$$= \arg\min_{w} \; y^T y - y^T Xw - (Xw)^T y + (Xw)^T (Xw)$$

$$= \arg\min_{w} \; \cancel{y^T y} - \underbrace{y^T Xw}_{1\times n} - w^T X^T y + w^T X^T Xw$$

More identities:

$$(ABC)^T = C^T B^T A^T$$

Minimizing wrt w, so?

Check dimensions

Solving least squares in matrix notation

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

As before, first simplify the function to make it easier to work with

$$= \arg\min_{w} \; y^T y - y^T X w - (Xw)^T y + (Xw)^T (Xw)$$

$$= \arg\min_{w} \; \cancel{y^T y} - \underbrace{y^T X w}_{1\times n \; n\times d} - w^T X^T y + w^T X^T X w$$

More identities:

$$(ABC)^T = C^T B^T A^T$$

Minimizing wrt w, so?

Check dimensions

Solving least squares in matrix notation

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

As before, first simplify the function to make it easier to work with

$$= \arg\min_{w} \; y^T y - y^T X w - (Xw)^T y + (Xw)^T (Xw)$$

$$= \arg\min_{w} \; \cancel{y^T y} - \underbrace{y^T X w}_{1\times n \; n\times d \; d\times 1} - w^T X^T y + w^T X^T X w$$

More identities:

$$(ABC)^T = C^T B^T A^T$$

Minimizing wrt w, so?

Check dimensions

Solving least squares in matrix notation

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

As before, first simplify the function to make it easier to work with

$$= \arg\min_{w} \; y^T y - y^T X w - (Xw)^T y + (Xw)^T (Xw)$$

$$= \arg\min_{w} \; \cancel{y^T y} - \underbrace{y^T X w}_{1\times n \; n\times d \; d\times 1} - w^T X^T y + w^T X^T X w$$

1×1

More identities:

$$(ABC)^T = C^T B^T A^T$$

Minimizing wrt w, so?

Check dimensions

Solving least squares in matrix notation

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

As before, first simplify the function to make it easier to work with

$$= \arg\min_{w} \; y^T y - y^T X w - (Xw)^T y + (Xw)^T (Xw)$$

$$= \arg\min_{w} \; \cancel{y^T y} - \underbrace{y^T X w}_{1\times n \; n\times d \; d\times 1} - w^T X^T y + w^T X^T X w$$

1×1

More identities:

$$(ABC)^T = C^T B^T A^T$$

Minimizing wrt w, so?

Check dimensions

Solving least squares in matrix notation

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

As before, first simplify the function to make it easier to work with

$$= \arg\min_{w} \; y^T y - y^T Xw - (Xw)^T y + (Xw)^T (Xw)$$

$$= \arg\min_{w} \; y^T y - \underset{1\times n \; n\times d \; d\times 1}{y^T Xw} - \underset{1\times d}{w^T X^T} y + w^T X^T Xw$$

1×1

More identities:

$$(ABC)^T = C^T B^T A^T$$

Minimizing wrt w, so?

Check dimensions

Solving least squares in matrix notation

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

As before, first simplify the function to make it easier to work with

$$= \arg\min_{w} \; y^T y - y^T X w - (Xw)^T y + (Xw)^T (Xw)$$

$$= \arg\min_{w} \; \cancel{y^T y} - \underbrace{y^T X w}_{1\times n \; n\times d \; d\times 1} - \underbrace{w^T X^T y}_{1\times d \; d\times n} + w^T X^T X w$$

1×1

More identities:

$$(ABC)^T = C^T B^T A^T$$

Minimizing wrt w, so?

Check dimensions

Solving least squares in matrix notation

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

As before, first simplify the function to make it easier to work with

$$= \arg\min_{w} \; y^T y - y^T X w - (Xw)^T y + (Xw)^T (Xw)$$

$$= \arg\min_{w} \; \cancel{y^T y} - \underbrace{y^T X w}_{1\times n \; n\times d \; d\times 1} - \underbrace{w^T X^T y}_{1\times d \; d\times n \; n\times 1} + w^T X^T X w$$

1×1

More identities:

$$(ABC)^T = C^T B^T A^T$$

Minimizing wrt w, so?

Check dimensions

Solving least squares in matrix notation

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

As before, first simplify the function to make it easier to work with

$$= \arg\min_{w} \; y^T y - y^T Xw - (Xw)^T y + (Xw)^T (Xw)$$

$$= \arg\min_{w} \; y^T y - \underbrace{y^T Xw}_{1\times n \; n\times d \; d\times 1} - \underbrace{w^T X^T y}_{1\times d \; d\times n \; n\times 1} + w^T X^T Xw$$

1×1 $\qquad$ 1×1

More identities:

$$(ABC)^T = C^T B^T A^T$$

Minimizing wrt w, so?

Check dimensions

Solving least squares in matrix notation

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

As before, first simplify the function to make it easier to work with

$$= \arg\min_{w} \; y^T y - y^T Xw - (Xw)^T y + (Xw)^T (Xw)$$

$$= \arg\min_{w} \; \cancel{y^T y} - \underbrace{y^T Xw}_{1\times n \; n\times d \; d\times 1} - \underbrace{w^T X^T y}_{1\times d \; d\times n \; n\times 1} + w^T X^T Xw$$

1×1 $\qquad$ 1×1

More identities:

$$(ABC)^T = C^T B^T A^T$$

Minimizing wrt w, so?

Check dimensions

$$s \in \mathbb{R} \rightarrow s = s^T$$

Solving least squares in matrix notation

$$= \arg\min_w (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

As before, first simplify the function to make it easier to work with

$$= \arg\min_w \; y^T y - y^T Xw - (Xw)^T y + (Xw)^T (Xw)$$

$$= \arg\min_w \; \cancel{y^T y} - \underbrace{y^T Xw}_{1\times n \; n\times d \; d\times 1} - \underbrace{(w^T X^T y)}_{1\times d \; d\times n \; n\times 1}^T + w^T X^T Xw$$

1×1 $\qquad$ 1×1

More identities:

$$(ABC)^T = C^T B^T A^T$$

Minimizing wrt w, so?

Check dimensions

$$s \in \mathbb{R} \rightarrow s = s^T$$

Solving least squares in matrix notation

$$= \arg\min_w (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

As before, first simplify the function to make it easier to work with

$$= \arg\min_w \; y^T y - y^T Xw - (Xw)^T y + (Xw)^T (Xw)$$

$$= \arg\min_w \; \cancel{y^T y} - \underbrace{y^T Xw}_{1\times n \; n\times d \; d\times 1} - \underbrace{(w^T X^T y)}_{1\times d \; d\times n \; n\times 1}{}^T + w^T X^T Xw$$

1×1 $\quad$ 1×1

$$= \arg\min_w \; - y^T Xw - y^T Xw + w^T X^T Xw$$

More identities:

$$(ABC)^T = C^T B^T A^T$$

Minimizing wrt w, so?

Check dimensions

$$s \in \mathbb{R} \rightarrow s = s^T$$

Solving least squares in matrix notation

$$= \arg\min_w (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

As before, first simplify the function to make it easier to work with

$$= \arg\min_w \; y^T y - y^T Xw - (Xw)^T y + (Xw)^T (Xw)$$

$$= \arg\min_w \; \cancel{y^T y} - \underbrace{y^T Xw}_{1\times n \; n\times d \; d\times 1} - \underbrace{(w^T X^T y)^T}_{1\times d \; d\times n \; n\times 1} + w^T X^T Xw$$

1×1 $\quad$ 1×1

$$= \arg\min_w \; - y^T Xw - y^T Xw + w^T X^T Xw$$

$$= \arg\min_w \; - 2y^T Xw + w^T X^T Xw$$

More identities:

$$(ABC)^T = C^T B^T A^T$$

Minimizing wrt w, so?

Check dimensions

$$s \in \mathbb{R} \rightarrow s = s^T$$

Solving least squares in matrix notation

$$= \arg\min_w (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

As before, first simplify the function to make it easier to work with

$$= \arg\min_w \; y^T y - y^T Xw - (Xw)^T y + (Xw)^T (Xw)$$

$$= \arg\min_w \; \cancel{y^T y} - \underbrace{y^T Xw}_{1\times n \; n\times d \; d\times 1} - \underbrace{(w^T X^T y)}_{1\times d \; d\times n \; n\times 1}^T + w^T X^T Xw$$

1×1 $\quad$ 1×1

$$= \arg\min_w \; - y^T Xw - y^T Xw + w^T X^T Xw$$

$$= \arg\min_w \; - 2y^T Xw + w^T X^T Xw$$

More identities:

$$(ABC)^T = C^T B^T A^T$$

Minimizing wrt w, so?

Check dimensions

$$s \in \mathbb{R} \rightarrow s = s^T$$

Pretty simple!

Solving least squares in matrix notation

$$= \arg\min_w (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

As before, first simplify the function to make it easier to work with

$$= \arg\min_w \; y^T y - y^T Xw - (Xw)^T y + (Xw)^T (Xw)$$

$$= \arg\min_w \; \cancel{y^T y} - \underbrace{y^T Xw}_{1\times n \; n\times d \; d\times 1} - \underbrace{(w^T X^T y)}_{1\times d \; d\times n \; n\times 1}^T + w^T X^T Xw$$

1×1 $\quad$ 1×1

$$= \arg\min_w \; - y^T Xw - y^T Xw + w^T X^T Xw$$

$$= \arg\min_w \; - 2y^T Xw + w^T X^T Xw$$

More identities:

$$(ABC)^T = C^T B^T A^T$$

Minimizing wrt w, so?

Check dimensions

$$s \in \mathbb{R} \rightarrow s = s^T$$

Pretty simple!

So now what?

Least squares: take the gradient and set to 0

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

Take derivative, set to 0!

$$\nabla_w [-2y^T Xw + w^T X^T Xw] = 0$$

Least squares: take the gradient and set to 0

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

$$\nabla_w [-2y^T Xw + w^T X^T Xw] = 0$$

Take derivative, set to 0!

Useful matrix gradient identities

(A) $\nabla_w x^T w = x$

(B) $\nabla_w x^T Aw = A^T x$

(C) $\nabla_w w^T Aw = (A + A^T)w$

Quadratic form

Least squares: take the gradient and set to 0

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

$$\nabla_w [-2y^T Xw + w^T X^T Xw] = 0$$

Take derivative, set to 0!

Useful matrix gradient identities

(A) $\nabla_w x^T w = x$

(B) $\nabla_w x^T Aw = A^T x$

(C) $\nabla_w w^T Aw = (A + A^T)w$

Quadratic form

If A is symmetric ($A = A^T$),
Then $\nabla_w w^T Aw = 2Aw$

Least squares: take the gradient and set to 0

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

$$\nabla_w [-2y^T Xw + w^T X^T Xw] = 0$$

Take derivative, set to 0!

Useful matrix gradient identities

(A) $\nabla_w x^T w = x$

(B) $\nabla_w x^T A w = A^T x$

(C) $\nabla_w w^T A w = (A + A^T) w$

Quadratic form

If A is symmetric ($A = A^T$),
Then $\nabla_w w^T A w = 2Aw$

Least squares: take the gradient and set to 0

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

$$\nabla_w [-2y^T Xw + w^T X^T Xw] = 0$$

(B)

Take derivative, set to 0!

Useful matrix gradient identities

(A) $\nabla_w x^T w = x$

(B) $\nabla_w x^T Aw = A^T x$

(C) $\nabla_w w^T Aw = (A + A^T)w$

Quadratic form

If A is symmetric ($A = A^T$),
Then $\nabla_w w^T Aw = 2Aw$

Least squares: take the gradient and set to 0

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

$$\nabla_w [\, -2y^T Xw + w^T X^T Xw] = 0$$

(B)

Take derivative, set to 0!

Useful matrix gradient identities

(A) $\nabla_w x^T w = x$

(B) $\nabla_w x^T Aw = A^T x$

(C) $\nabla_w w^T Aw = (A + A^T)w$

Quadratic form

If A is symmetric ($A = A^T$),
Then $\nabla_w w^T Aw = 2Aw$

Least squares: take the gradient and set to 0

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

$$\nabla_w [\underline{-2y^T Xw} + \underline{w^T X^T Xw}] = 0$$

(B)

Take derivative, set to 0!

Useful matrix gradient identities

(A) $\nabla_w x^T w = x$

(B) $\nabla_w x^T A w = A^T x$

(C) $\nabla_w w^T A w = (A + A^T) w$

Quadratic form

If A is symmetric ($A = A^T$),
Then $\nabla_w w^T A w = 2Aw$

Symmetric: $(X^T X)^T = X^T X$

Least squares: take the gradient and set to 0

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

$$\nabla_w [\underbrace{-2y^T Xw}_{(B)} + \underbrace{w^T X^T Xw}_{(C)}] = 0$$

Take derivative, set to 0!

Useful matrix gradient identities

(A) $\nabla_w x^T w = x$

(B) $\nabla_w x^T A w = A^T x$

(C) $\nabla_w w^T A w = (A + A^T) w$

Quadratic form

If A is symmetric ($A = A^T$),
Then $\nabla_w w^T A w = 2Aw$

Symmetric: $(X^T X)^T = X^T X$

Least squares: take the gradient and set to 0

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

Take derivative, set to 0!

$$\nabla_w [\underbrace{-2y^T Xw}_{(B)} + \underbrace{w^T X^T Xw}_{(C)}] = 0$$

Useful matrix gradient identities

(A) $\nabla_w x^T w = x$

(B) $\nabla_w x^T A w = A^T x$

(C) $\nabla_w w^T A w = (A + A^T) w$

Quadratic form

If A is symmetric ($A = A^T$),
Then $\nabla_w w^T A w = 2Aw$

$$= -2X^T y + 2X^T Xw = 0$$

Symmetric: $(X^T X)^T = X^T X$

Least squares: take the gradient and set to 0

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

Take derivative, set to 0!

$$\nabla_w [\underbrace{-2y^T Xw}_{(B)} + \underbrace{w^T X^T Xw}_{(C)}] = 0$$

Useful matrix gradient identities

(A) $\nabla_w x^T w = x$

(B) $\nabla_w x^T A w = A^T x$

(C) $\nabla_w w^T A w = (A + A^T) w$

Quadratic form

If A is symmetric ($A = A^T$),
Then $\nabla_w w^T A w = 2Aw$

$$= -2X^T y + 2X^T Xw = 0$$

Symmetric: $(X^T X)^T = X^T X$

Least squares: take the gradient and set to 0

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

Take derivative, set to 0!

$$\nabla_w [\underbrace{-2y^T Xw}_{(B)} + \underbrace{w^T X^T Xw}_{(C)}] = 0$$

Useful matrix gradient identities

(A) $\nabla_w x^T w = x$

(B) $\nabla_w x^T A w = A^T x$

(C) $\nabla_w w^T A w = (A + A^T) w$

Quadratic form

If A is symmetric ($A = A^T$),
Then $\nabla_w w^T A w = 2Aw$

$$= -\cancel{2}X^T y + \cancel{2}X^T Xw = 0$$

Symmetric: $(X^T X)^T = X^T X$

Least squares: take the gradient and set to 0

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

Take derivative, set to 0!

$$\nabla_w [\underbrace{-2y^T Xw}_{(B)} + \underbrace{w^T X^T Xw}_{(C)}] = 0$$

Useful matrix gradient identities

(A) $\nabla_w x^T w = x$

(B) $\nabla_w x^T A w = A^T x$

(C) $\nabla_w w^T A w = (A + A^T) w$

Quadratic form

If A is symmetric ($A = A^T$),
Then $\nabla_w w^T A w = 2Aw$

$$= -\cancel{2}X^T y + \cancel{2}X^T Xw = 0$$

$$X^T Xw = X^T y$$

Symmetric: $(X^T X)^T = X^T X$

Least squares: take the gradient and set to 0

$$= \arg\min_w (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

Take derivative, set to 0!

$$\nabla_w [\underbrace{-2y^T Xw}_{(B)} + \underbrace{w^T X^T Xw}_{(C)}] = 0$$

Useful matrix gradient identities

(A) $\nabla_w x^T w = x$

(B) $\nabla_w x^T Aw = A^T x$

(C) $\nabla_w w^T Aw = (A + A^T)w$

Quadratic form

If A is symmetric ($A = A^T$),
Then $\nabla_w w^T Aw = 2Aw$

$$= -2X^T y + 2X^T Xw = 0$$

Symmetric: $(X^T X)^T = X^T X$

$$X^T Xw = X^T y$$

$$\hat{w}_{\text{MLE}} = (X^T X)^{-1} X^T y$$

Least squares: take the gradient and set to 0

$$= \arg\min_w (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

Take derivative, set to 0!

$$\nabla_w [\underbrace{-2y^T Xw}_{(B)} + \underbrace{w^T X^T Xw}_{(C)}] = 0$$

Useful matrix gradient identities

(A) $\nabla_w x^T w = x$

(B) $\nabla_w x^T Aw = A^T x$

(C) $\nabla_w w^T Aw = (A + A^T)w$

Quadratic form

If A is symmetric ($A = A^T$),
Then $\nabla_w w^T Aw = 2Aw$

$$= -2X^T y + 2X^T Xw = 0$$

Symmetric: $(X^T X)^T = X^T X$

$$X^T Xw = X^T y$$

$$\hat{w}_{\text{MLE}} = (X^T X)^{-1} X^T y$$

Least squares solution in matrix notation

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}$$

d : # of features
n : # of examples/datapoints

More identities: ℓ_2 norm: $\|z\|_2 = \sqrt{\sum_{i=1}^{n} z_i^2} = \sqrt{z^\top z}$

$$\begin{aligned} \widehat{w}_{LS} &= \arg\min_{w} ||\mathbf{y} - \mathbf{X}w||_2^2 \\ &= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w) \end{aligned}$$

$$\hat{w}_{LS} = \hat{w}_{MLE} = \left(\boldsymbol{X}^T \boldsymbol{X}\right)^{-1} \boldsymbol{X}^T \boldsymbol{y}$$

Least squares solution in matrix notation

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}$$

d : # of features
n : # of examples/datapoints

More identities: ℓ_2 norm: $\|z\|_2 = \sqrt{\sum_{i=1}^{n} z_i^2} = \sqrt{z^\top z}$

$$\begin{aligned} \widehat{w}_{LS} &= \arg\min_{w} ||\mathbf{y} - \mathbf{X}w||_2^2 \\ &= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w) \end{aligned}$$

$$\hat{w}_{LS} = \hat{w}_{MLE} = \left(\boldsymbol{X}^T \boldsymbol{X}\right)^{-1} \boldsymbol{X}^T \boldsymbol{y}$$

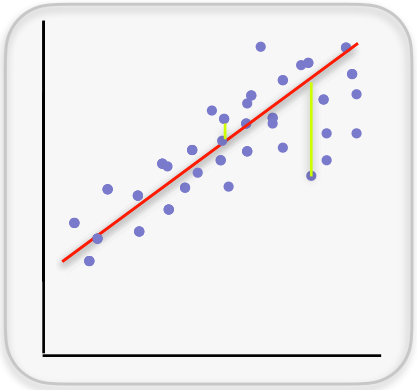
Why LS? “Ordinary Least Squares”

Regression… what about an intercept?

$$\begin{aligned}\widehat{w}_{LS} &= \arg\min_{w} ||\mathbf{y} - \mathbf{X}w||_2^2 \\ &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}\end{aligned}$$

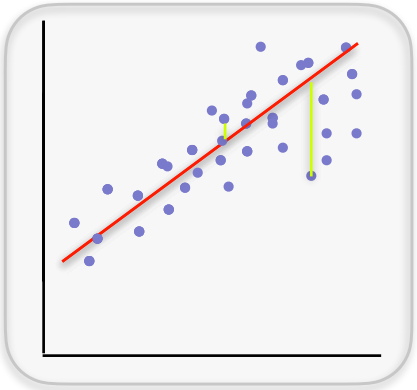
Regression… what about an intercept?

$$\widehat{w}_{LS} = \arg\min_{w} ||\mathbf{y} - \mathbf{X}w||_2^2$$
$$= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$



Regression… what about an intercept?

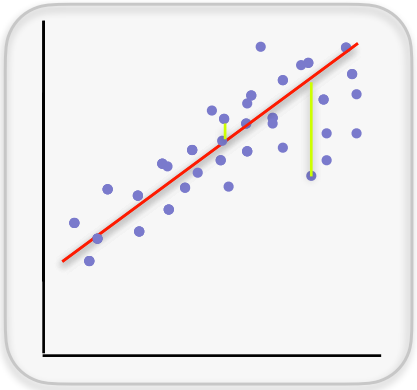
$$\widehat{w}_{LS} = \arg\min_{w} ||\mathbf{y} - \mathbf{X}w||_2^2$$
$$= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$



What about an offset?

Regression… what about an intercept?

$$\begin{aligned}\widehat{w}_{LS} &= \arg\min_{w} ||\mathbf{y} - \mathbf{X}w||_2^2 \\ &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}\end{aligned}$$

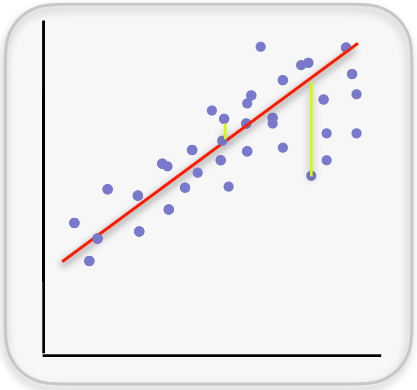


What about an offset?

$$\widehat{w}_{LS}, \widehat{b}_{LS} = \arg\min_{w,b} \sum_{i=1}^{n} \left(y_i - (x_i^T w + b)\right)^2$$

Regression… what about an intercept?

$$\widehat{w}_{LS} = \arg\min_{w} ||\mathbf{y} - \mathbf{X}w||_2^2$$
$$= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$



What about an offset?

$$\widehat{w}_{LS}, \widehat{b}_{LS} = \arg\min_{w,b} \sum_{i=1}^{n} \left(y_i - (x_i^T w + b)\right)^2$$
$$= \arg\min_{w,b} ||\mathbf{y} - (\mathbf{X}w + \mathbf{1}b)||_2^2$$

Dealing with an offset

$$\widehat{w}_{LS}, \widehat{b}_{LS} = \arg\min_{w,b} ||\mathbf{y} - (\mathbf{X}w + \mathbf{1}b)||_2^2$$

$$\mathbf{X}^T\mathbf{X}\widehat{w}_{LS} + \widehat{b}_{LS}\mathbf{X}^T\mathbf{1} = \mathbf{X}^T\mathbf{y}$$

$$\mathbf{1}^T\mathbf{X}\widehat{w}_{LS} + \widehat{b}_{LS}\mathbf{1}^T\mathbf{1} = \mathbf{1}^T\mathbf{y}$$

If $\mathbf{X}^T\mathbf{1} = 0$ (i.e., if each feature is mean-zero) then

$$\widehat{w}_{LS} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

$$\widehat{b}_{LS} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

Dealing with an offset

$$\widehat{w}_{LS}, \widehat{b}_{LS} = \arg\min_{w,b} ||\mathbf{y} - (\mathbf{X}w + \mathbf{1}b)||_2^2$$

$$\mathbf{X}^T\mathbf{X}\widehat{w}_{LS} + \widehat{b}_{LS}\mathbf{X}^T\mathbf{1} = \mathbf{X}^T\mathbf{y}$$
$$\mathbf{1}^T\mathbf{X}\widehat{w}_{LS} + \widehat{b}_{LS}\mathbf{1}^T\mathbf{1} = \mathbf{1}^T\mathbf{y}$$

If $\mathbf{X}^T\mathbf{1} = 0$,

$$\widehat{w}_{LS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

$$\widehat{b}_{LS} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

In general, when $\mathbf{X}^T\mathbf{1} \neq 0$,

$$\mu = \frac{1}{n}\mathbf{X}^T\mathbf{1}$$

$$\widetilde{\mathbf{X}} = \mathbf{X} - \mathbf{1}\mu^T$$

$$\widehat{w}_{\mathrm{LS}} = (\widetilde{\mathbf{X}}^T\widetilde{\mathbf{X}})^{-1}\widetilde{\mathbf{X}}^T\mathbf{y}$$

$$\widehat{b}_{\mathrm{LS}} = \frac{1}{n}\mathbf{1}^T\mathbf{y} - \mu^T\widehat{w}_{\mathrm{LS}}$$

How do we make predictions from this model?

$$\widehat{w}_{LS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

$$\widehat{b}_{LS} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

A new potato order is ready. How long will drone delivery take?

How do we make predictions from this model?

$$\widehat{w}_{LS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

$$\widehat{b}_{LS} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

A new potato order is ready. How long will drone delivery take?

$$\hat{y}_{\text{new}} = x_{\text{new}}^T\hat{w}_{LS} + \hat{b}_{LS}$$

Process

Decide on a **model** for the likelihood function $f(x; \theta)$

Find the function which fits the data best

Choose a loss function- least squares

Pick the function which minimizes loss on data

Use function to make prediction on new examples

Process

Decide on a **model:** $y_i = x_i^T w + b + \epsilon_i$

Choose a loss function - least squares, or log likelihood

Pick the function which minimizes loss on data

$$\widehat{w}_{LS}, \widehat{b}_{LS} = \arg\min_{w,b} \sum_{i=1}^{n} \left(y_i - (x_i^T w + b)\right)^2$$

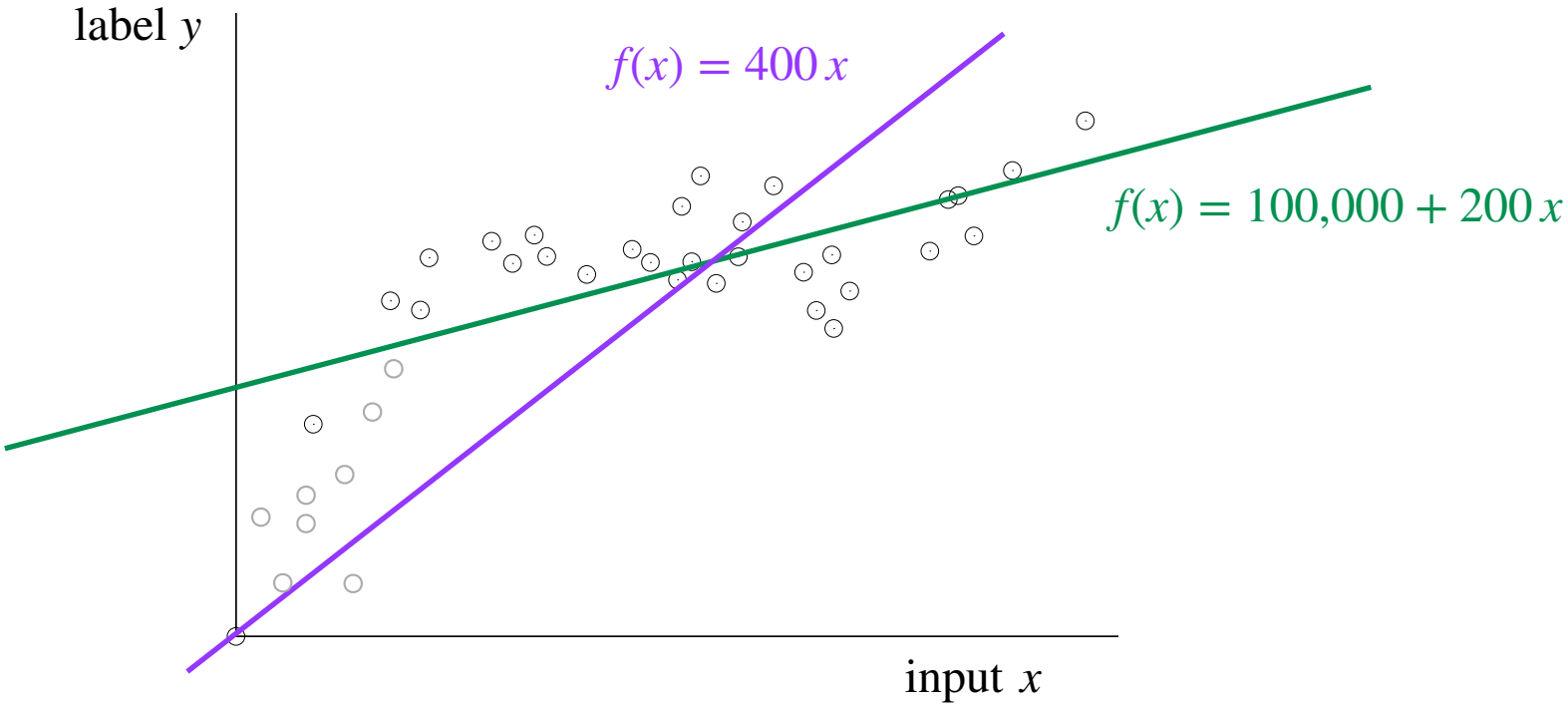
Use function to make prediction on new examples

$$\hat{y}_{\text{new}} = x_{\text{new}}^T \hat{w}_{LS} + \hat{b}_{LS}$$

~~Linear~~ Regression

Generalized Linear/ Polynomial

Recap: Linear Regression



- In general high-dimensions, we fit a linear model with intercept

$$y_i \simeq w^T x_i + b\,, \quad \text{or equivalently} \quad y_i = w^T x_i + b + \epsilon_i$$

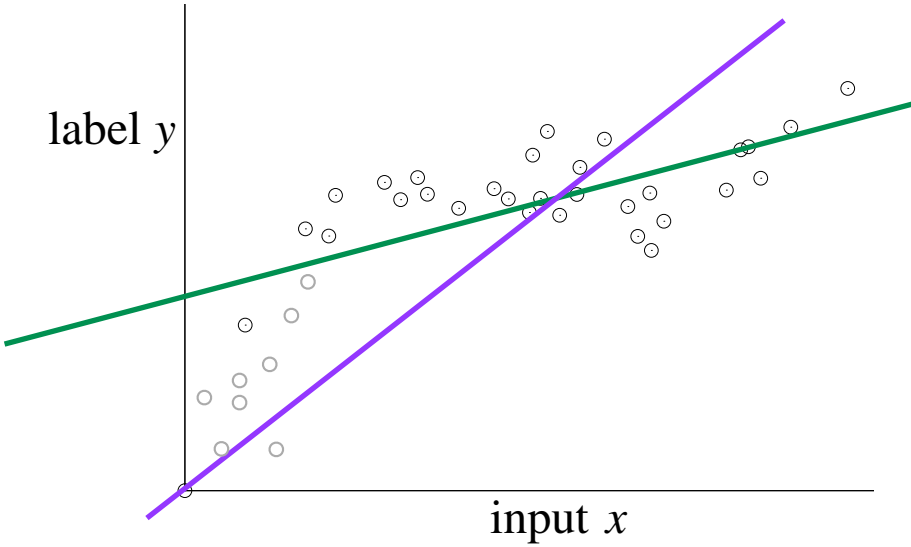
with model parameters $(w \in \mathbb{R}^d, b \in \mathbb{R})$ that minimizes ℓ_2-loss

$$\mathscr{L}(w,b) \;=\; \sum_{i=1}^{n} \underbrace{(y_i - (w^T x_i + b))^2}_{\text{error } \epsilon_i} \qquad \text{\# MSE}$$

Quadratic regression in 1-dimension

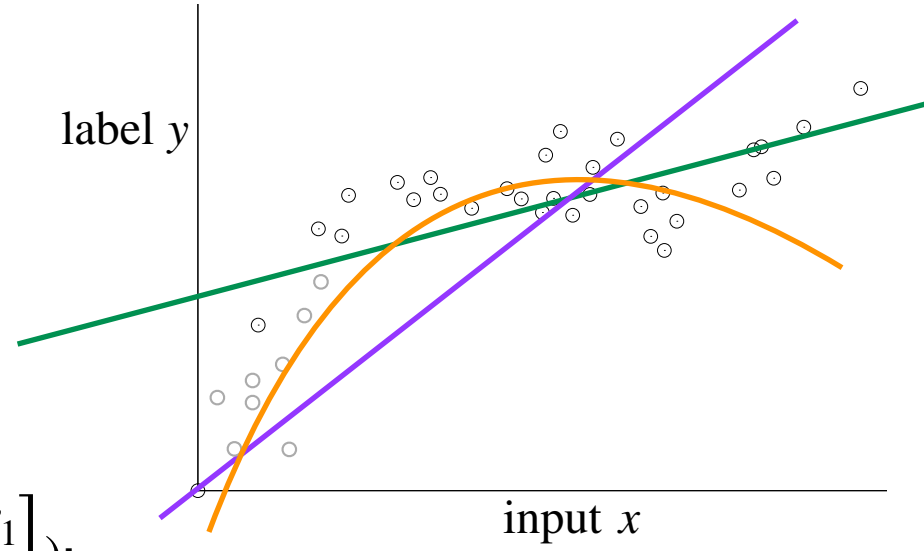
- **Data:** $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$

- **Linear model with parameter** (b, w_1)**:**
  - $\hat{y}_i = b + w_1 x_i$



Quadratic regression in 1-dimension

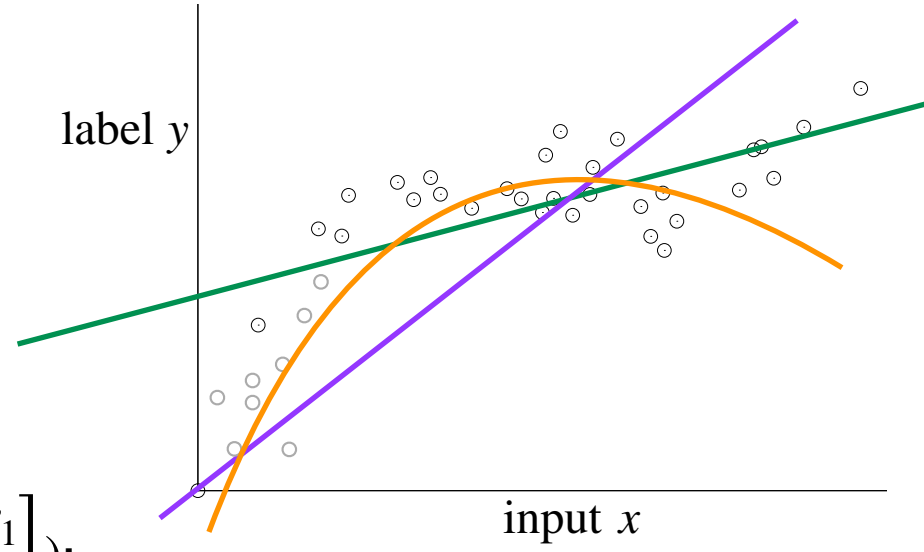
- **Data:** $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$



- **Linear model with parameter** (b, w_1)**:**
  - $\hat{y}_i = b + w_1 x_i$
- **Quadratic model with parameter** $(b, w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix})$**:**
  - $\hat{y}_i = b + w_1 x_i + w_2 x_i^2$

Quadratic regression in 1-dimension

- **Data:** $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$

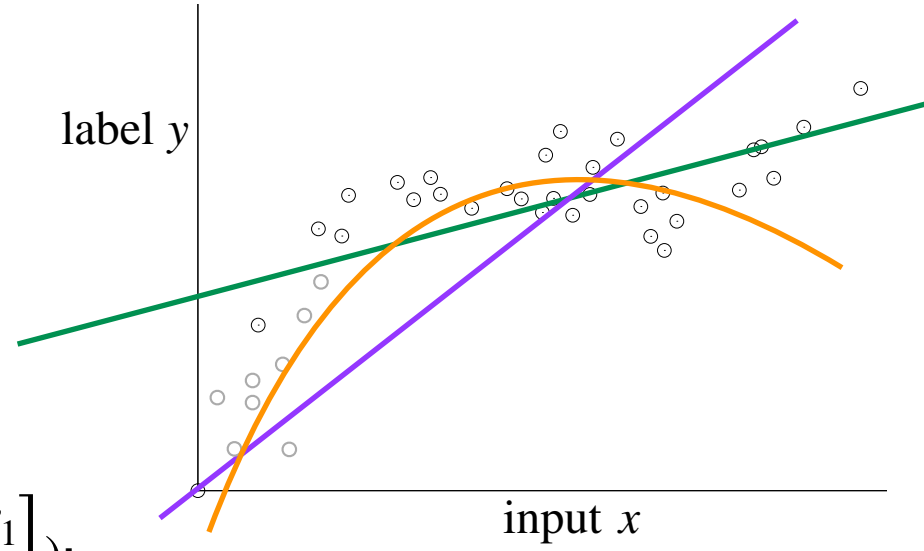


- **Linear model with parameter** (b, w_1)**:**
  - $\hat{y}_i = b + w_1 x_i$
- **Quadratic model with parameter** $(b, w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix})$**:**
  - $\hat{y}_i = b + w_1 x_i + w_2 x_i^2$

$$h(x_i) = \begin{bmatrix} 1 \\ x_i \\ x_i^2 \end{bmatrix}$$

Quadratic regression in 1-dimension

- **Data:** $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$

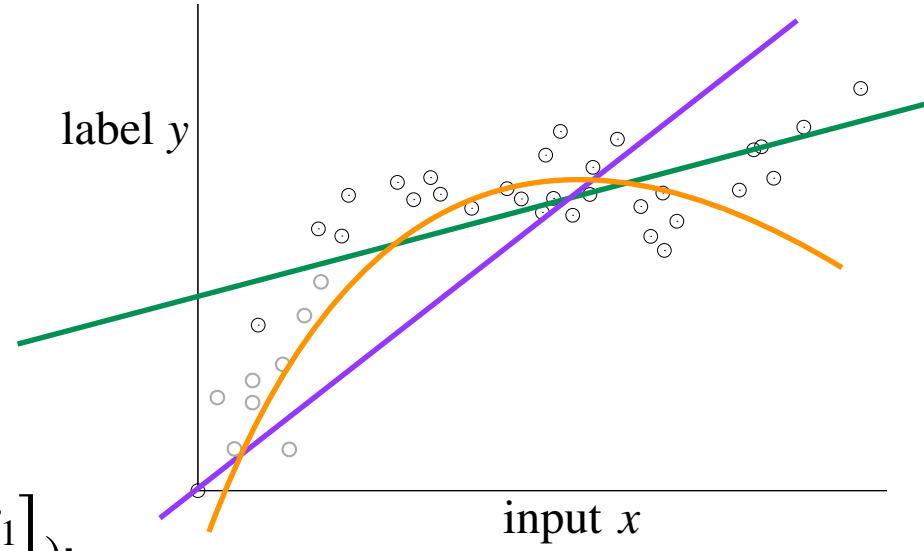


- **Linear model with parameter** (b, w_1)**:**
  - $\hat{y}_i = b + w_1 x_i$
- **Quadratic model with parameter** $(b, w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix})$**:**
  - $\hat{y}_i = b + w_1 x_i + w_2 x_i^2$

$$h(x_i) = \begin{bmatrix} 1 \\ x_i \\ x_i^2 \end{bmatrix} \quad \hat{y} = h(x_i)^T \begin{bmatrix} b \\ w_1 \\ w_2 \\ \vdots \end{bmatrix}$$

Quadratic regression in 1-dimension

- **Data:** $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$



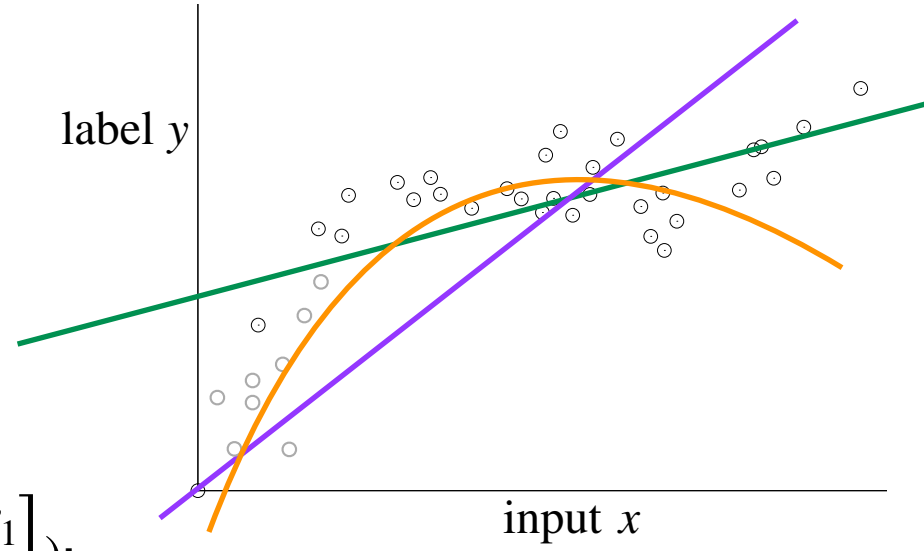
- **Linear model with parameter** (b, w_1)**:**
  - $\hat{y}_i = b + w_1 x_i$
- **Quadratic model with parameter** $(b, w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix})$**:**
  - $\hat{y}_i = b + w_1 x_i + w_2 x_i^2$

$$h(x_i) = \begin{bmatrix} 1 \\ x_i \\ x_i^2 \end{bmatrix} \quad \hat{y} = h(x_i)^T \begin{bmatrix} b \\ w_1 \\ w_2 \\ \vdots \end{bmatrix}$$

Still linear regression, but on quadratic features

Quadratic regression in 1-dimension

- **Data:** $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$



- **Linear model with parameter** (b, w_1)**:**
  - $\hat{y}_i = b + w_1 x_i$
- **Quadratic model with parameter** $(b, w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix})$**:**
  - $\hat{y}_i = b + w_1 x_i + w_2 x_i^2$

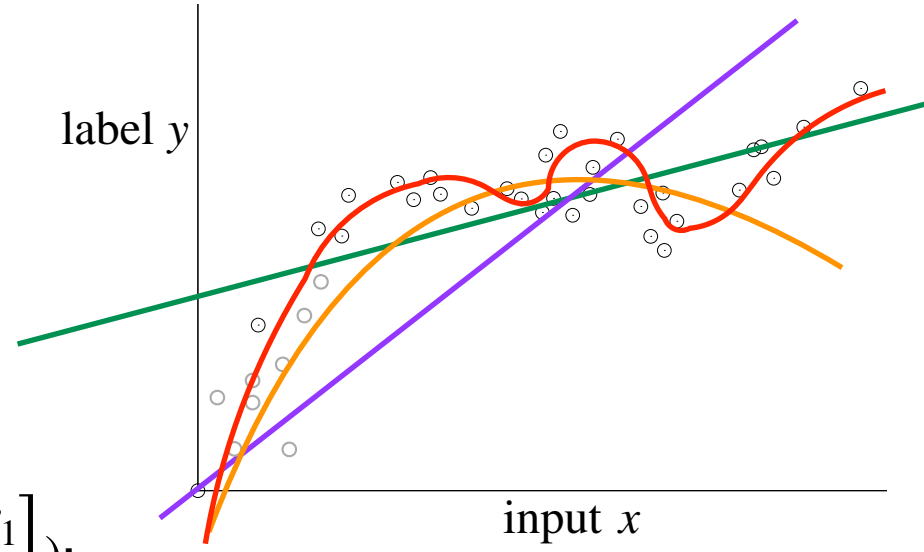
$$h(x_i) = \begin{bmatrix} 1 \\ x_i \\ x_i^2 \end{bmatrix} \quad \hat{y} = h(x_i)^T \begin{bmatrix} b \\ w_1 \\ w_2 \\ \vdots \end{bmatrix}$$

Still linear regression, but on quadratic features

Linear layer fit to non-linear features

Quadratic regression in 1-dimension

- **Data:** $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$



- **Linear model with parameter** (b, w_1)**:**
  - $\hat{y}_i = b + w_1 x_i$
- **Quadratic model with parameter** $(b, w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix})$**:**
  - $\hat{y}_i = b + w_1 x_i + w_2 x_i^2$
- **Degree-p polynomial model with *p* parameters**
  - $\hat{y}_i = b + w_1 x_i + w_2 x_i^2 + \ldots + w_p x_i^p$

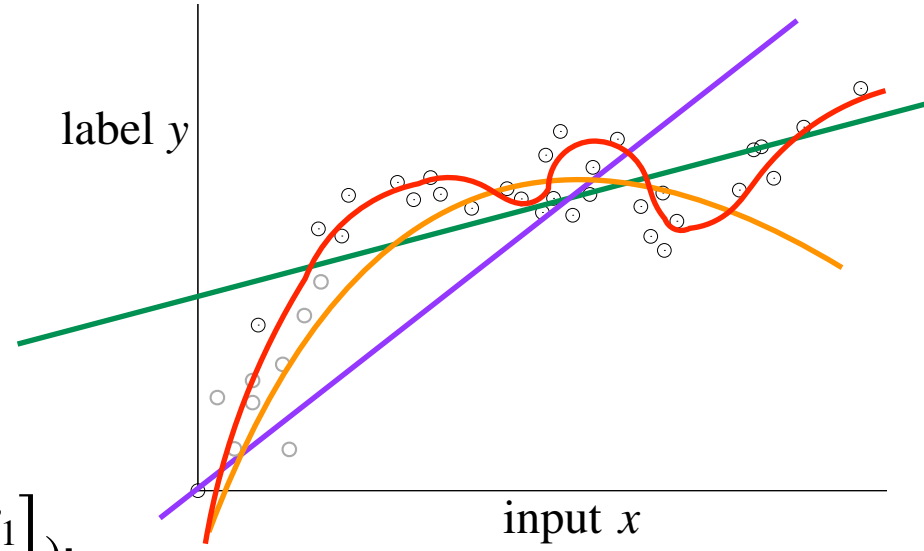
$$h(x_i) = \begin{bmatrix} 1 \\ x_i \\ x_i^2 \end{bmatrix} \quad \hat{y} = h(x_i)^T \begin{bmatrix} b \\ w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix}$$

Still linear regression, but on quadratic features

Linear layer fit to non-linear features

Quadratic regression in 1-dimension

- **Data:** $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$



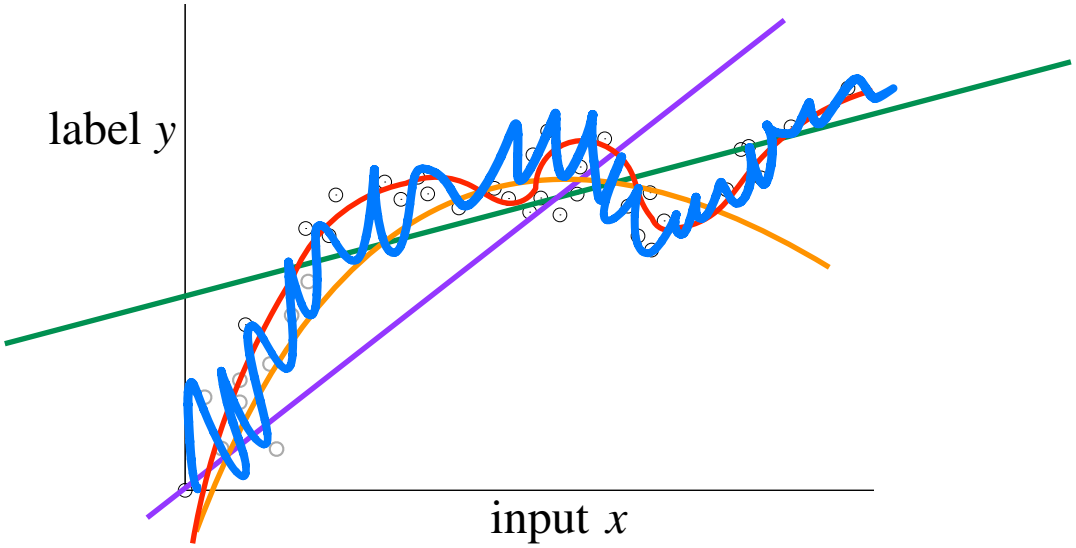
- **Linear model with parameter** (b, w_1)**:**
  - $\hat{y}_i = b + w_1 x_i$
- **Quadratic model with parameter** $(b, w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix})$**:**
  - $\hat{y}_i = b + w_1 x_i + w_2 x_i^2$
- **Degree-p polynomial model with *p* parameters**
  - $\hat{y}_i = b + w_1 x_i + w_2 x_i^2 + \ldots + w_p x_i^p$
- **General p-features with parameter** $w = \begin{bmatrix} w_1 \\ \vdots \\ w_p \end{bmatrix}$**:**
  - $\hat{y}_i = \langle w, h(x_i) \rangle$ where $h : \mathbb{R} \to \mathbb{R}^p$

$$h(x_i) = \begin{bmatrix} 1 \\ x_i \\ x_i^2 \end{bmatrix} \quad \hat{y} = h(x_i)^T \begin{bmatrix} b \\ w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix}$$

Still linear regression, but on quadratic features

Linear layer fit to non-linear features

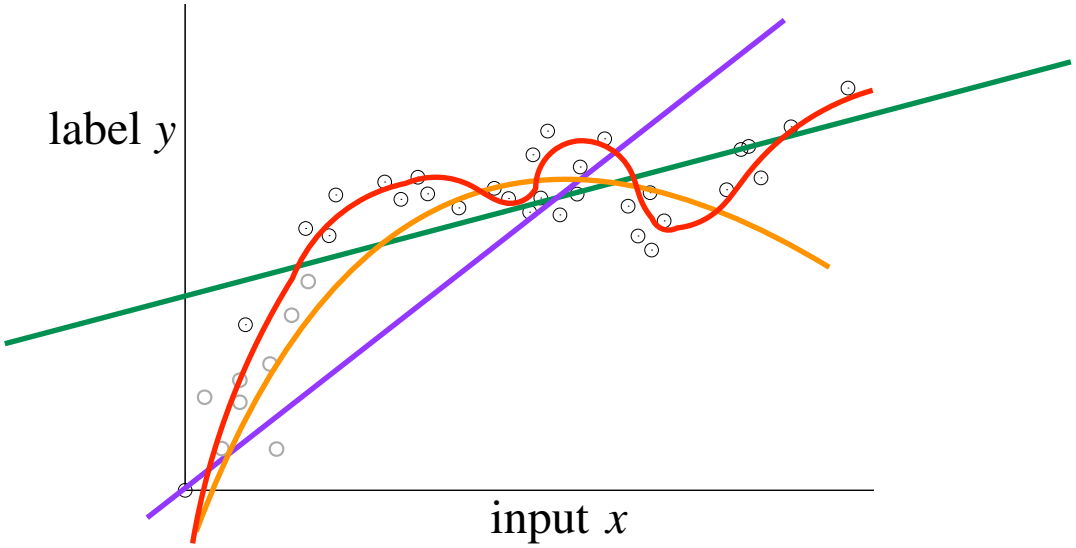
Quadratic regression in 1-dimension



Which line gives us the best fit?

- **Green** & **purple**?
- **Orange**?
- **Red**?

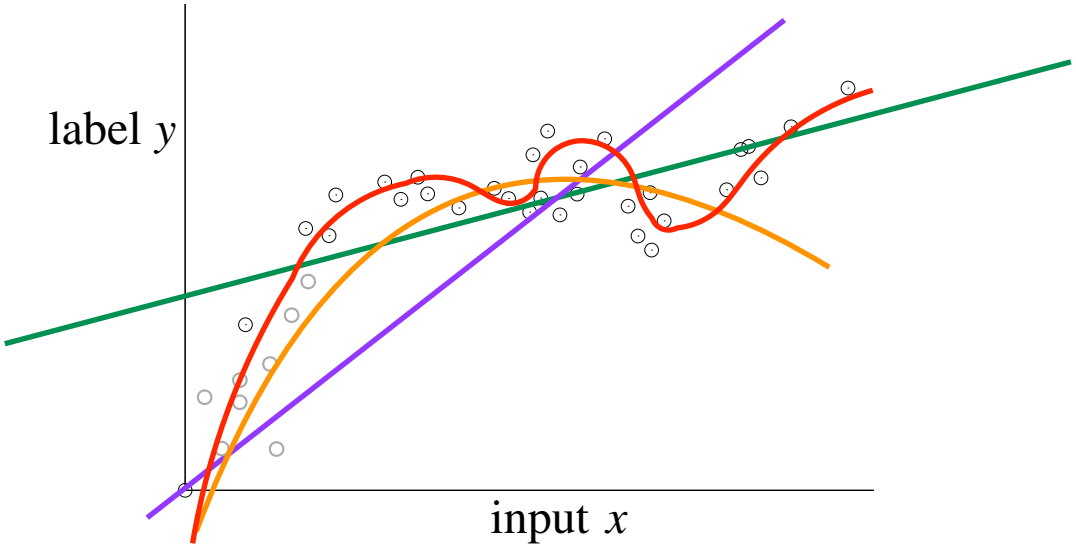
Quadratic regression in 1-dimension



Which line gives us the best fit?

- **Green** & **purple**? Can't capture variance. **Underfitting**
- **Orange**?
- **Red**?

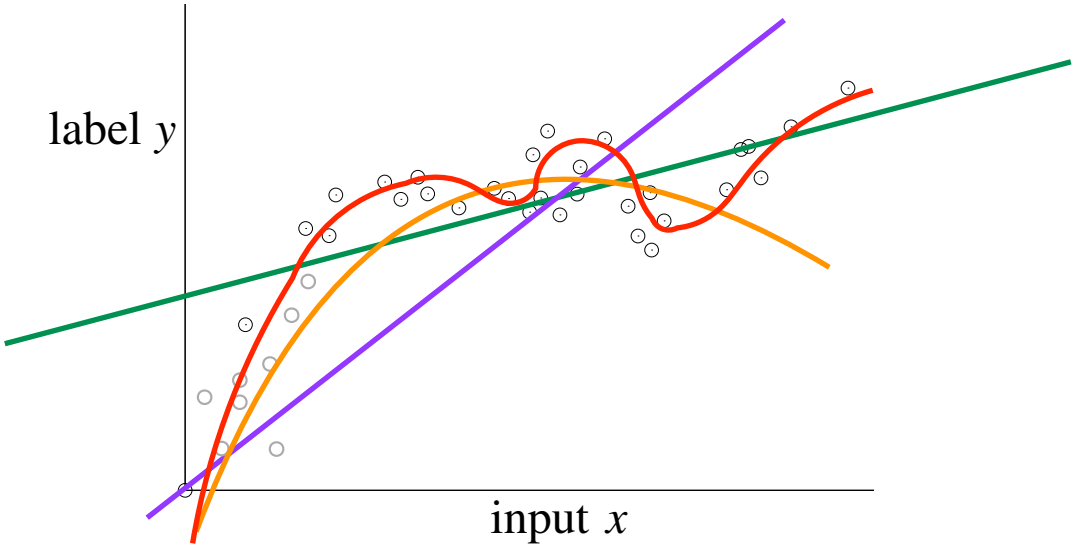
Quadratic regression in 1-dimension



Which line gives us the best fit?

- **Green** & **purple**? Can't capture variance. **Underfitting**
- **Orange**? Can't capture variance. **Underfitting**
- **Red**?

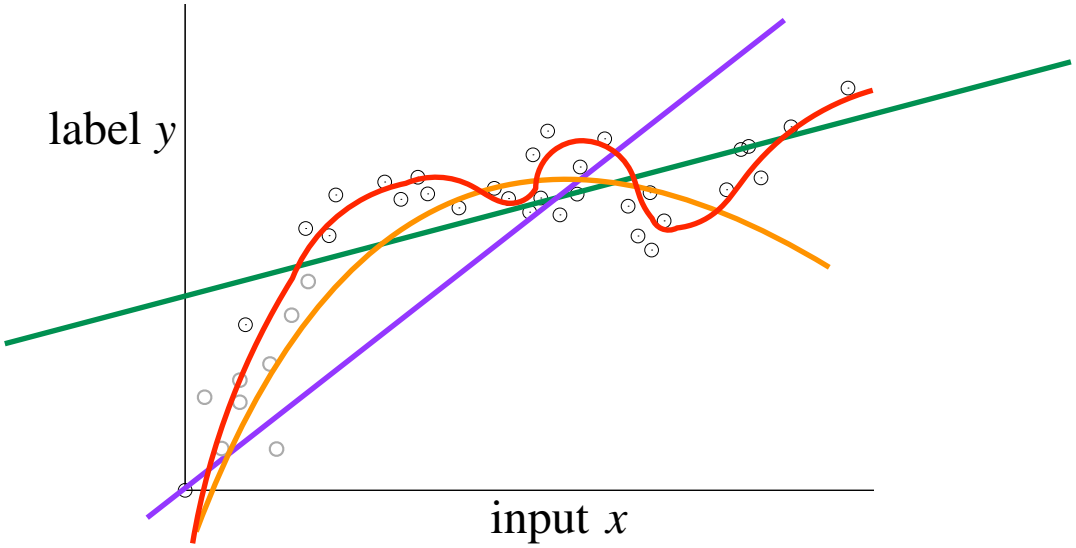
Quadratic regression in 1-dimension



Which line gives us the best fit?

- **Green** & **purple**? Can't capture variance. **Underfitting**
- **Orange**? Can't capture variance. **Underfitting**
- **Red**? Just right?

Quadratic regression in 1-dimension

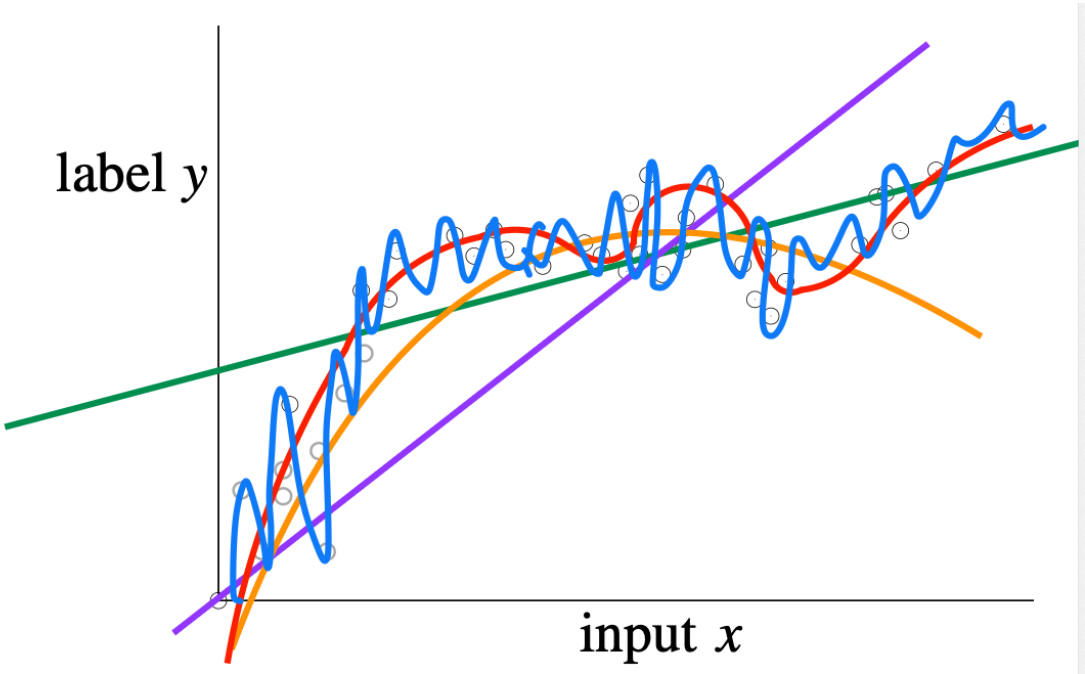


Which line gives us the best fit?

- **Green** & **purple**? Can't capture variance. **Underfitting**
- **Orange**? Can't capture variance. **Underfitting**
- **Red**? Just right?

  Could we take it too far?

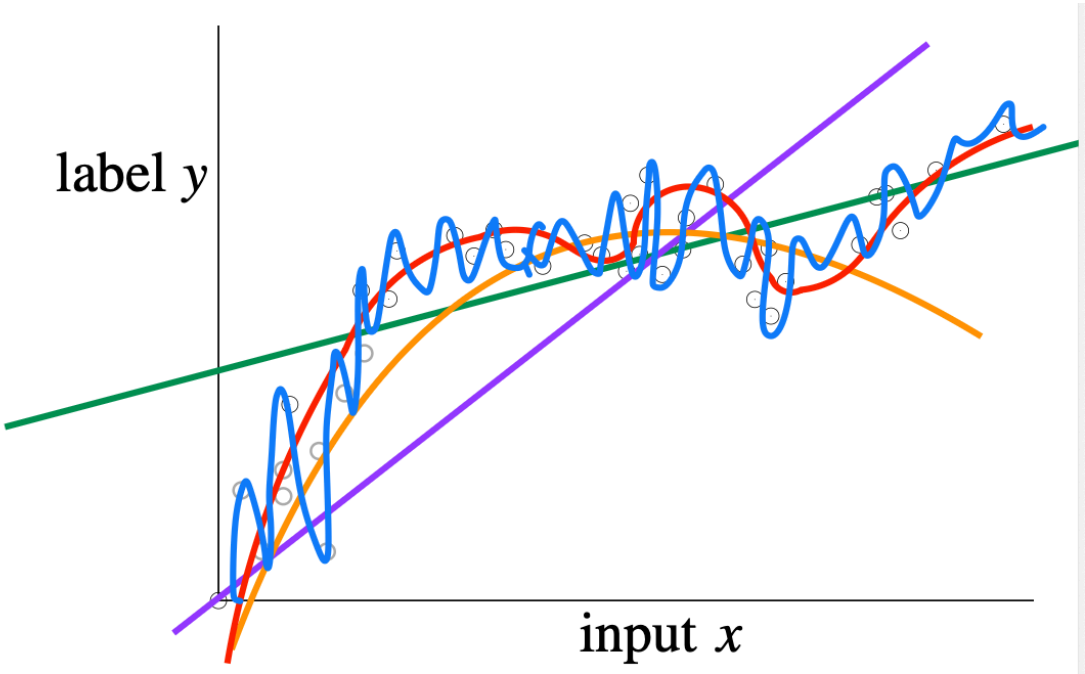
Quadratic regression in 1-dimension



Which line gives us the best fit?

- **Green** & **purple**? Can't capture variance. **Underfitting**
- **Orange**? Can't capture variance. **Underfitting**
- **Red**? Just right?
- **Blue**?

Quadratic regression in 1-dimension

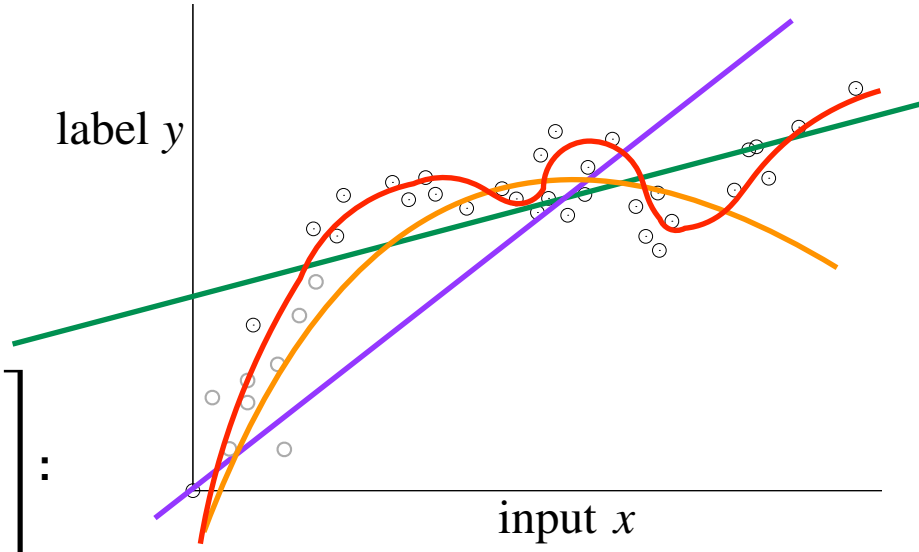


Which line gives us the best fit?

- **Green** & **purple**? Can't capture variance. **Underfitting**
- **Orange**? Can't capture variance. **Underfitting**
- **Red**? Just right?
- **Blue**? Fits random fluctuations / measurement errors in training dataset. **Overfitting.**

Quadratic regression in 1-dimension

- **Data:** $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$



- **General p-features with parameter** $w = \begin{bmatrix} w_1 \\ \vdots \\ w_p \end{bmatrix}$:
  - $\hat{y}_i = \langle w, h(x_i) \rangle$ where $h : \mathbb{R} \to \mathbb{R}^p$

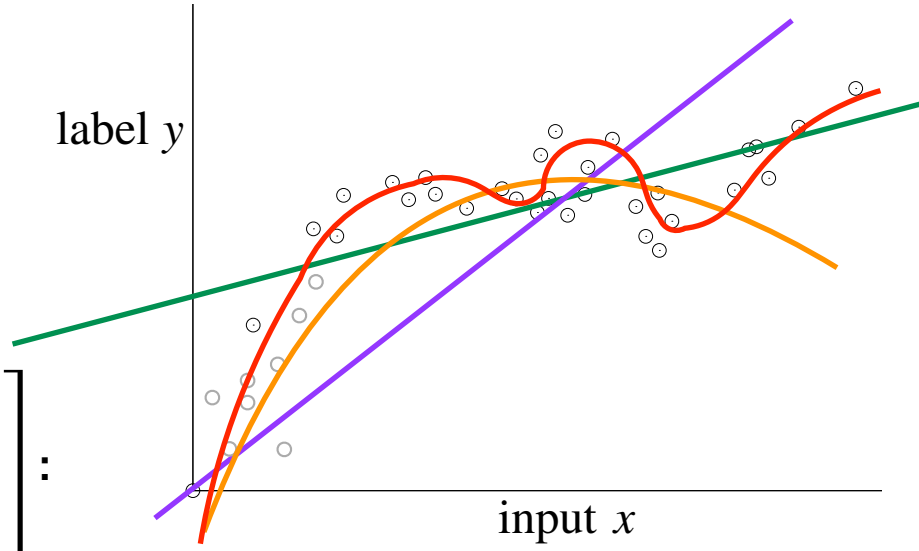
~~Breakout discussion~~:

Note: h can be arbitrary non-linear functions!

$$h(x) = \left[\log(x), x^2, \sin(x), \sqrt{x}\right]^\top$$

Quadratic regression in 1-dimension

- **Data:** $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$



- **General p-features with parameter** $w = \begin{bmatrix} w_1 \\ \vdots \\ w_p \end{bmatrix}$:
  - $\hat{y}_i = \langle w, h(x_i) \rangle$ where $h : \mathbb{R} \to \mathbb{R}^p$

Breakout discussion:

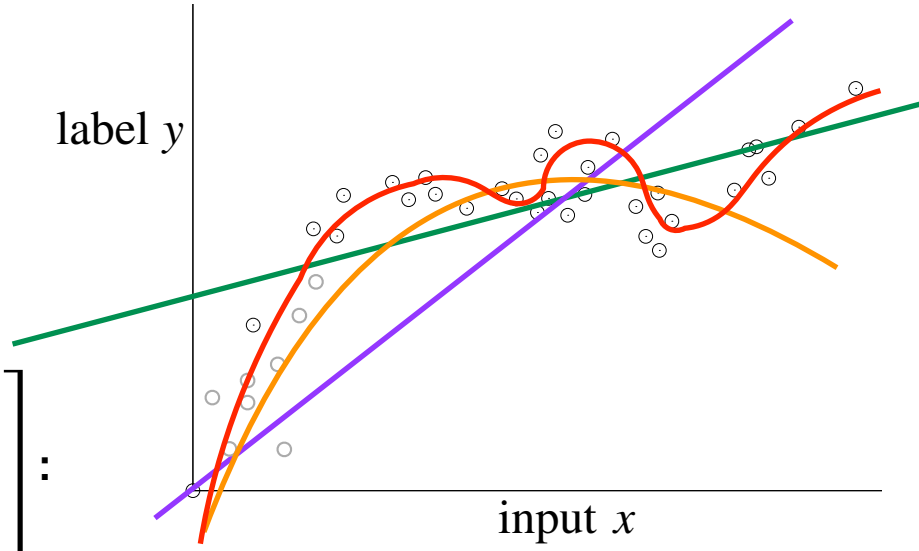
Note: h can be arbitrary non-linear functions!

$$h(x) = \left[\log(x), x^2, \sin(x), \sqrt{x}\right]^\top$$

Transform features however

Quadratic regression in 1-dimension

- **Data:** $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$



- **General p-features with parameter** $w = \begin{bmatrix} w_1 \\ \vdots \\ w_p \end{bmatrix}$:
  - $\hat{y}_i = \langle w, h(x_i) \rangle$ where $h : \mathbb{R} \to \mathbb{R}^p$

Note: h can be arbitrary non-linear functions!

$$h(x) = \left[\log(x), x^2, \sin(x), \sqrt{x}\right]^\top$$

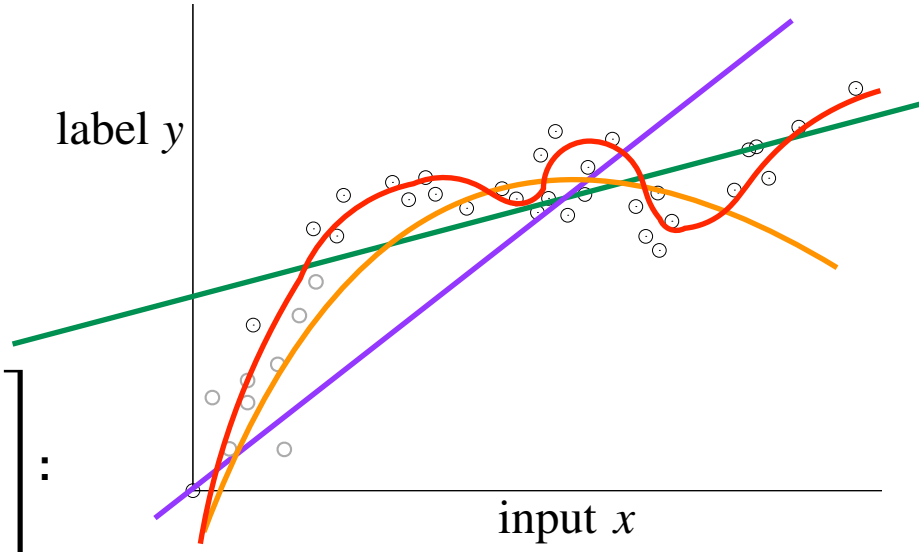
Transform features however

Breakout discussion:

- When would you use sin(x)?
- How should I transform the loc feature: zip code?

Quadratic regression in 1-dimension

- **Data:** $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$



- **General p-features with parameter** $w = \begin{bmatrix} w_1 \\ \vdots \\ w_p \end{bmatrix}$:
  - $\hat{y}_i = \langle w, h(x_i) \rangle$ where $h : \mathbb{R} \to \mathbb{R}^p$

Note: h can be arbitrary non-linear functions!

$$h(x) = \left[\log(x), x^2, \sin(x), \sqrt{x}\right]^\top$$

Transform features however

Breakout discussion:

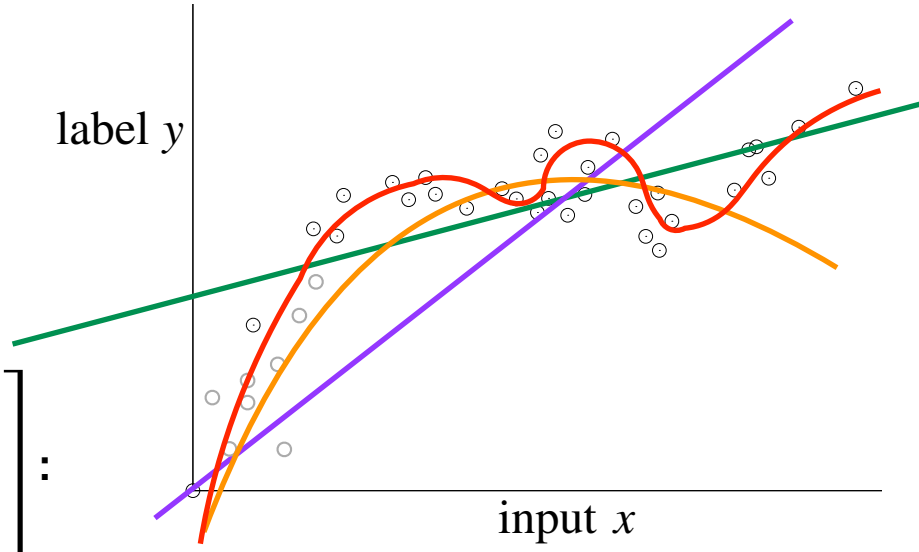
- When would you use sin(x)?

  # Cyclic data e.g. weather

- How should I transform the loc feature: zip code?

Quadratic regression in 1-dimension

- **Data:** $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$



- **General p-features with parameter** $w = \begin{bmatrix} w_1 \\ \vdots \\ w_p \end{bmatrix}$:
  - $\hat{y}_i = \langle w, h(x_i) \rangle$ where $h : \mathbb{R} \to \mathbb{R}^p$

Note: h can be arbitrary non-linear functions!

$$h(x) = \left[\log(x), x^2, \sin(x), \sqrt{x}\right]^\top$$

Transform features however

Breakout discussion:

- When would you use sin(x)?

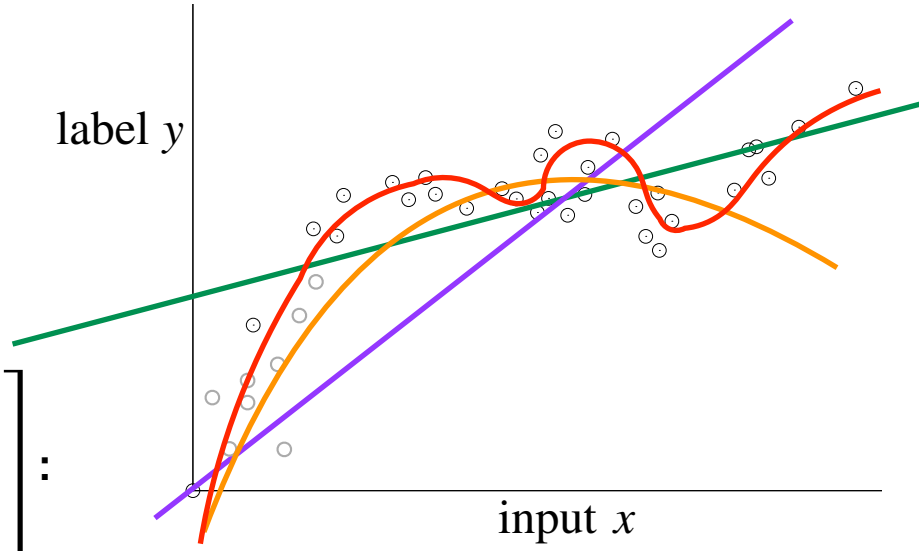
  # Cyclic data e.g. weather

- How should I transform the loc feature: zip code?

  # Lat / long

Quadratic regression in 1-dimension

- **Data:** $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$



- **General p-features with parameter** $w = \begin{bmatrix} w_1 \\ \vdots \\ w_p \end{bmatrix}$:
  - $\hat{y}_i = \langle w, h(x_i) \rangle$ where $h : \mathbb{R} \to \mathbb{R}^p$

Note: h can be arbitrary non-linear functions!

$$h(x) = \left[\log(x), x^2, \sin(x), \sqrt{x}\right]^\top$$

Transform features however

Breakout discussion:

- When would you use sin(x)?

  # Cyclic data e.g. weather

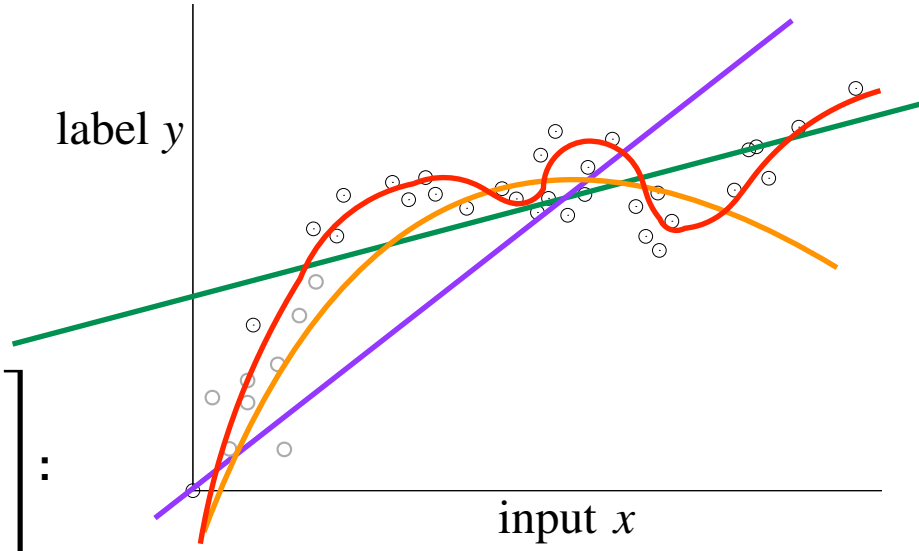
- How should I transform the loc feature: zip code?

  # Lat / long

This is the art of feature engineering

Quadratic regression in 1-dimension

- **Data:** $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$



- **General p-features with parameter** $w = \begin{bmatrix} w_1 \\ \vdots \\ w_p \end{bmatrix}$:
  - $\hat{y}_i = \langle w, h(x_i) \rangle$ where $h : \mathbb{R} \rightarrow \mathbb{R}^p$

Note: h can be arbitrary non-linear functions!

$$h(x) = \left[\log(x), x^2, \sin(x), \sqrt{x}\right]^\top$$

Transform features however

How will I know if I have good features?

Breakout discussion:

- When would you use sin(x)?

  # Cyclic data e.g. weather

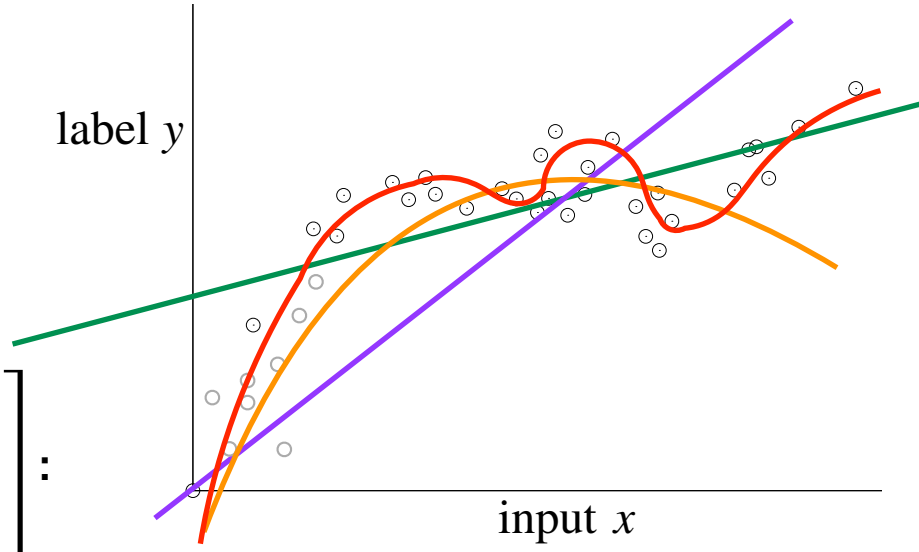
- How should I transform the loc feature: zip code?

  # Lat / long

This is the art of feature engineering

Quadratic regression in 1-dimension

- **Data:** $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$



- **General p-features with parameter** $w = \begin{bmatrix} w_1 \\ \vdots \\ w_p \end{bmatrix}$:
  - $\hat{y}_i = \langle w, h(x_i) \rangle$ where $h : \mathbb{R} \to \mathbb{R}^p$

Note: h can be arbitrary non-linear functions!

$$h(x) = \left[\log(x), x^2, \sin(x), \sqrt{x}\right]^\top$$

Transform features however

How will I know if I have good features?

PLOT PLOT PLOT PLOT PLOT

Breakout discussion:

- When would you use sin(x)?

  # Cyclic data e.g. weather

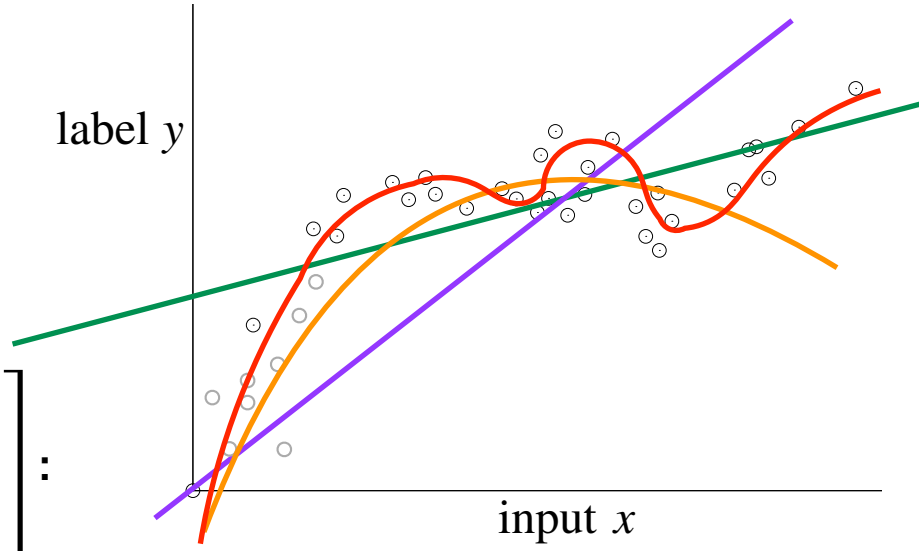
- How should I transform the loc feature: zip code?

  # Lat / long

This is the art of feature engineering

Quadratic regression in 1-dimension

- **Data:** $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$

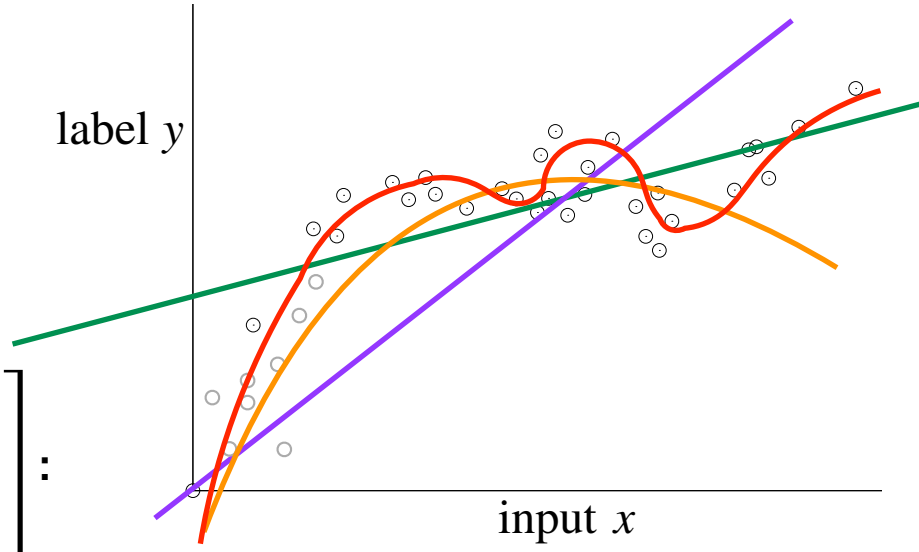


- **General p-features with parameter** $w = \begin{bmatrix} w_1 \\ \vdots \\ w_p \end{bmatrix}$:
  - $\hat{y}_i = \langle w, h(x_i) \rangle$ where $h : \mathbb{R} \to \mathbb{R}^p$

How do we learn w?

Quadratic regression in 1-dimension

- **Data:** $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$



- **General p-features with parameter** $w = \begin{bmatrix} w_1 \\ \vdots \\ w_p \end{bmatrix}$:
  - $\hat{y}_i = \langle w, h(x_i) \rangle$ where $h : \mathbb{R} \to \mathbb{R}^p$

How do we learn w?

$$\mathbf{H} = \begin{bmatrix} - - h(x_1)^\top - - \\ \vdots \\ - - h(x_n)^\top - - \end{bmatrix} \in \mathbb{R}^{n \times p}$$

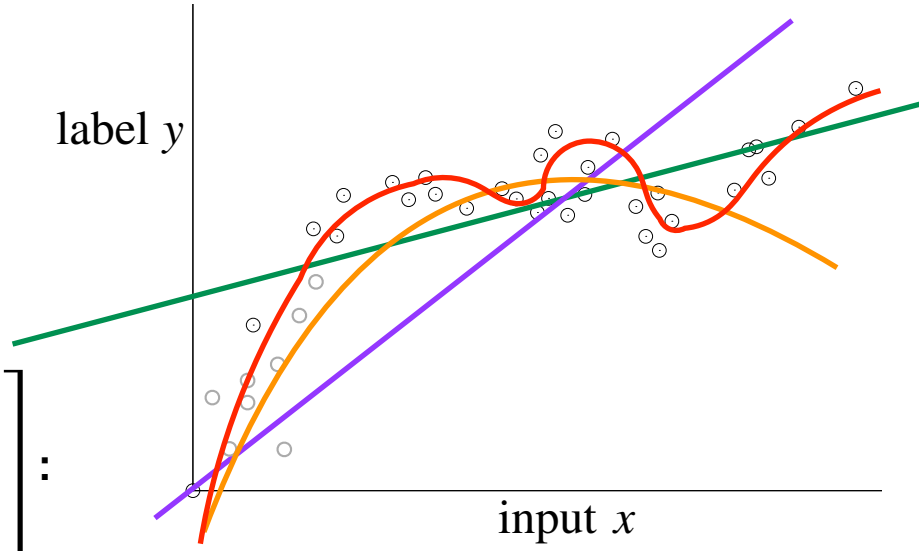
$$\widehat{w} = \arg\min_w \|\mathbf{H}w - \mathbf{y}\|_2^2$$

For a new test point x, predict

$\hat{y} = \langle \widehat{w}, h(x) \rangle$

Quadratic regression in 1-dimension

- **Data:** $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$



- **General p-features with parameter** $w = \begin{bmatrix} w_1 \\ \vdots \\ w_p \end{bmatrix}$:
  - $\hat{y}_i = \langle w, h(x_i) \rangle$ where $h : \mathbb{R} \to \mathbb{R}^p$

How do we learn w?

$$\mathbf{H} = \begin{bmatrix} - - h(x_1)^\top - - \\ \vdots \\ - - h(x_n)^\top - - \end{bmatrix} \in \mathbb{R}^{n \times p}$$

$$\widehat{w} = \arg\min_w \|\mathbf{H}w - \mathbf{y}\|_2^2$$

For a new test point x, predict

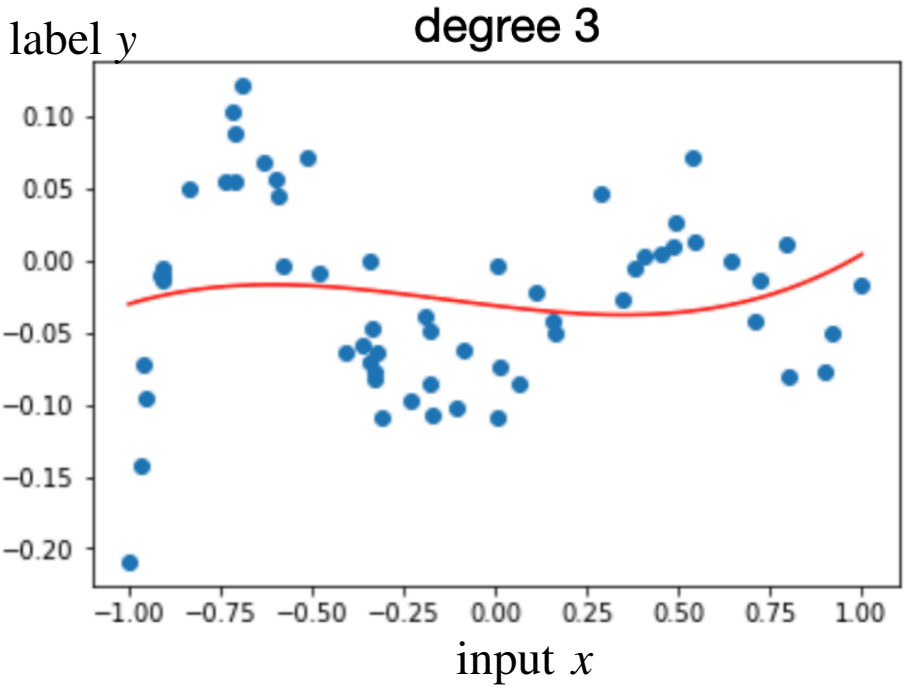
$\hat{y} = \langle \widehat{w}, h(x) \rangle$

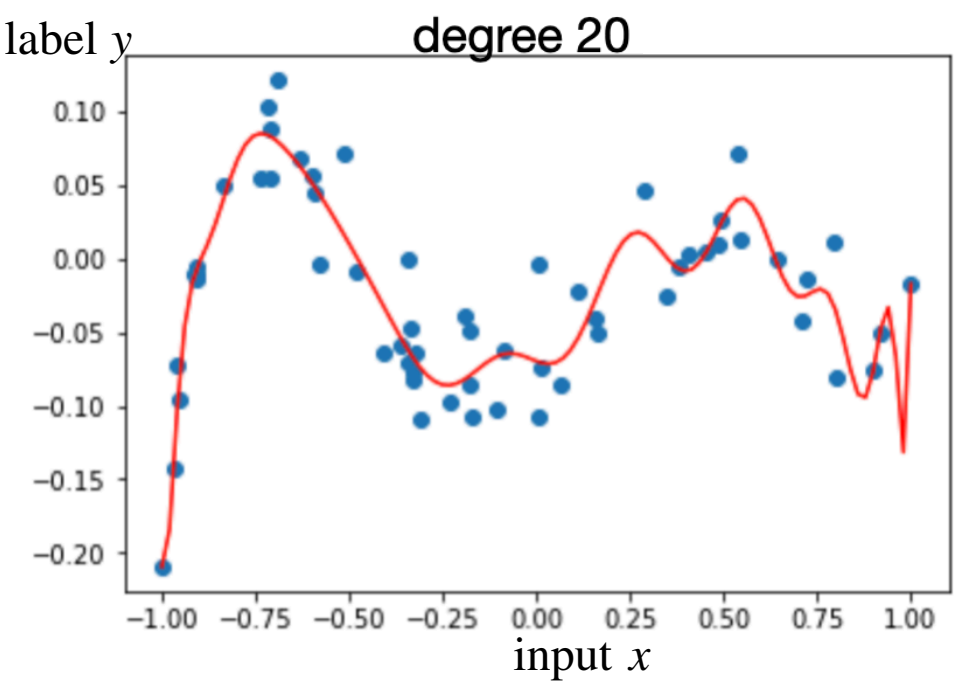
$\hat{w}_{\text{MLE}} = (H^T H)^{-1} H^T y$

- Training and test dataset split
- Cross validation

Which p should we choose?

- First instance of class of models with different representation power = model complexity

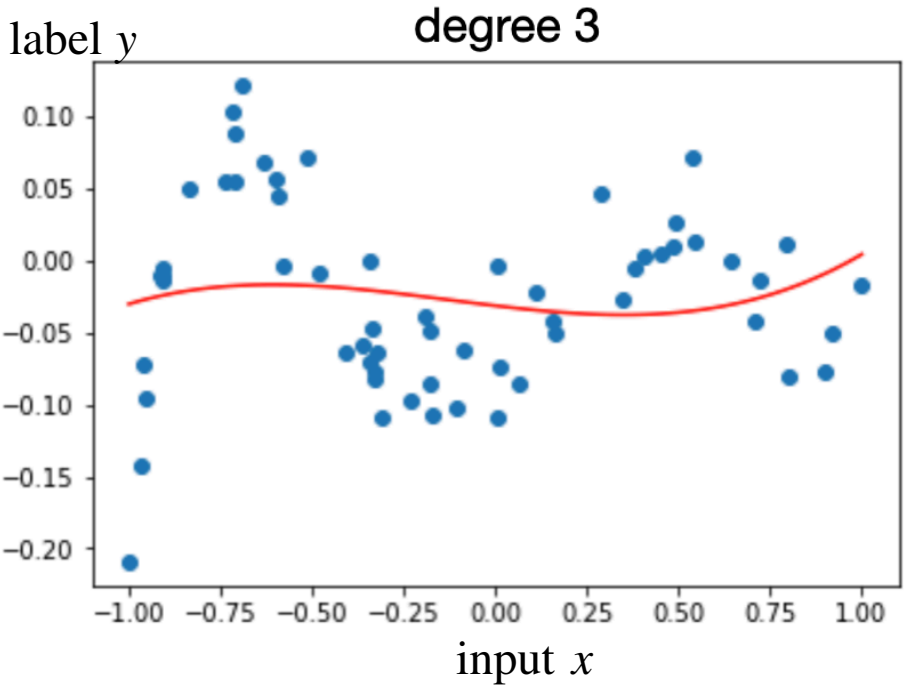


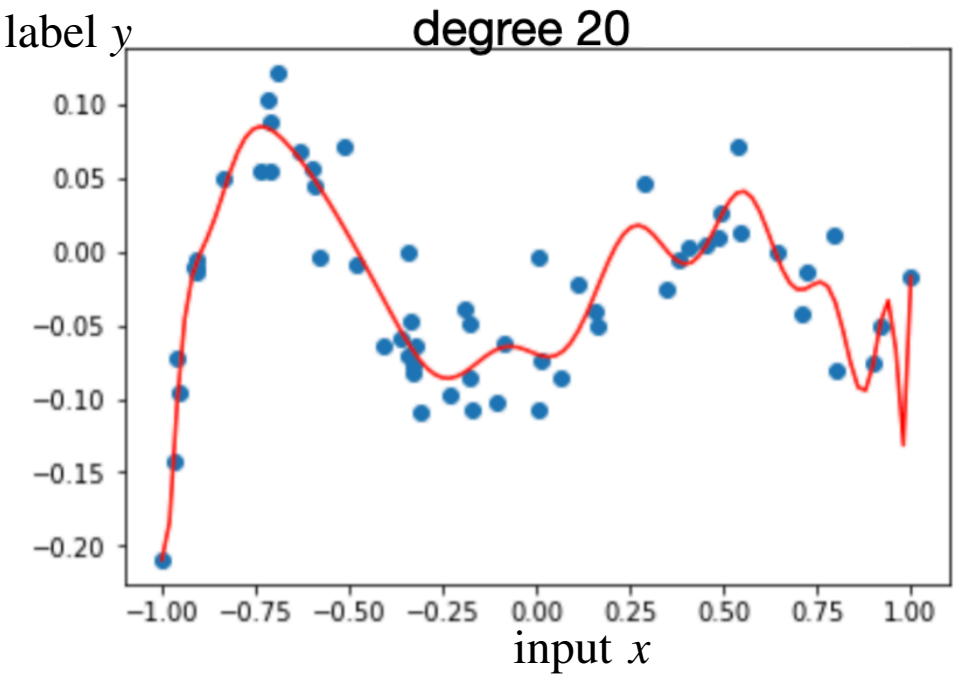


- How do we determine which is better model?

Which p should we choose?

- First instance of class of models with different representation power = model complexity



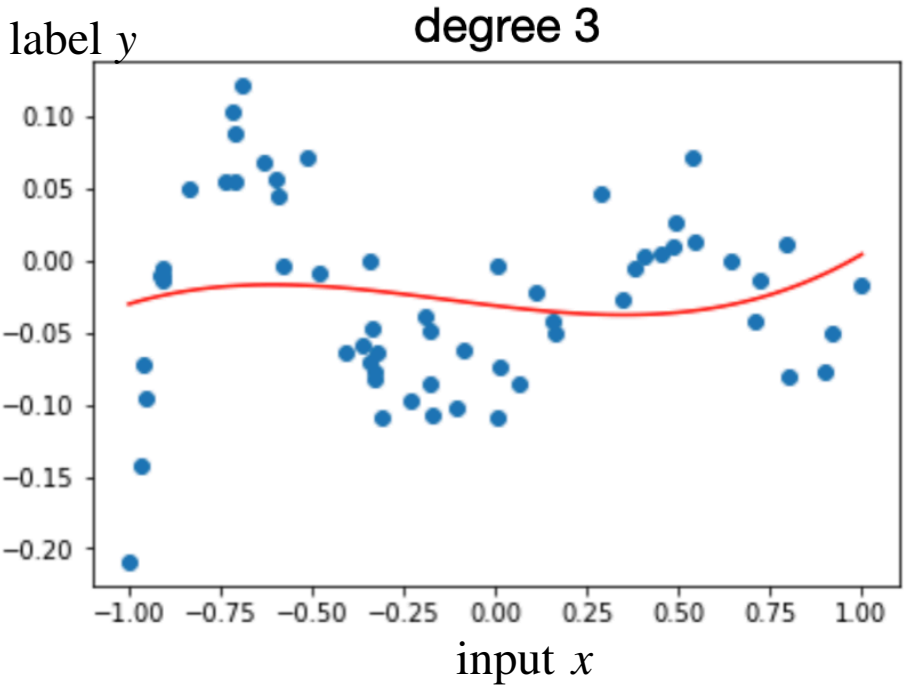


Underfitting: not fitting

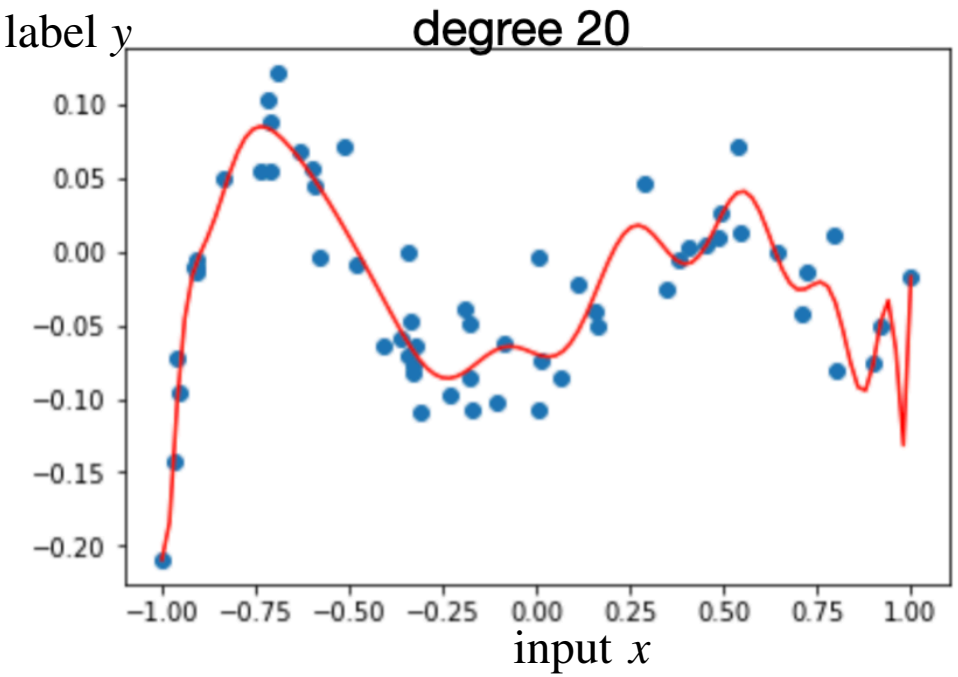
- How do we determine which is better model?

Which p should we choose?

- First instance of class of models with different representation power = model complexity



Underfitting: not fitting

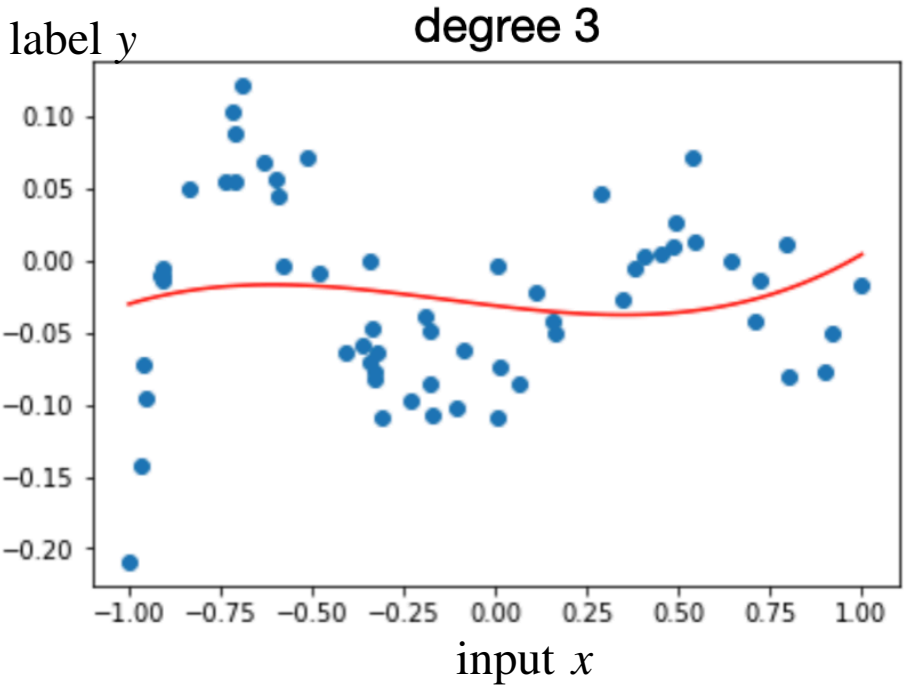


Overfitting: fitting “idiosyncrasies” of training data that won’t generalize

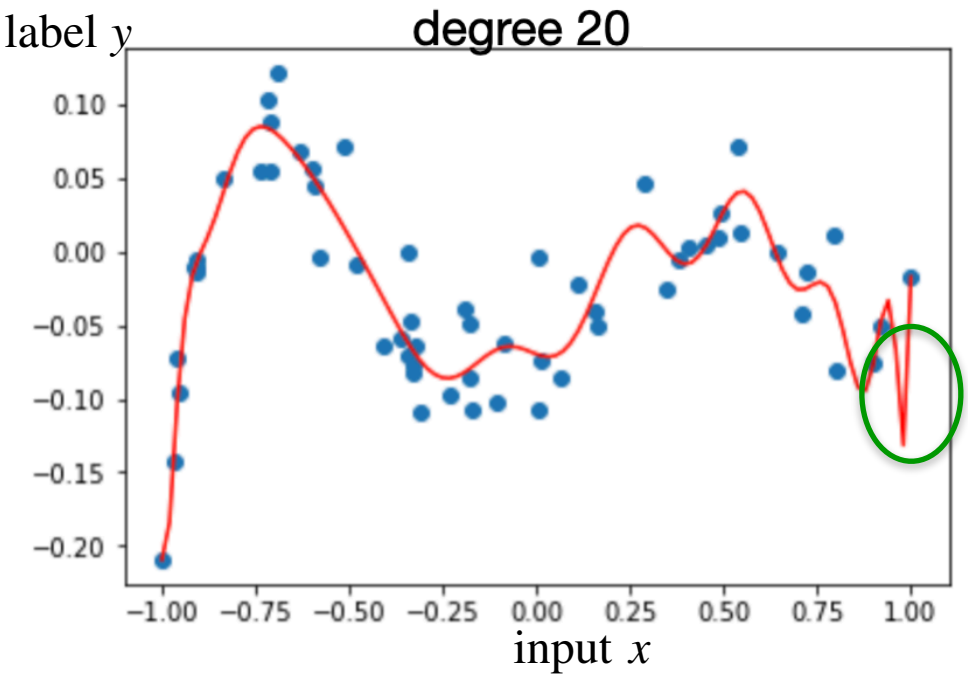
- How do we determine which is better model?

Which p should we choose?

- First instance of class of models with different representation power = model complexity



Underfitting: not fitting



Overfitting: fitting “idiosyncrasies” of training data that won’t generalize

- How do we determine which is better model?

Generalization

- we say a predictor **generalizes** if it performs as well on unseen data as on training data (we will formalize shortly)
- the data used to train a predictor is **training data** or **in-sample data**
- we want the predictor to work on **out-of-sample data**
- we say a predictor **fails to generalize** if it performs well on in-sample data but does not perform well on out-of-sample data

Generalization

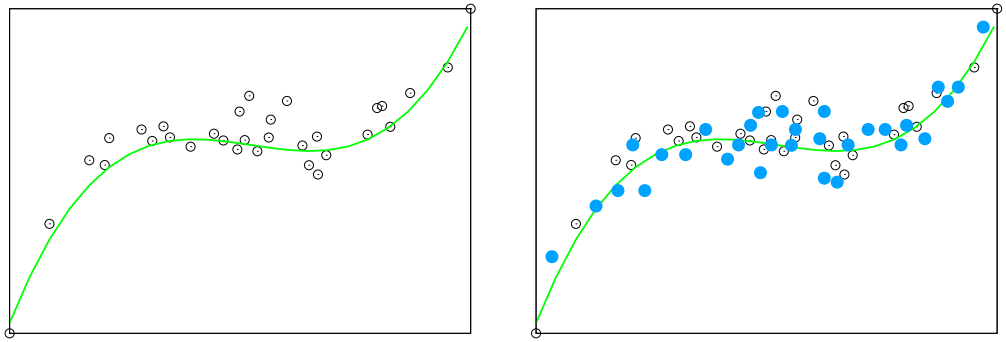
Test data

- we say a predictor **generalizes** if it performs as well on unseen data as on training data (we will formalize shortly)
- the data used to train a predictor is **training data** or **in-sample data**
- we want the predictor to work on **out-of-sample data**
- we say a predictor **fails to generalize** if it performs well on in-sample data but does not perform well on out-of-sample data

Generalization

Consider the following:

- **train** a cubic predictor on 32 (**in-sample**) white circles: Mean Squared Error (MSE) 174
- **predict** label y for 30 (**out-of-sample**) blue circles: MSE 192
- **Would you conclude this predictor generalizes effectively?**

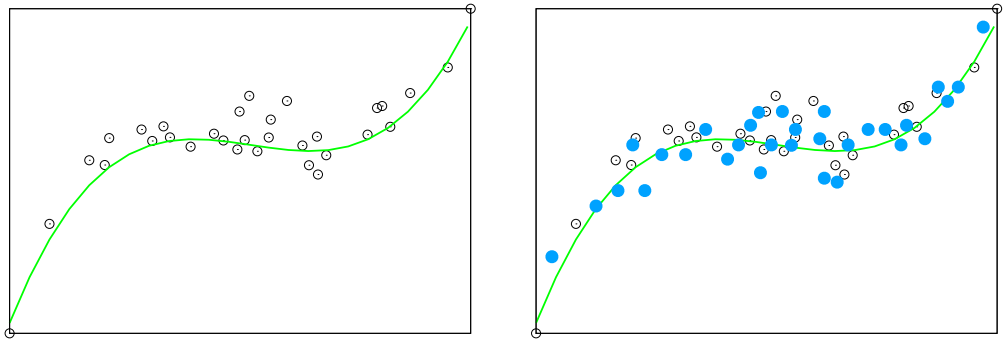


Train Test

Generalization

Consider the following:

- **train** a cubic predictor on 32 (**in-sample**) white circles: Mean Squared Error (MSE) 174
- **predict** label y for 30 (**out-of-sample**) blue circles: MSE 192
- **Would you conclude this predictor generalizes effectively?**
  - **Yes**, as in-sample MSE $\simeq$ out-of-sample MSE

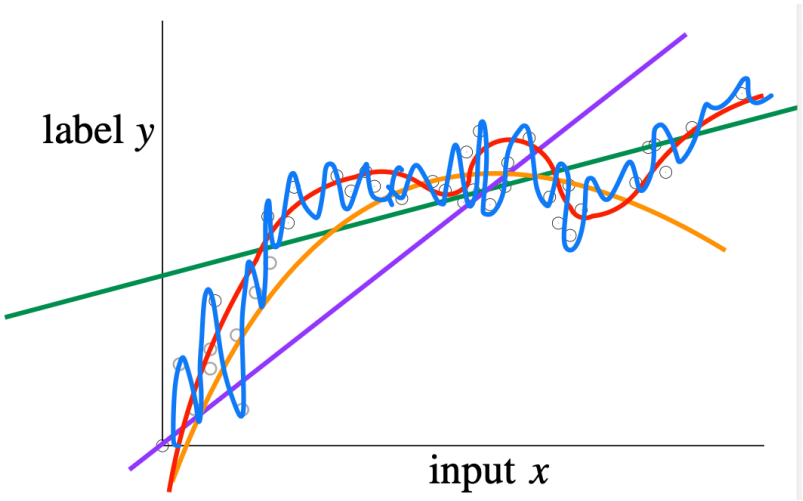


Train Test

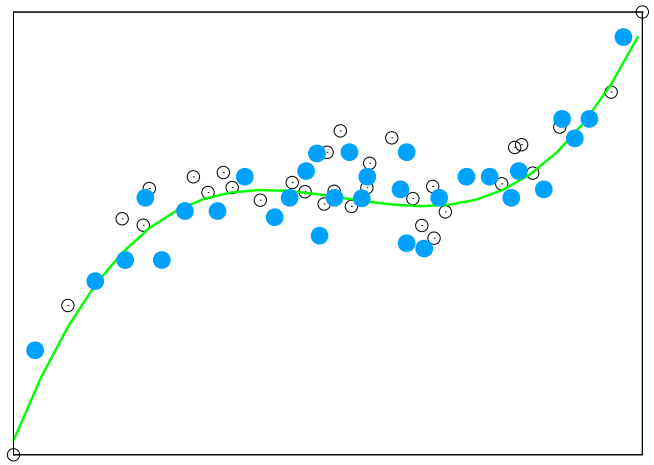
Generalization

Consider the following:

- **train** a cubic predictor on 32 (**in-sample**) white circles: Mean Squared Error (MSE) = 2
- **predict** label y for 30 (**out-of-sample**) blue circles: MSE = 356
- **Would you conclude this predictor generalizes effectively?**
    - **No**, as in-sample MSE << out-of-sample MSE



Train



Test

Split the data into training and testing

- a way to mimic how the predictor performs on unseen data
- given a single dataset $S = \{(x_i, y_i)\}_{i=1}^{n}$
- we split the dataset into two: training set and test set (e.g., 90/10)
- **training set** used to train the model
  - $$\text{minimize} \quad \mathscr{L}_{\text{train}}(w) = \frac{1}{|S_{\text{train}}|} \sum_{i \in S_{\text{train}}} (y_i - x_i^T w)^2$$

  # Fit the params of the model
- **test set** used to evaluate the model
  - $$\mathscr{L}_{\text{test}}(w) = \frac{1}{|S_{\text{test}}|} \sum_{i \in S_{\text{test}}} (y_i - x_i^T w)^2$$
- this assumes that test set is similar to unseen data
- **test set should never be used in training or picking unknowns!!**

Split the data into training and testing

- a way to mimic how the predictor performs on unseen data
- given a single dataset $S = \{(x_i, y_i)\}_{i=1}^{n}$
- we split the dataset into two: training set and test set (e.g., 90/10)
- **training set** used to train the model
  - $$\text{minimize} \quad \mathscr{L}_{\text{train}}(w) = \frac{1}{|S_{\text{train}}|} \sum_{i \in S_{\text{train}}} (y_i - x_i^T w)^2$$

  # Fit the params of the model
- **test set** used to evaluate the model
  - $$\mathscr{L}_{\text{test}}(w) = \frac{1}{|S_{\text{test}}|} \sum_{i \in S_{\text{test}}} (y_i - x_i^T w)^2$$
- this assumes that test set is similar to unseen data
- **test set should never be used in training or picking unknowns!!**

Split the data into training and testing

- a way to mimic how the predictor performs on unseen data
- given a single dataset $S = \{(x_i, y_i)\}_{i=1}^{n}$
- we split the dataset into two: training set and test set (e.g., 90/10)
- **training set** used to train the model
  - $$\text{minimize} \quad \mathscr{L}_{\text{train}}(w) = \frac{1}{|S_{\text{train}}|} \sum_{i \in S_{\text{train}}} (y_i - x_i^T w)^2$$

  # Fit the params of the model
- **test set** used to evaluate the model
  - $$\mathscr{L}_{\text{test}}(w) = \frac{1}{|S_{\text{test}}|} \sum_{i \in S_{\text{test}}} (y_i - x_i^T w)^2$$
- this assumes that test set is similar to unseen data
- **test set should never be used in training or picking unknowns!!**



Split the data into training and testing

- a way to mimic how the predictor performs on unseen data
- given a single dataset $S = \{(x_i, y_i)\}_{i=1}^{n}$
- we split the dataset into two: training set and test set (e.g., 90/10)
- **training set** used to train the model
  - $$\text{minimize} \quad \mathscr{L}_{\text{train}}(w) = \frac{1}{|S_{\text{train}}|} \sum_{i \in S_{\text{train}}} (y_i - x_i^T w)^2$$

Fit the params of the model

- **test set** used to evaluate the model
  - $$\mathscr{L}_{\text{test}}(w) = \frac{1}{|S_{\text{test}}|} \sum_{i \in S_{\text{test}}} (y_i - x_i^T w)^2$$
- this assumes that test set is similar to unseen data
- **test set should never be used in training or picking unknowns!!**



This is cheating and will lead you to the wrong conclusions. Big statistical no no

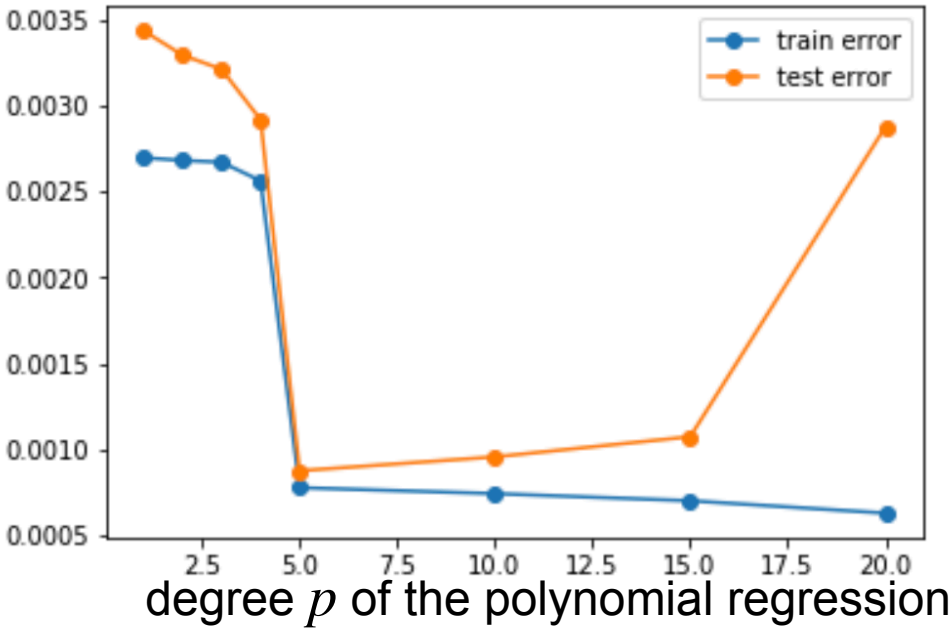
Demo: Polynomial Regression

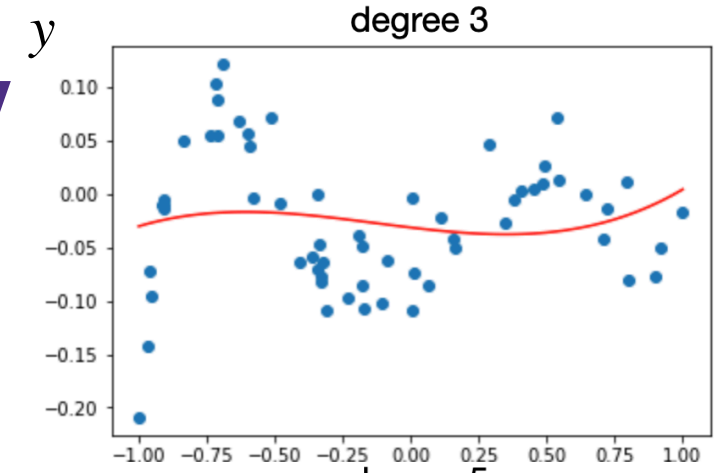
See colab notebook demo

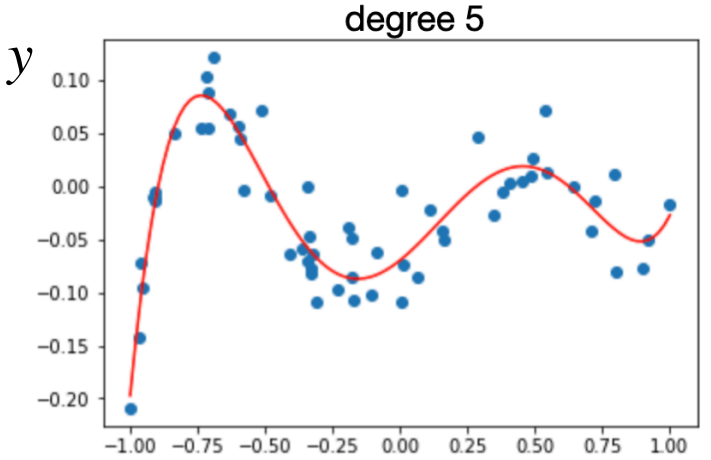


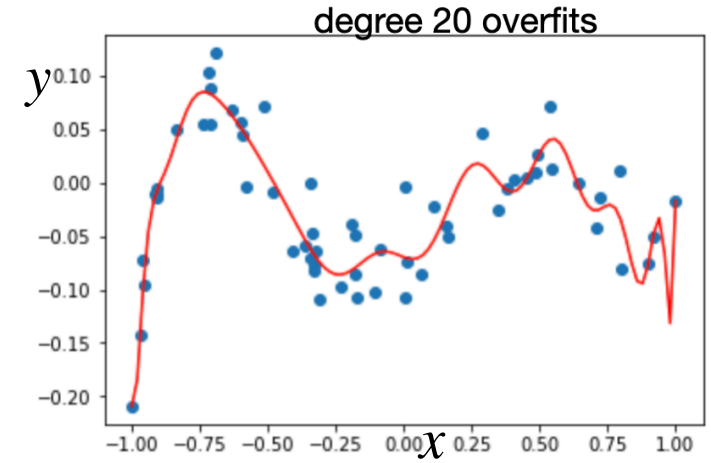
Train/test error vs. complexity

Error (hypothetical graph)









- Degree $p = 5$, if it achieves **minimum validation error**
- **Train error** monotonically decreases with model complexity
- **Test error** has a U shape. There is a best value

test set should never be used in training or picking degree

Cross-Validation



How… How… How???????

- How do we pick the number of basis functions…

- We could use the test data, but…

How… How… How???????

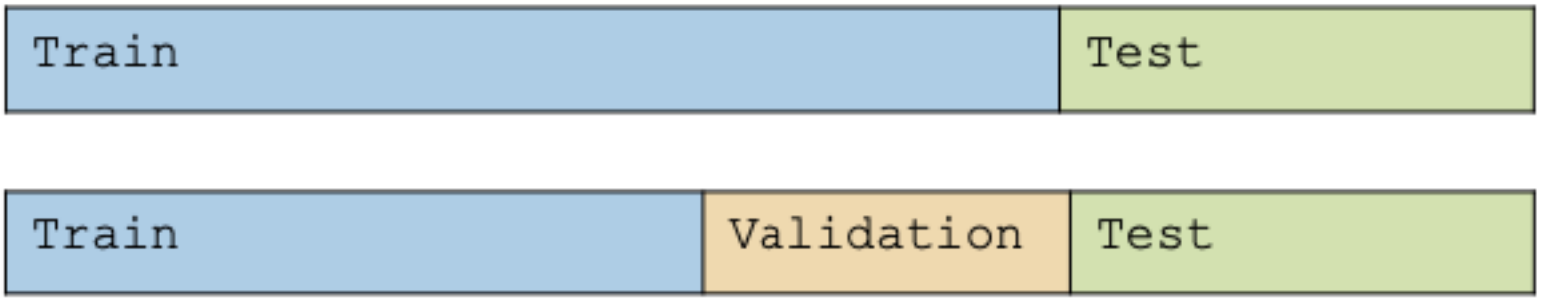
- How do we pick the number of basis functions…

- We could use the test data, but…

  - Never ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever train, or even look at the performance on the test data if you're making any decisions based on that performance.

Validation Set

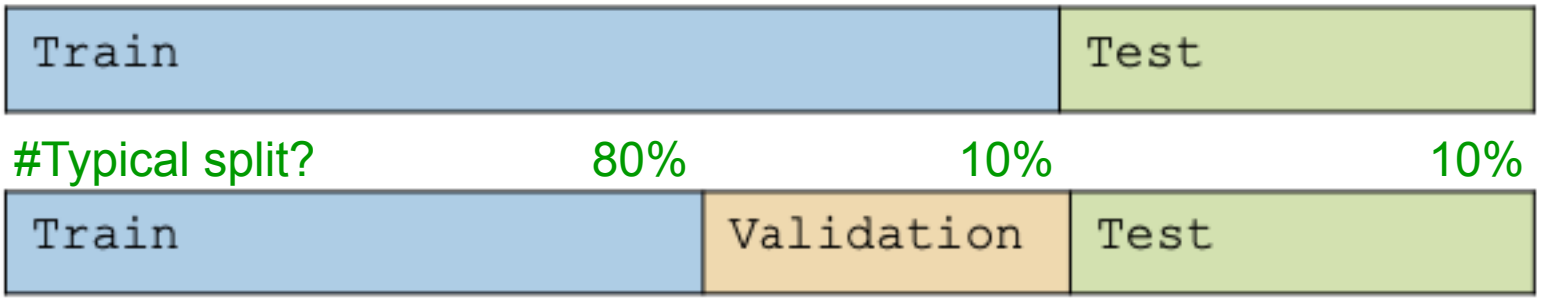
- Simplest approach is to add another dataset partition called the **validation set**



- Each set has its own role
  - **Train:** Used to train a model of each model complexity
  - **Validation:** Used as a hold-out to evaluate each model. Generally choose model complexity with lowest validation error.
  - **Test:** Once the model is chosen, can get an estimate of future error by test. This would be what you report. **Only do this once!**

Validation Set

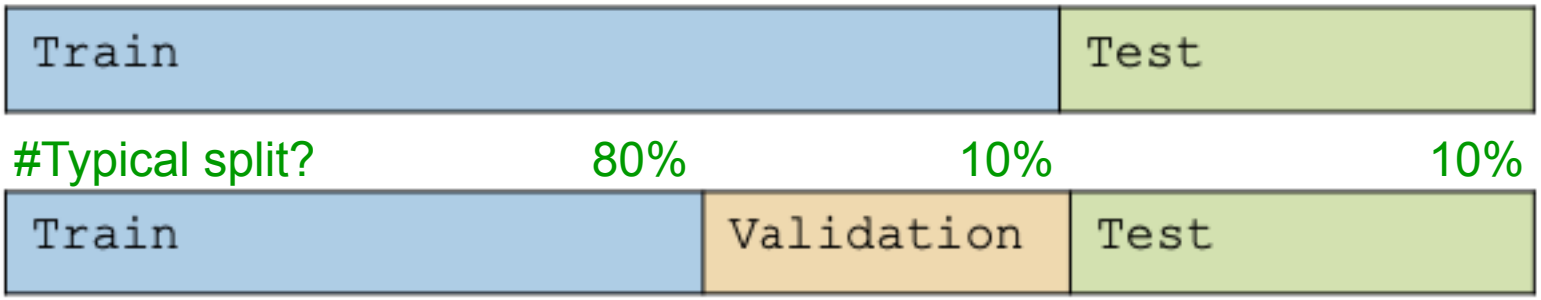
- Simplest approach is to add another dataset partition called the **validation set**



- Each set has its own role
  - **Train:** Used to train a model of each model complexity
  - **Validation:** Used as a hold-out to evaluate each model. Generally choose model complexity with lowest validation error.
  - **Test:** Once the model is chosen, can get an estimate of future error by test. This would be what you report. **Only do this once!**

Validation Set

- Simplest approach is to add another dataset partition called the **validation set**



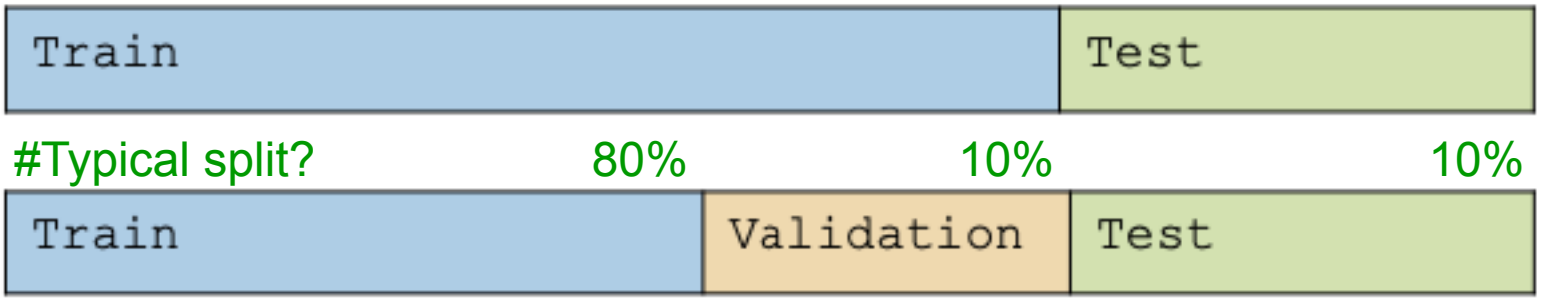
- Each set has its own role

Fit model params like w

  - **Train:** Used to train a model of each model complexity
  - **Validation:** Used as a hold-out to evaluate each model. Generally choose model complexity with lowest validation error.
  - **Test:** Once the model is chosen, can get an estimate of future error by test. This would be what you report. **Only do this once!**

Validation Set

- Simplest approach is to add another dataset partition called the **validation set**



- Each set has its own role

  # Fit model params like w

  - **Train:** Used to train a model of each model complexity
  - **Validation:** Used as a hold-out to evaluate each model. Generally choose model complexity with lowest validation error. # Choose *hyper*parameters, like polynomial degree p
  - **Test:** Once the model is chosen, can get an estimate of future error by test. This would be what you report. **Only do this once!**

(LOO) Leave-one-out cross validation

- **Consider a validation set with 1 example:**
  - **D – training data**
  - **D\j – training data with j th data point (x_j ,y_j) moved to validation set**
- **Learn model $f_{D\setminus j}$ with $D\setminus j$ dataset**
- **Estimate true error** as squared error on predicting $\mathbf{y_j}$:
  - **Unbiased estimate of $\text{error}_{\text{true}}(f_{D\setminus j})$!**

(LOO) Leave-one-out cross validation

- **Consider a validation set with 1 example:**
  - **D – training data** # n samples
  - **D\j – training data with j th data point (x_j ,y_j) moved to validation set**
- **Learn model $f_{D\setminus j}$ with $D\setminus j$ dataset**
- **Estimate true error** as squared error on predicting $\mathbf{y_j}$:
  - **Unbiased estimate of $\text{error}_{\text{true}}(f_{D\setminus j})$!**

(LOO) Leave-one-out cross validation

- **Consider a validation set with 1 example:**
  - **D – training data** # n samples
  - **D\j – training data with j th data point (x_j ,y_j) moved to validation set**
- **Learn model $f_{D\setminus j}$ with $D\setminus j$ dataset** # n-1 samples
- **Estimate true error** as squared error on predicting $\mathbf{y_j}$:
  - **Unbiased estimate of $\text{error}_{\text{true}}(f_{D\setminus j})$!**

(LOO) Leave-one-out cross validation

- **Consider a validation set with 1 example:**
  - **D – training data**
  - **D\j – training data with j th data point (x_j ,y_j) moved to validation set**
- **Learn model $f_{D\setminus j}$ with $D\setminus j$ dataset**
- **Estimate true error** as squared error on predicting $\mathbf{y_j}$:
  - **Unbiased estimate of $\text{error}_{true}(f_{D\setminus j})$!**

- **LOO cross validation**: Average over all data points *j*:
  - **For each data point you leave out, learn a new model $f_{D\setminus j}$**
  - **Estimate error** as:

$$\text{error}_{LOO} = \frac{1}{n}\sum_{j=1}^{n}(y_j - f_{\mathcal{D}\setminus j}(x_j))^2$$

LOO cross validation is (almost) unbiased estimate!

- **When computing LOOCV error, we only use *N-1* data points**
  - **So it's not quite an estimate of true error of learning with *N* data points**
  - **Usually pessimistic, though – learning with less data typically gives worse answer**

- **LOO is almost unbiased**! Why not **use LOO error for model selection!**
  - **E.g., picking degree**



LOO cross validation is (almost) unbiased estimate!

- **When computing LOOCV error, we only use *N-1* data points**
  - **So it's not quite an estimate of true error of learning with *N* data points**
  - **Usually pessimistic, though – learning with less data typically gives worse answer**

- **LOO is almost unbiased**! Why not **use LOO error for model selection!**
  - **E.g., picking degree**



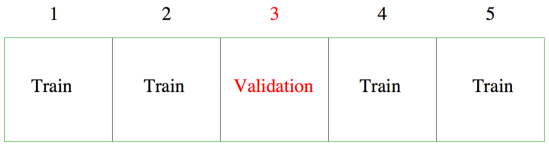
You just fit n models!!

Computational cost of LOO

- Suppose you have 100,000 data points
- You implemented a great version of your learning algorithm
  - Learns in only 1 second
- Computing LOO will take about 1 day!!!
  -

Use *k*-fold cross validation

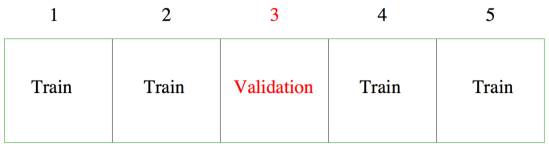
- Randomly divide training data into *k* equal parts
  - $D_1,\ldots,D_k$
- For each *i*
  - Learn model $f_{D\backslash Di}$ using data point not in D_i
  - Estimate error of $f_{D\backslash Di}$ on validation set D_i:



$$\text{error}_{\mathcal{D}_i} = \frac{1}{|\mathcal{D}_i|} \sum_{(x_j,y_j)\in\mathcal{D}_i} (y_j - f_{\mathcal{D}\backslash\mathcal{D}_i}(x_j))^2$$

Use *k*-fold cross validation

- Randomly divide training data into *k* equal parts
  - $D_1,\ldots,D_k$
- For each *i*
  - Learn model $f_{D\setminus Di}$ using data point not in D_i
  - Estimate error of $f_{D\setminus Di}$ on validation set D_i:



$$\text{error}_{\mathcal{D}_i} = \frac{1}{|\mathcal{D}_i|} \sum_{(x_j,y_j)\in\mathcal{D}_i} (y_j - f_{\mathcal{D}\setminus\mathcal{D}_i}(x_j))^2$$

- k-fold cross validation error is average over data splits:

$$error_{k-fold} = \frac{1}{k}\sum_{i=1}^{k} error_{\mathcal{D}_i}$$

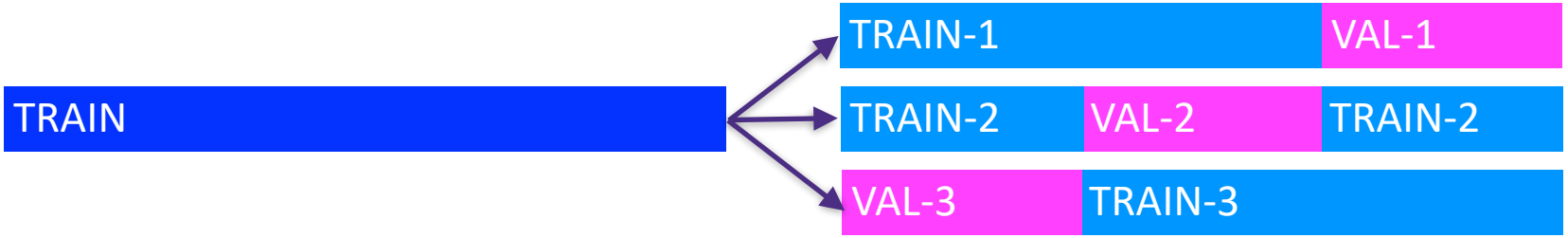
- k-fold cross validation properties:
  - Much faster to compute than LOO
  - More (pessimistically) biased – using much less data, only *n(k-1)/k*
  - Usually, k = 10

Recap

- Given a dataset, begin by splitting into



- Model selection: Use k-fold cross-validation on **TRAIN** to train predictor and choose magic parameters such as degree



- Model assessment: Use **TEST** to assess the accuracy of the model you output
  - **Never ever ever ever ever train or choose parameters based on the test data**

Example 1

- **You wish to predict the stock price of zoom.us given historical stock price data**
- **You use all daily stock price up to Jan 1, 2020 as TRAIN and Jan 2, 2020 - April 13, 2020 as TEST**
- **What's "wrong" with this procedure?**

Example 1

- **You wish to predict the stock price of zoom.us given historical stock price data**
- **You use all daily stock price up to Jan 1, 2020 as TRAIN and Jan 2, 2020 - April 13, 2020 as TEST**
- **What's "wrong" with this procedure?**

  - ML assumes data is i.i.d.; is this?

Example 1

- **You wish to predict the stock price of zoom.us given historical stock price data**
- **You use all daily stock price up to Jan 1, 2020 as TRAIN and Jan 2, 2020 - April 13, 2020 as TEST**
- **What's "wrong" with this procedure?**

  - ML assumes data is i.i.d.; is this?
    - Nope! Future data is o.o.d. (out of distribution) from the training data

Example 1

- **You wish to predict the stock price of zoom.us given historical stock price data**
- **You use all daily stock price up to Jan 1, 2020 as TRAIN and Jan 2, 2020 - April 13, 2020 as TEST**
- **What's "wrong" with this procedure?**

  - ML assumes data is i.i.d.; is this?
    - Nope! Future data is o.o.d. (out of distribution) from the training data

  - Why is "wrong" in square quotes?

Example 1

- **You wish to predict the stock price of zoom.us given historical stock price data**
- **You use all daily stock price up to Jan 1, 2020 as TRAIN and Jan 2, 2020 - April 13, 2020 as TEST**
- **What's "wrong" with this procedure?**

  - ML assumes data is i.i.d.; is this?
    - Nope! Future data is o.o.d. (out of distribution) from the training data

  - Why is "wrong" in square quotes?
    - You might still want to do this. It's economically valuable, if you could.

Example 2

- **Given 10,000-dimensional data and n examples, we pick a subset of 50 dimensions that have the highest correlation with labels in the entire dataset:**

50 indices j that have largest $\frac{|\sum_{i=1}^{n} x_{i,j} y_i|}{\sqrt{\sum_{i=1}^{n} x_{i,j}^2}}$

- **After picking our 50 features, we then break data into train and test dataset.**
- **We train linear regression on these selected features on the training set. We compute the test error and report it**
- **What's wrong with this procedure?**

Example 2

- **Given 10,000-dimensional data and n examples, we pick a subset of 50 dimensions that have the highest correlation with labels in the entire dataset:**

50 indices j that have largest $\frac{|\sum_{i=1}^{n} x_{i,j} y_i|}{\sqrt{\sum_{i=1}^{n} x_{i,j}^2}}$

- **After picking our 50 features, we then break data into train and test dataset.**
- **We train linear regression on these selected features on the training set. We compute the test error and report it**
- **What's wrong with this procedure?**
  - Need to hold out the test data first, otherwise you're contaminating your test data.

Example 2

- **Given 10,000-dimensional data and n examples, we pick a subset of 50 dimensions that have the highest correlation with labels in the entire dataset:**

50 indices j that have largest $\frac{|\sum_{i=1}^{n} x_{i,j} y_i|}{\sqrt{\sum_{i=1}^{n} x_{i,j}^2}}$

- **After picking our 50 features, we then break data into train and test dataset.**
- **We train linear regression on these selected features on the training set. We compute the test error and report it**
- **What's wrong with this procedure?**
  - Need to hold out the test data first, otherwise you're contaminating your test data.
    - This happened in my first ML project. Thought the accuracy was 90%. When I fixed, it went down to 60%.

Example 2

- **Given 10,000-dimensional data and n examples, we pick a subset of 50 dimensions that have the highest correlation with labels in the entire dataset:**

50 indices j that have largest $$\frac{|\sum_{i=1}^{n} x_{i,j} y_i|}{\sqrt{\sum_{i=1}^{n} x_{i,j}^2}}$$

- **After picking our 50 features, we then break data into train and test dataset.**
- **We train linear regression on these selected features on the training set. We compute the test error and report it**
- **What's wrong with this procedure?**
  - Need to hold out the test data first, otherwise you're contaminating your test data.
    - This happened in my first ML project. Thought the accuracy was 90%. When I fixed, it went down to 60%.
  - Top 50 features might be redundant with each other.

Recap

- Learning is…
  - Collect some data
    - E.g., potato delivery features and delivery time
  - Randomly split dataset into TRAIN, VAL, and TEST
    - E.g., 80%, 10%, and 10%, respectively
  - Choose a hypothesis class or model
    - E.g., linear with non-linear transformations
  - Choose a loss function
    - E.g., least squares on TRAIN
  - Choose an optimization procedure
    - E.g., set derivative to zero to obtain estimator, cross-validation on VAL to pick num. features
- Justifying the accuracy of the estimate
    - E.g., report TEST error

Bias-Variance Tradeoff



Optimal Prediction

Goal: Predict $Y \in \mathbb{R}$ **given** $X \in \mathbb{R}^d$ **if** $(X, Y) \sim P_{XY}$

Find function η that minimizes

$$\mathbb{E}_{XY}[(Y - \eta(X))^2] = \mathbb{E}_X\left[\mathbb{E}_{Y|X}[(Y - \eta(x))^2 | X = x]\right]$$

(Hint: for any x, $\eta(x) = c_x$ where c_x minimizes $\mathbb{E}_{Y|X}[(Y - c_x)^2 | X = x]$)

Optimal Prediction

True distribution, don't actually have access to it

Goal: Predict $Y \in \mathbb{R}$ **given** $X \in \mathbb{R}^d$ **if** $(X, Y) \sim P_{XY}$

Find function η that minimizes

$$\mathbb{E}_{XY}[(Y - \eta(X))^2] = \mathbb{E}_X\left[\mathbb{E}_{Y|X}[(Y - \eta(x))^2 | X = x]\right]$$

(Hint: for any x, $\eta(x) = c_x$ where c_x minimizes $\mathbb{E}_{Y|X}[(Y - c_x)^2 | X = x]$)

Optimal Prediction

True distribution, don't actually have access to it

Goal: Predict $Y \in \mathbb{R}$ **given** $X \in \mathbb{R}^d$ **if** $(X, Y) \sim P_{XY}$

Find function η that minimizes

Tower rule: E[Y] = E$_X$[E[Y|X]]

$$\mathbb{E}_{XY}[(Y - \eta(X))^2] = \mathbb{E}_X\left[\mathbb{E}_{Y|X}[(Y - \eta(x))^2 | X = x]\right]$$

(Hint: for any x, $\eta(x) = c_x$ where c_x minimizes $\mathbb{E}_{Y|X}[(Y - c_x)^2 | X = x]$)

Optimal Prediction

True distribution, don't actually have access to it

Goal: Predict $Y \in \mathbb{R}$ **given** $X \in \mathbb{R}^d$ **if** $(X, Y) \sim P_{XY}$

Find function η that minimizes

Tower rule: E[Y] = E$_X$[E[Y|X]]

$$\mathbb{E}_{XY}[(Y - \eta(X))^2] = \mathbb{E}_X\left[\mathbb{E}_{Y|X}[(Y - \eta(x))^2 | X = x]\right]$$

(Hint: for any x, $\eta(x) = c_x$ where c_x minimizes $\mathbb{E}_{Y|X}[(Y - c_x)^2 | X = x]$)

$$\frac{d}{dc}\mathbb{E}_{Y|X}[(Y - c)^2 \,|\, X = x] = 0$$

Optimal Prediction

True distribution, don't actually have access to it

Goal: Predict $Y \in \mathbb{R}$ **given** $X \in \mathbb{R}^d$ **if** $(X, Y) \sim P_{XY}$

Find function η that minimizes

Tower rule: E[Y] = E$_X$[E[Y|X]]

$$\mathbb{E}_{XY}[(Y - \eta(X))^2] = \mathbb{E}_X\left[\mathbb{E}_{Y|X}[(Y - \eta(x))^2 | X = x]\right]$$

(Hint: for any x, $\eta(x) = c_x$ where c_x minimizes $\mathbb{E}_{Y|X}[(Y - c_x)^2 | X = x]$)

$$\frac{d}{dc}\mathbb{E}_{Y|X}[(Y - c)^2 \,|\, X = x] = 0$$

$$= \mathbb{E}_{Y|X}[2(Y - c)(-1) \,|\, X = x] = 0$$

Optimal Prediction

True distribution, don't actually have access to it

Goal: Predict $Y \in \mathbb{R}$ **given** $X \in \mathbb{R}^d$ **if** $(X, Y) \sim P_{XY}$

Find function η that minimizes

Tower rule: E[Y] = E$_X$[E[Y|X]]

$$\mathbb{E}_{XY}[(Y - \eta(X))^2] = \mathbb{E}_X\left[\mathbb{E}_{Y|X}[(Y - \eta(x))^2 | X = x]\right]$$

(Hint: for any x, $\eta(x) = c_x$ where c_x minimizes $\mathbb{E}_{Y|X}[(Y - c_x)^2 | X = x]$)

$$\frac{d}{dc}\mathbb{E}_{Y|X}[(Y - c)^2 \,|\, X = x] = 0$$

$$= \mathbb{E}_{Y|X}[2(Y - c)(-1) \,|\, X = x] = 0$$

$$= -2[\mathbb{E}_{Y|X}[Y \,|\, X = x] - c] = 0$$

Optimal Prediction

True distribution, don't actually have access to it

Goal: Predict $Y \in \mathbb{R}$ **given** $X \in \mathbb{R}^d$ **if** $(X, Y) \sim P_{XY}$

Find function η that minimizes

Tower rule: E[Y] = E$_X$[E[Y|X]]

$$\mathbb{E}_{XY}[(Y - \eta(X))^2] = \mathbb{E}_X\left[\mathbb{E}_{Y|X}[(Y - \eta(x))^2 | X = x]\right]$$

(Hint: for any x, $\eta(x) = c_x$ where c_x minimizes $\mathbb{E}_{Y|X}[(Y - c_x)^2 | X = x]$)

$$\frac{d}{dc}\mathbb{E}_{Y|X}[(Y - c)^2 \,|\, X = x] = 0$$

$$= \mathbb{E}_{Y|X}[2(Y - c)(-1) \,|\, X = x] = 0$$

$$= -2[\mathbb{E}_{Y|X}[Y \,|\, X = x] - c] = 0$$

$$c_x = \mathbb{E}_{Y|X}[Y \,|\, X = x] = \eta(x)$$

Optimal Prediction

True distribution, don't actually have access to it

Goal: Predict $Y \in \mathbb{R}$ **given** $X \in \mathbb{R}^d$ **if** $(X, Y) \sim P_{XY}$

Find function η that minimizes

Tower rule: E[Y] = E$_X$[E[Y|X]]

$$\mathbb{E}_{XY}[(Y - \eta(X))^2] = \mathbb{E}_X\left[\mathbb{E}_{Y|X}[(Y - \eta(x))^2 | X = x]\right]$$

(Hint: for any x, $\eta(x) = c_x$ where c_x minimizes $\mathbb{E}_{Y|X}[(Y - c_x)^2 | X = x]$)

$$\frac{d}{dc}\mathbb{E}_{Y|X}[(Y - c)^2 \,|\, X = x] = 0$$

$$= \mathbb{E}_{Y|X}[2(Y - c)(-1) \,|\, X = x] = 0$$

$$= -2[\mathbb{E}_{Y|X}[Y \,|\, X = x] - c] = 0$$

$$c_x = \boxed{\mathbb{E}_{Y|X}[Y \,|\, X = x] = \eta(x)}$$

Optimal Prediction

Goal: Predict $Y \in \mathbb{R}$ given $X \in \mathbb{R}^d$ if $(X, Y) \sim P_{XY}$

Find function η that minimizes

$$\mathbb{E}_{XY}[(Y - \eta(X))^2] = \mathbb{E}_X\left[\mathbb{E}_{Y|X}[(Y - \eta(x))^2 | X = x]\right]$$

(Hint: for any x, $\eta(x) = c_x$ where c_x minimizes $\mathbb{E}_{Y|X}[(Y - c_x)^2 | X = x]$)

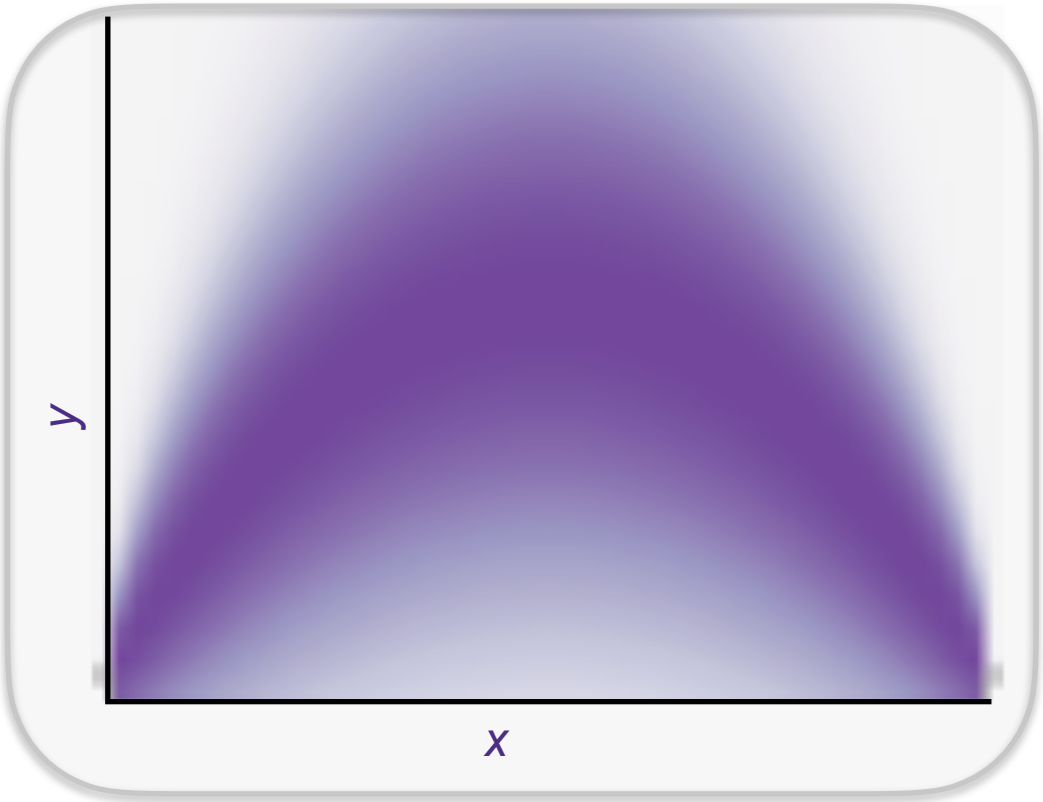
$$\begin{aligned} 0 &= \frac{d}{dc_x}\mathbb{E}_{Y|X}[(Y - c_x)^2 | X = x] \\ &= \mathbb{E}_{Y|X}[\frac{d}{dc_x}(Y - c_x)^2 | X = x] \\ &= \mathbb{E}_{Y|X}[-2(Y - c_x) | X = x] = -2\mathbb{E}_{Y|X}[Y | X = x] + 2c_x \end{aligned}$$

Squared Error Optimal Predictor: $\eta(x) = \mathbb{E}_{Y|X}[Y | X = x]$

Statistical Learning

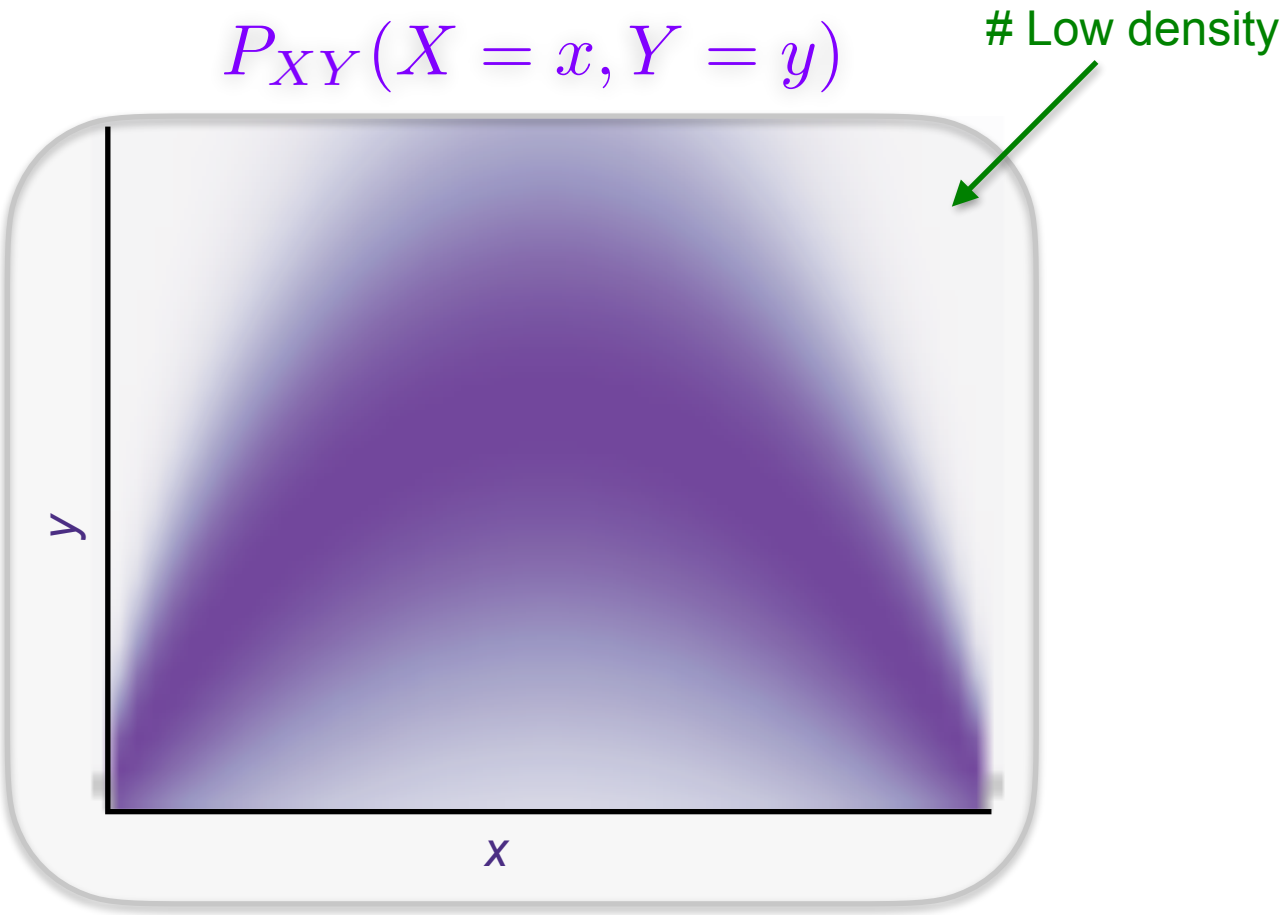
$$\mathbb{E}_{XY}[(Y - \eta(X))^2]$$

$$P_{XY}(X = x, Y = y)$$



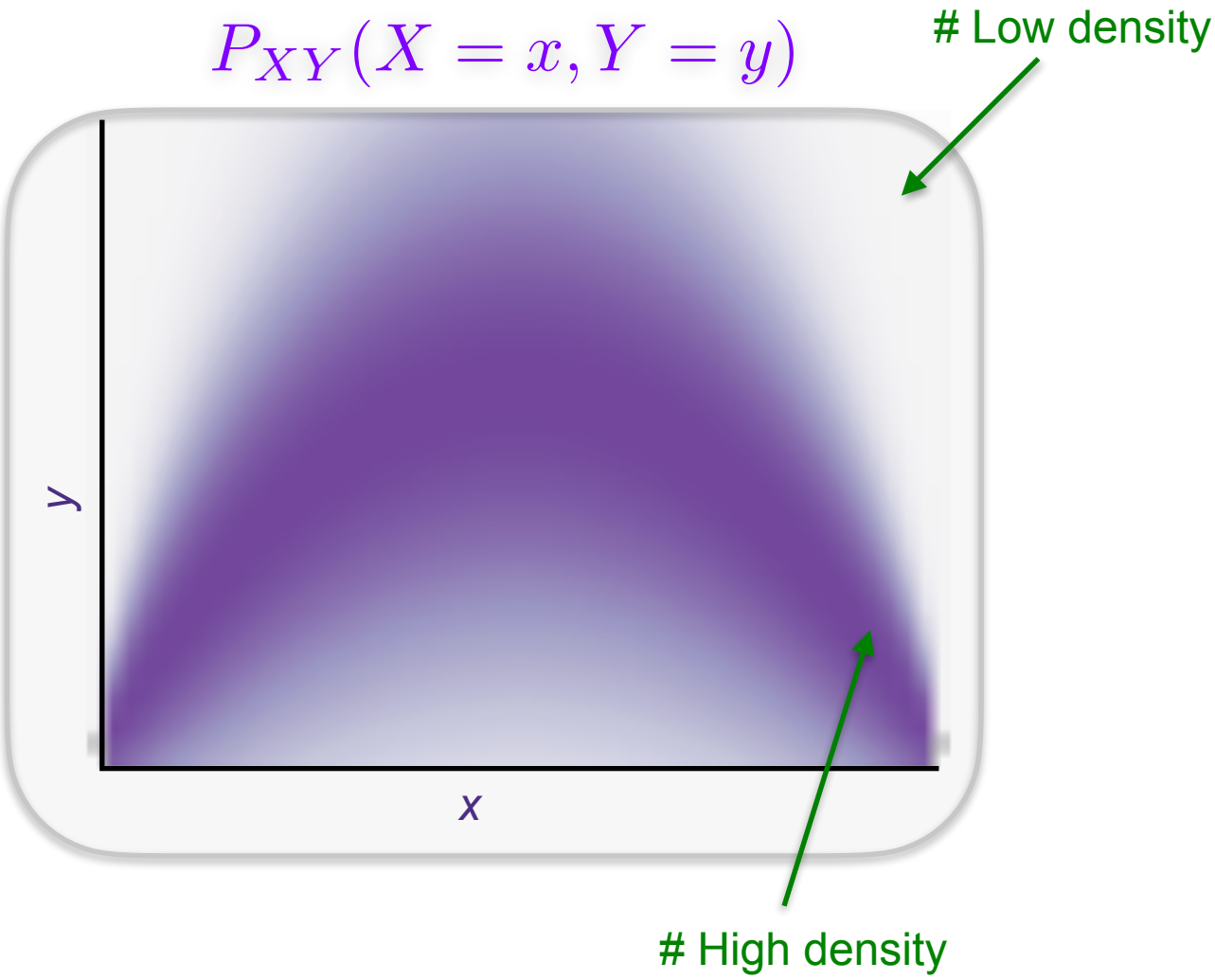
Statistical Learning

$$\mathbb{E}_{XY}[(Y - \eta(X))^2]$$



Statistical Learning

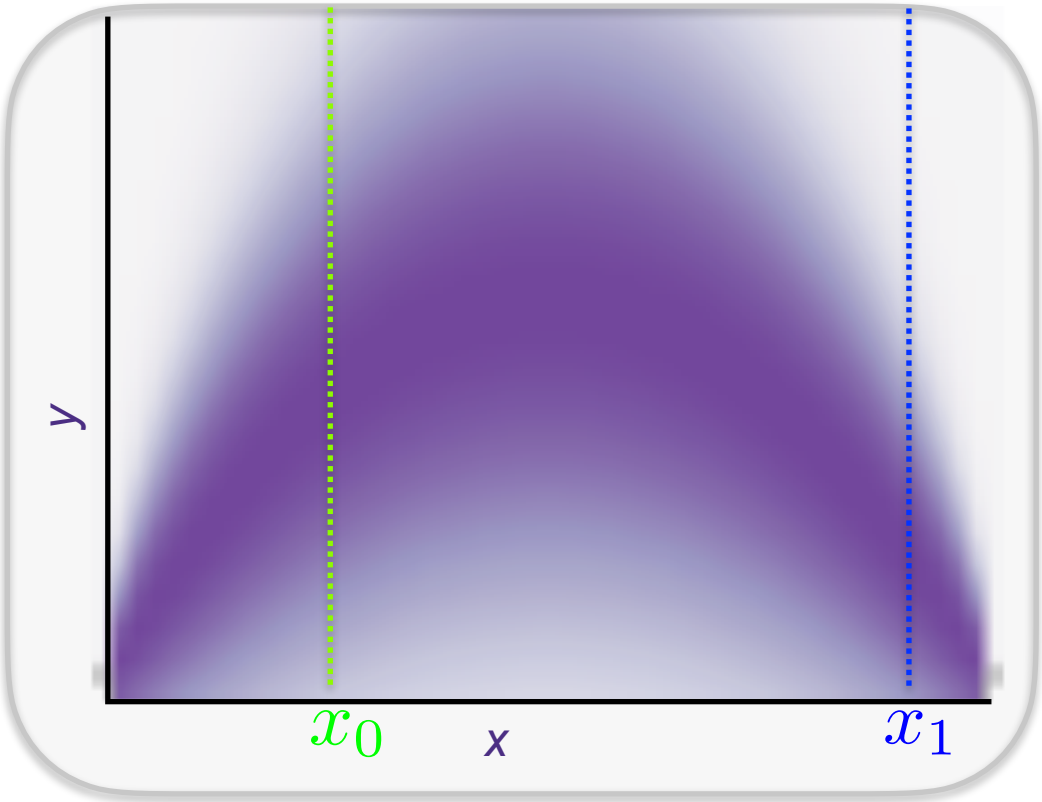
$$\mathbb{E}_{XY}[(Y - \eta(X))^2]$$



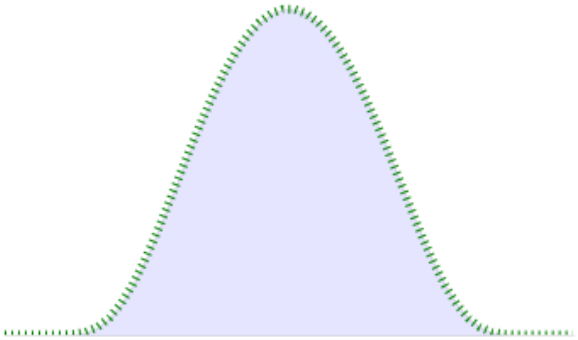
Statistical Learning

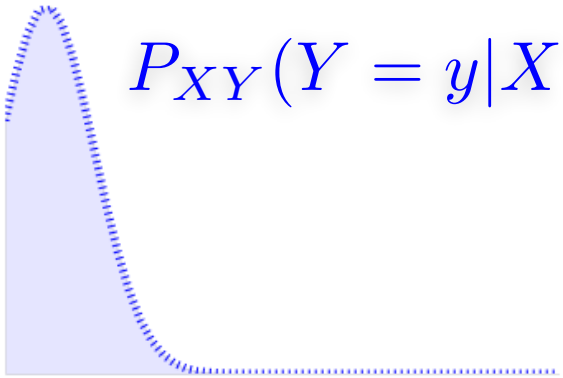
$$\mathbb{E}_{XY}[(Y - \eta(X))^2]$$

$P_{XY}(X = x, Y = y)$



$P_{XY}(Y = y|X = x_0)$

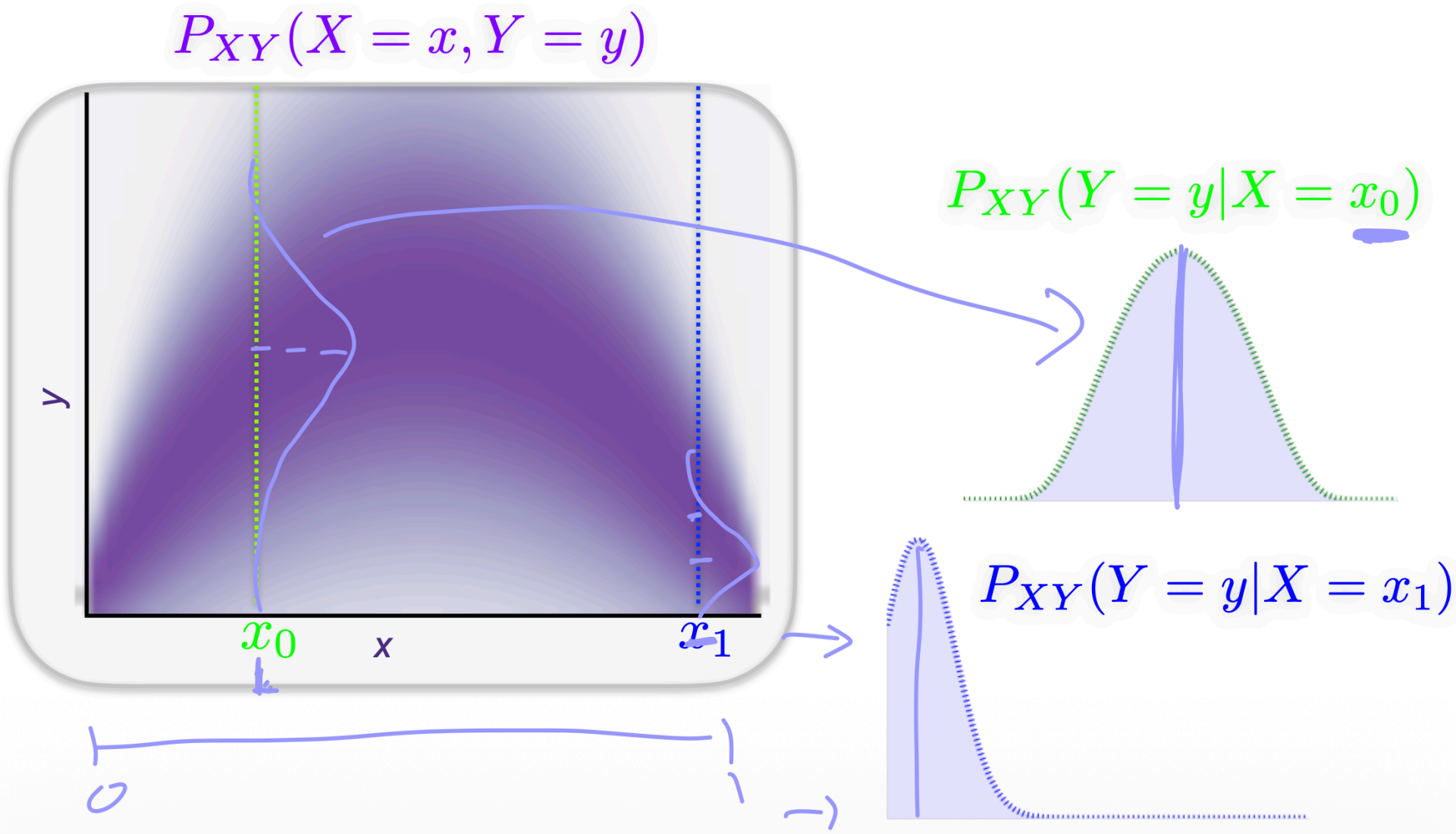




$P_{XY}(Y = y|X = x_1)$

Statistical Learning

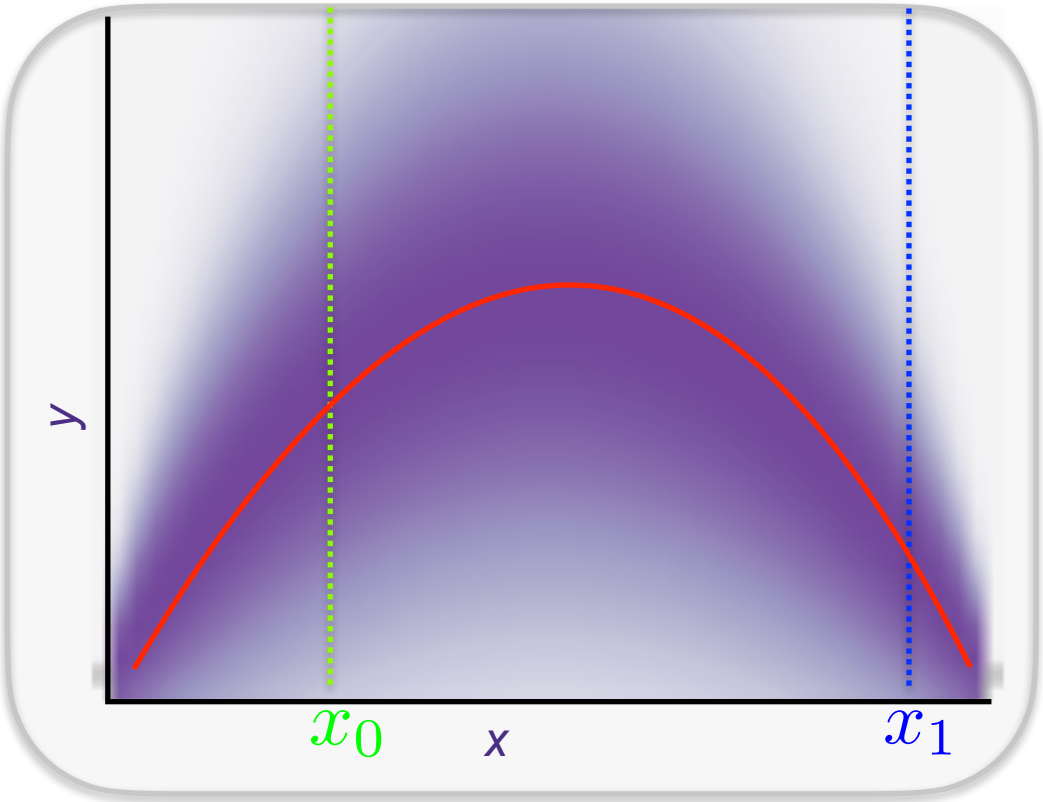
$$\mathbb{E}_{XY}[(Y - \eta(X))^2]$$



Statistical Learning

$$\mathbb{E}_{XY}[(Y - \eta(X))^2]$$

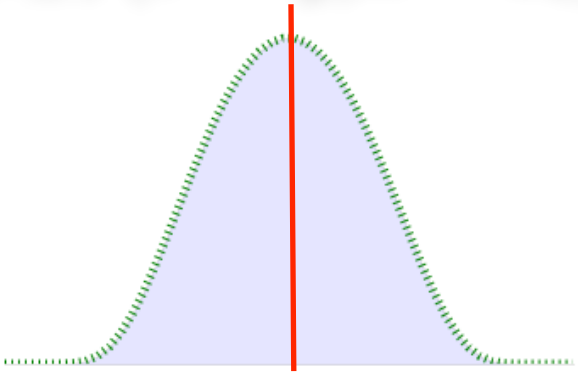
$P_{XY}(X = x, Y = y)$

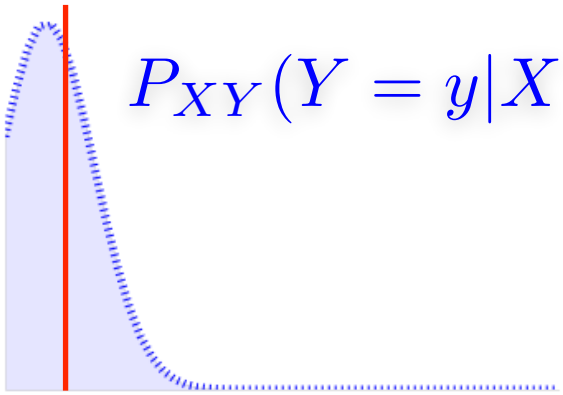


Ideally, we want to find:

$$\eta(x) = \mathbb{E}_{Y|X}[Y|X = x]$$

$P_{XY}(Y = y|X = x_0)$

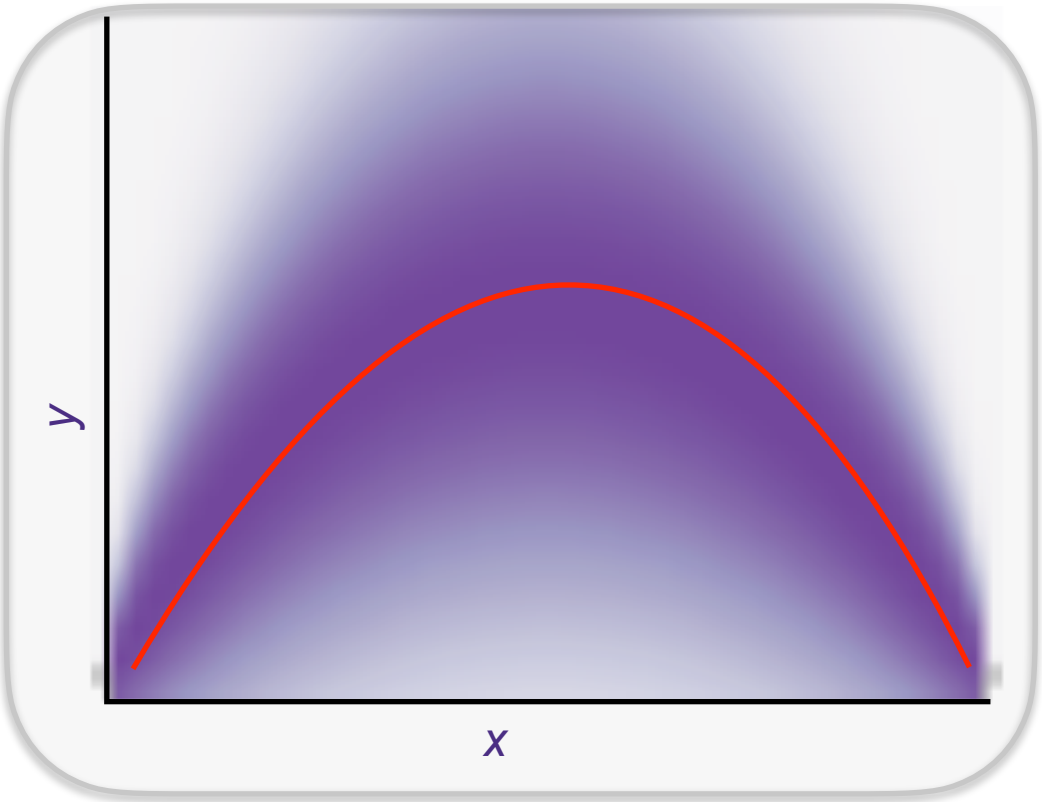




$P_{XY}(Y = y|X = x_1)$

Statistical Learning

$P_{XY}(X = x, Y = y)$

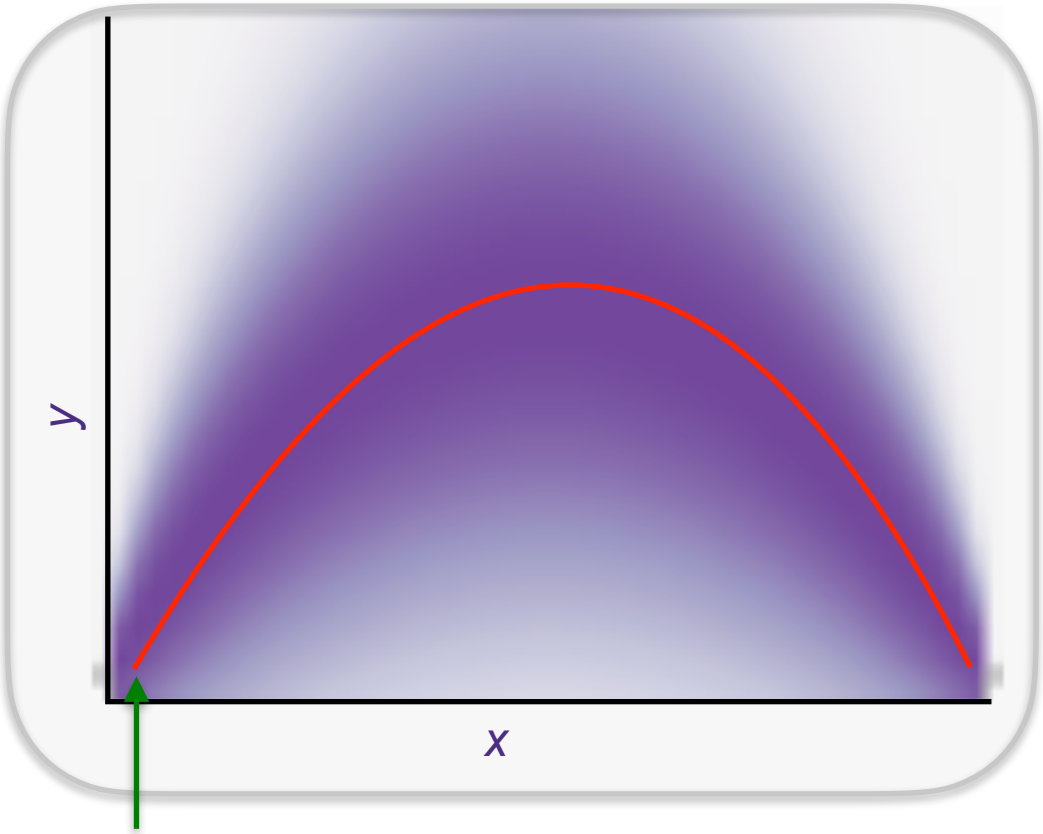


Ideally, we want to find:

$$\eta(x) = \mathbb{E}_{Y|X}[Y|X = x]$$

Statistical Learning

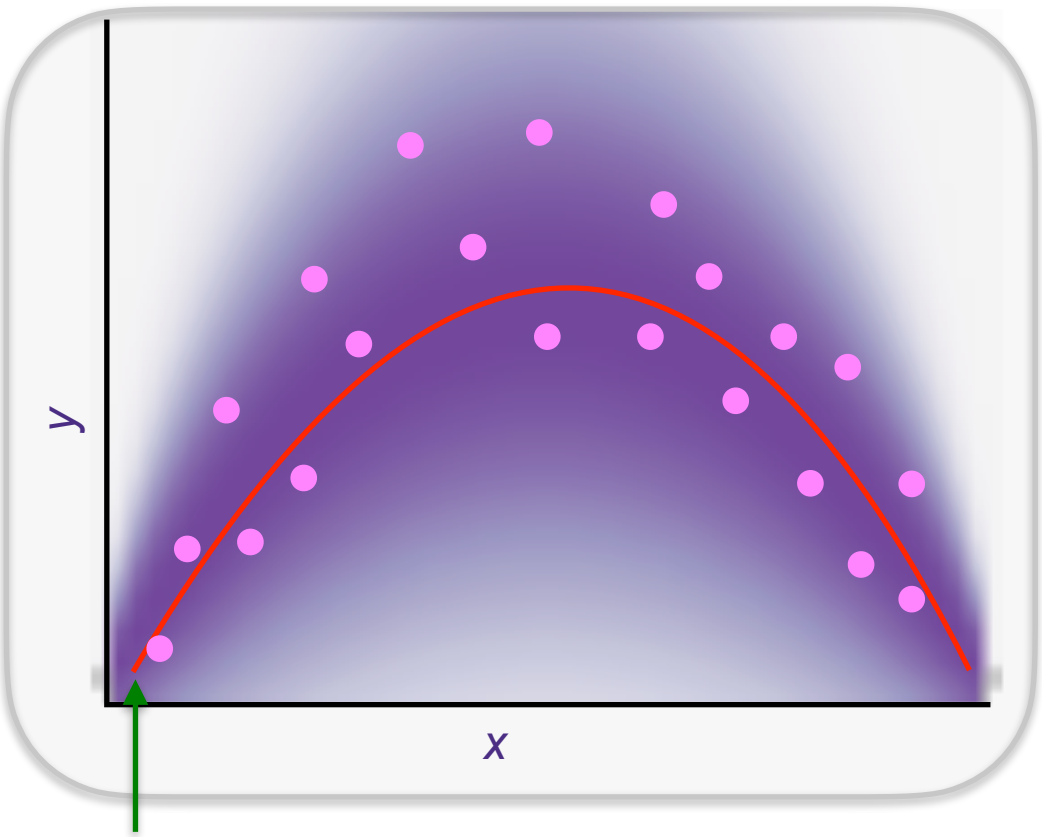
$P_{XY}(X = x, Y = y)$



Ideally, we want to find:

$$\eta(x) = \mathbb{E}_{Y|X}[Y|X = x]$$

Statistical Learning



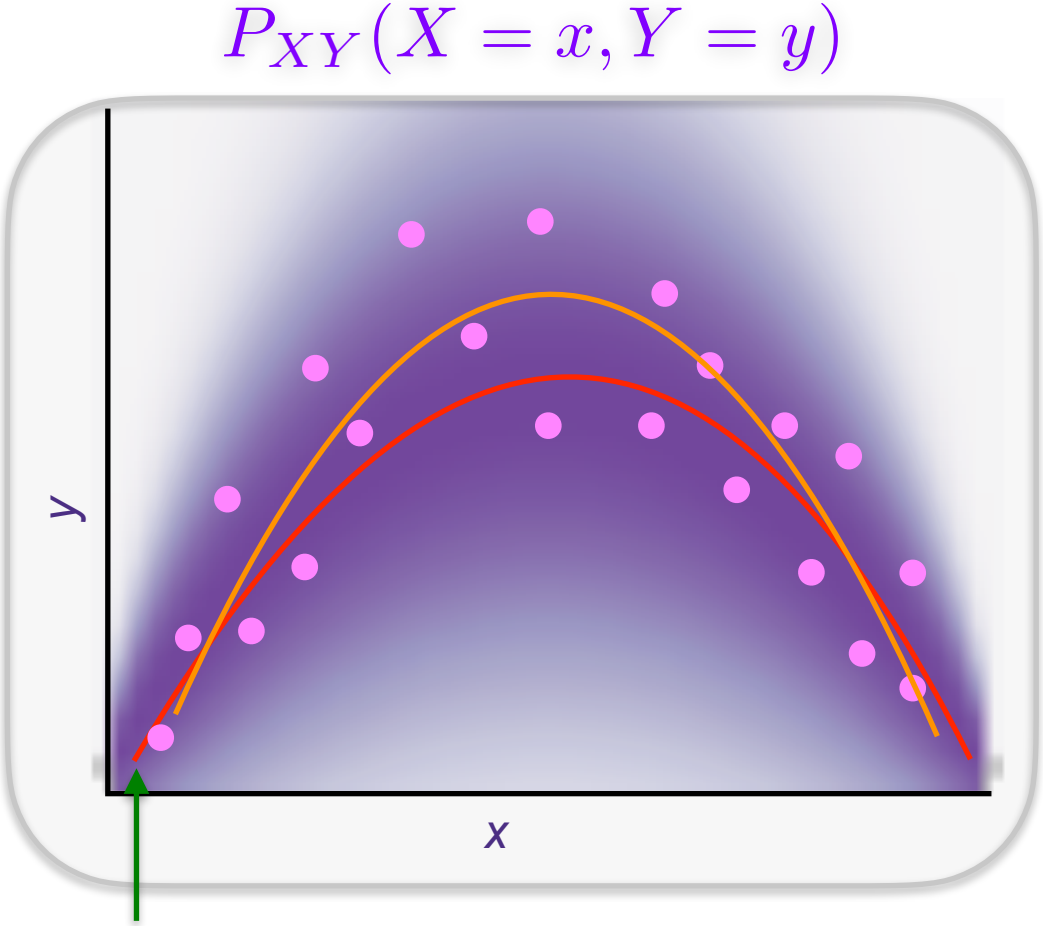
Ideally, we want to find:

$$\eta(x) = \mathbb{E}_{Y|X}[Y|X = x]$$

But we only have samples:

$(x_i, y_i) \overset{i.i.d.}{\sim} P_{XY}$ for $i = 1, \ldots, n$

Statistical Learning



Ideally, we want to find:

$$\eta(x) = \mathbb{E}_{Y|X}[Y|X = x]$$

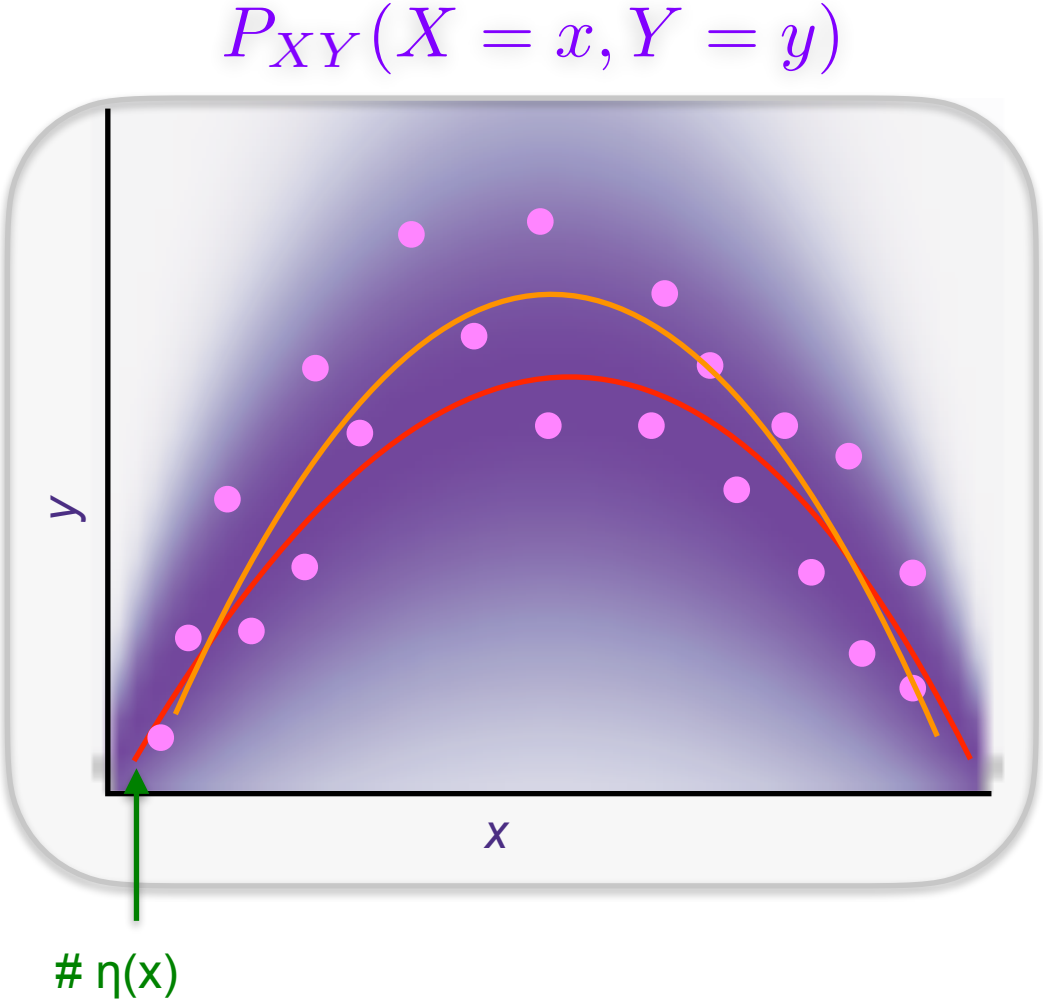
But we only have samples:

$$(x_i, y_i) \overset{i.i.d.}{\sim} P_{XY} \quad \text{for } i = 1, \ldots, n$$

and are restricted to a function class (e.g., linear) so we compute:

$$\widehat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

Statistical Learning



Ideally, we want to find:

$$\eta(x) = \mathbb{E}_{Y|X}[Y|X = x]$$

But we only have samples:

$(x_i, y_i) \overset{i.i.d.}{\sim} P_{XY}$ for $i = 1, \ldots, n$

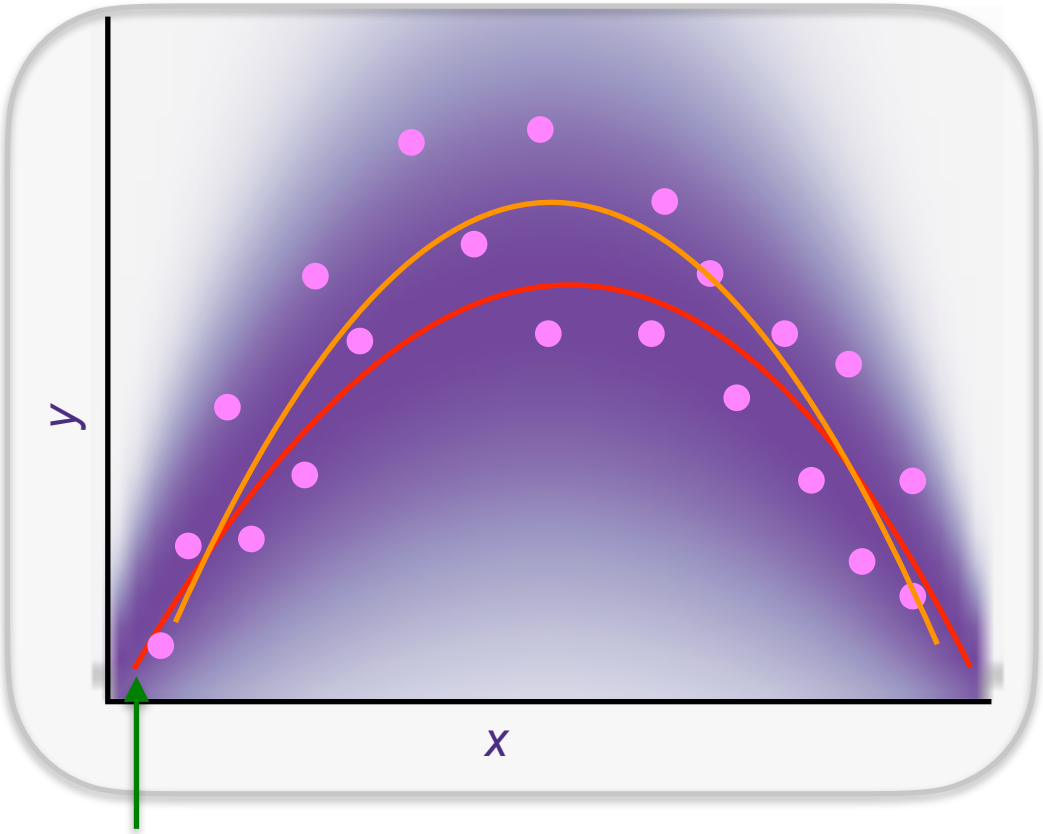
and are restricted to a function class (e.g., linear) so we compute:

$$\widehat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

$\hat{f} \neq \eta(x)$

Statistical Learning

$P_{XY}(X = x, Y = y)$



η(x)

Ideally, we want to find:

$$\eta(x) = \mathbb{E}_{Y|X}[Y|X = x]$$

But we only have samples:

$(x_i, y_i) \overset{i.i.d.}{\sim} P_{XY}$ for $i = 1, \ldots, n$

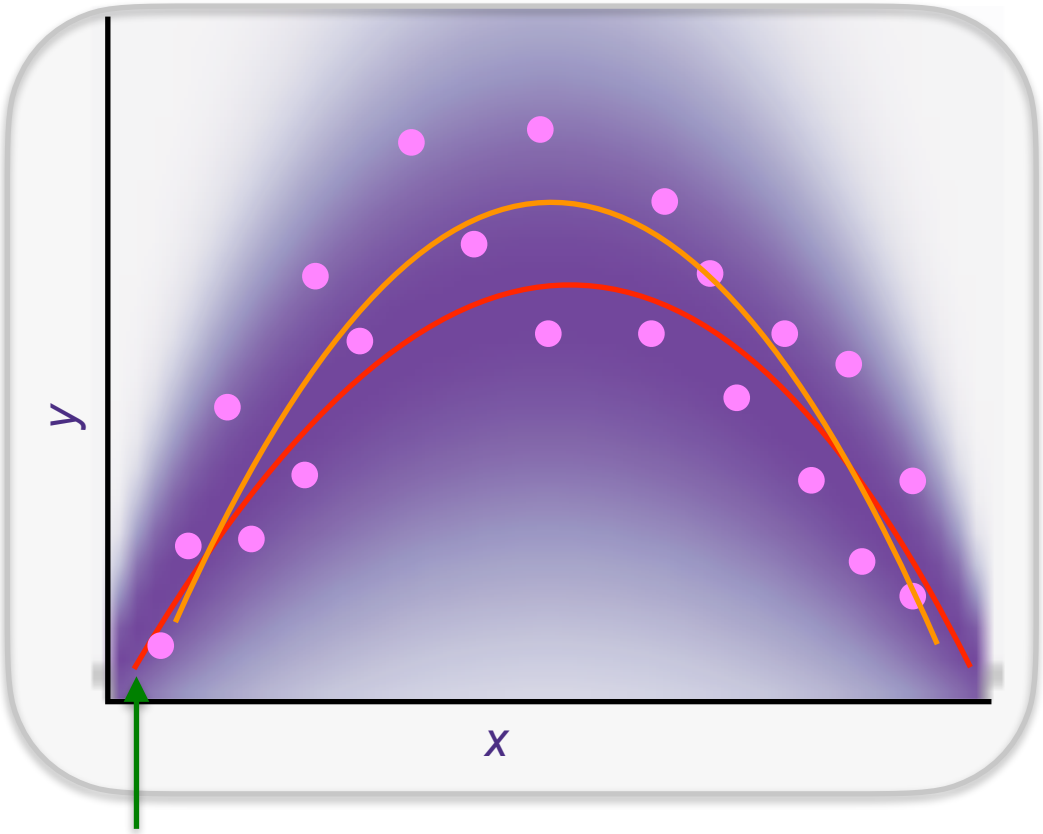
and are restricted to a function class (e.g., linear) so we compute:

$$\widehat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

$\hat{f} \neq \eta(x)$ Why?

Statistical Learning

$P_{XY}(X = x, Y = y)$



η(x)

Ideally, we want to find:

$$\eta(x) = \mathbb{E}_{Y|X}[Y|X = x]$$

But we only have samples:

$(x_i, y_i) \overset{i.i.d.}{\sim} P_{XY}$ for $i = 1, \ldots, n$

and are restricted to a function class (e.g., linear) so we compute:

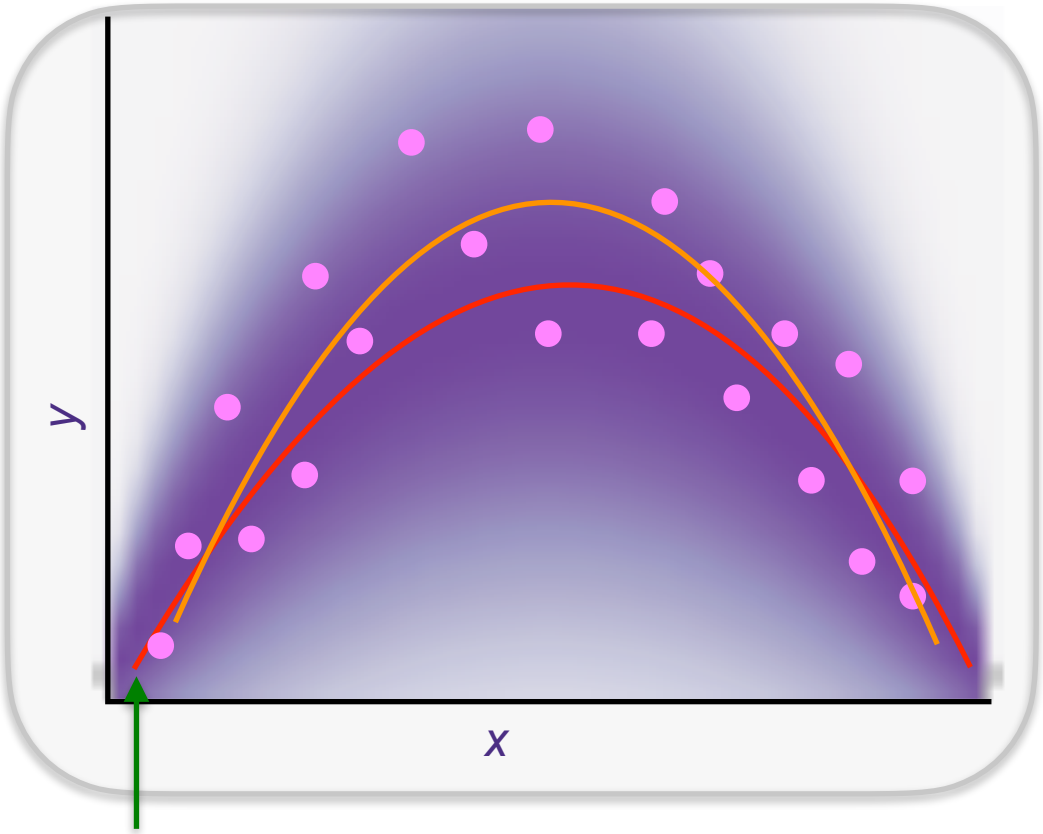
$$\widehat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

$\hat{f} \neq \eta(x)$ Why?

n is limited

Statistical Learning

$P_{XY}(X = x, Y = y)$



η(x)

Ideally, we want to find:

$$\eta(x) = \mathbb{E}_{Y|X}[Y|X = x]$$

But we only have samples:

$(x_i, y_i) \overset{i.i.d.}{\sim} P_{XY}$ for $i = 1, \ldots, n$

and are restricted to a function class (e.g., linear) so we compute:

$$\widehat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

$\hat{f} \neq \eta(x)$ Why?

n is limited

could have chosen wrong model class *F*

Empirical Reconstruction Error (ERE)

Empirical Reconstruction Error (ERE)

What is the $\mathbb{E}[\hat{f}]$?

Empirical Reconstruction Error (ERE)

What is the $\mathbb{E}[\hat{f}]$?

$$\mathbb{E}_{XY}[\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2]$$

Empirical Reconstruction Error (ERE)

What is the $\mathbb{E}[\hat{f}]$?

$$\mathbb{E}_{XY}[\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2]$$

By linearity of expectation

Empirical Reconstruction Error (ERE)

What is the $\mathbb{E}[\hat{f}]$?

$$\mathbb{E}_{XY}[\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2]$$

$$= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[(y_i - f(x_i))^2]$$ # By linearity of expectation

Empirical Reconstruction Error (ERE)

What is the $\mathbb{E}[\hat{f}]$?

$$\mathbb{E}_{XY}[\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2]$$

$$= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[(y_i - f(x_i))^2]$$ # By linearity of expectation

by IID: $\mathbb{E}[x_i]$ for any $i = \mathbb{E}[x] \forall x$

Empirical Reconstruction Error (ERE)

What is the $\mathbb{E}[\hat{f}]$?

$$\mathbb{E}_{XY}[\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2]$$

$$= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[(y_i - f(x_i))^2]$$ # By linearity of expectation

$$= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[(y - f(x))^2]$$ by IID: $\mathbb{E}[x_i]$ for any $i = \mathbb{E}[x] \forall x$

Empirical Reconstruction Error (ERE)

What is the $\mathbb{E}[\hat{f}]$?

$$\mathbb{E}_{XY}[\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2]$$

$$= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[(y_i - f(x_i))^2]$$ # By linearity of expectation

$$= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[(y - f(x))^2]$$ by IID: $\mathbb{E}[x_i]$ for any $i = \mathbb{E}[x] \forall x$

Empirical Reconstruction Error (ERE)

What is the $\mathbb{E}[\hat{f}]$?

$$\mathbb{E}_{XY}[\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2]$$

$$= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[(y_i - f(x_i))^2]$$ # By linearity of expectation

$$= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[(y - f(x))^2]$$ by IID: $\mathbb{E}[x_i]$ for any $i = \mathbb{E}[x] \forall x$

$$= \mathbb{E}[(y - f(x))^2]$$

Empirical Reconstruction Error (ERE)

What is the $\mathbb{E}[\hat{f}]$?

$$\mathbb{E}_{XY}[\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2]$$

$$= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[(y_i - f(x_i))^2]$$ # By linearity of expectation

$$= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[(y - f(x))^2]$$ by IID: $\mathbb{E}[x_i]$ for any $i = \mathbb{E}[x] \forall x$

$$= \mathbb{E}[(y - f(x))^2]$$ # Yet again, it works out to the mean squared error!

Empirical Reconstruction Error (ERE)

What is the $\mathbb{E}[\hat{f}]$?

$$\mathbb{E}_{XY}[\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2]$$

$$= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[(y_i - f(x_i))^2]$$ # By linearity of expectation

$$= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[(y - f(x))^2]$$ by IID: $\mathbb{E}[x_i]$ for any $i = \mathbb{E}[x] \forall x$

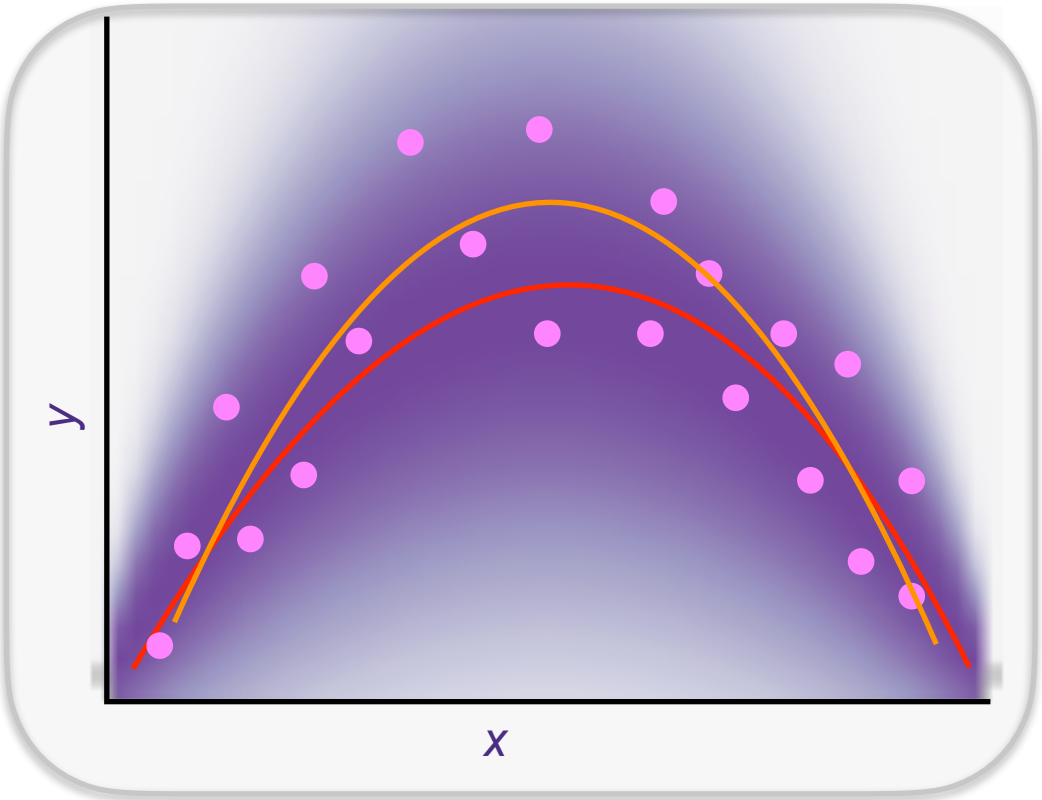
$$= \mathbb{E}[(y - f(x))^2]$$ # Yet again, it works out to the mean squared error!

Minimizing ERE → minimizing squared error loss in general

(in expectation….)

Statistical Learning

$P_{XY}(X = x, Y = y)$



Ideally, we want to find:

$$\eta(x) = \mathbb{E}_{Y|X}[Y|X = x]$$

But we only have samples:

$(x_i, y_i) \overset{i.i.d.}{\sim} P_{XY}$ for $i = 1, \ldots, n$

and are restricted to a function class (e.g., linear) so we compute:

$$\widehat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

We care about future predictions: $\mathbb{E}_{XY}[(Y - \widehat{f}(X))^2]$

Statistical Learning

$P_{XY}(X = x, Y = y)$



Ideally, we want to find:

$$\eta(x) = \mathbb{E}_{Y|X}[Y|X = x]$$

But we only have samples:

$(x_i, y_i) \overset{i.i.d.}{\sim} P_{XY}$ for $i = 1, \ldots, n$

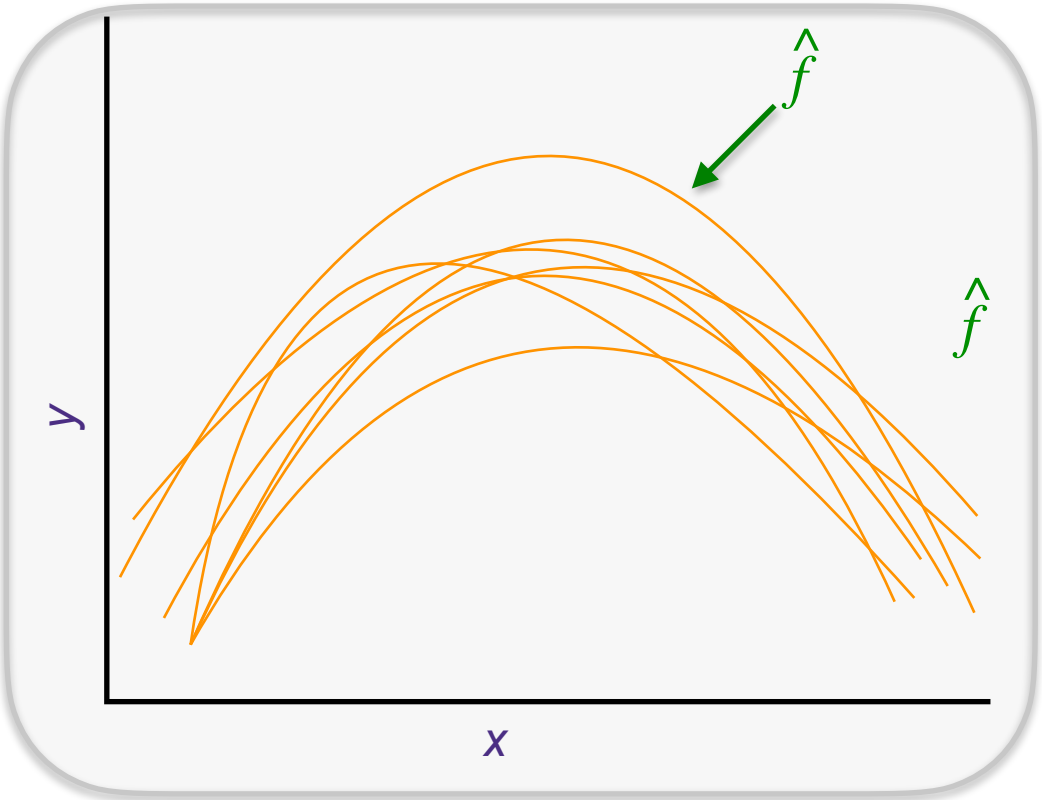
and are restricted to a function class (e.g., linear) so we compute:

$$\widehat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

Each draw $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ results in different $\widehat{f}$

Statistical Learning

$P_{XY}(X = x, Y = y)$



Ideally, we want to find:

$$\eta(x) = \mathbb{E}_{Y|X}[Y|X = x]$$

But we only have samples:

$(x_i, y_i) \overset{i.i.d.}{\sim} P_{XY}$ for $i = 1, \ldots, n$

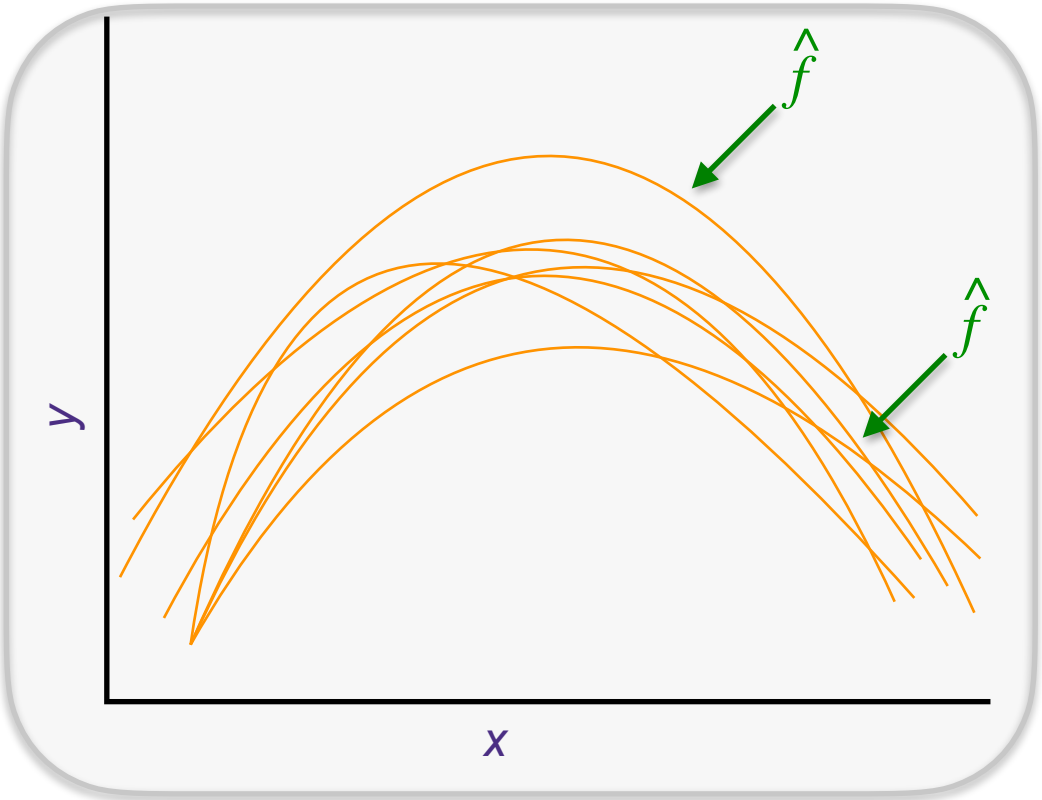
and are restricted to a function class (e.g., linear) so we compute:

$$\widehat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

Each draw $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ results in different $\widehat{f}$

Statistical Learning

$P_{XY}(X = x, Y = y)$



Ideally, we want to find:

$$\eta(x) = \mathbb{E}_{Y|X}[Y|X = x]$$

But we only have samples:

$(x_i, y_i) \overset{i.i.d.}{\sim} P_{XY}$ for $i = 1, \ldots, n$

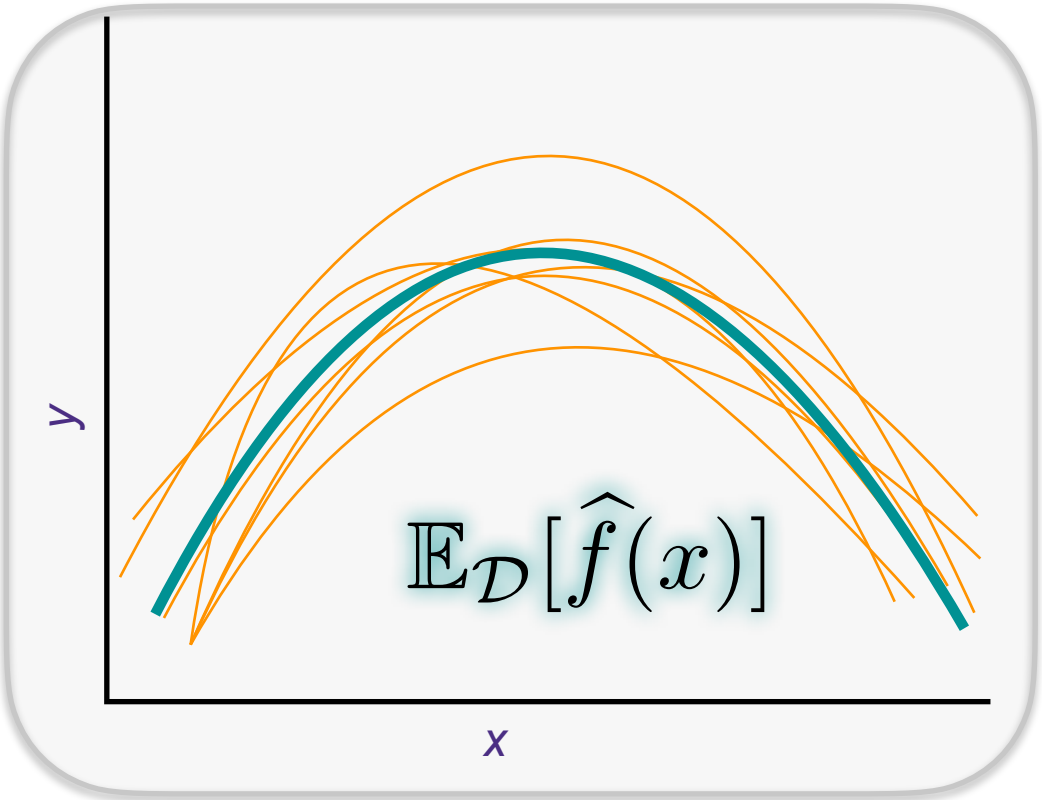
and are restricted to a function class (e.g., linear) so we compute:

$$\widehat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

Each draw $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ results in different $\widehat{f}$

Statistical Learning

$P_{XY}(X = x, Y = y)$



Ideally, we want to find:

$$\eta(x) = \mathbb{E}_{Y|X}[Y|X = x]$$

But we only have samples:

$(x_i, y_i) \overset{i.i.d.}{\sim} P_{XY}$ for $i = 1, \ldots, n$

and are restricted to a function class (e.g., linear) so we compute:

$$\widehat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

Each draw $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ results in different $\widehat{f}$

Bias-Variance Tradeoff

$$\eta(x) = \mathbb{E}_{Y|X}[Y|X = x] \qquad \widehat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

$$\mathbb{E}_{Y|X}[\mathbb{E}_{\mathcal{D}}[(Y - \widehat{f}_{\mathcal{D}}(x))^2] \big| X = x] = \mathbb{E}_{Y|X}[\mathbb{E}_{\mathcal{D}}[(Y - \eta(x) + \eta(x) - \widehat{f}_{\mathcal{D}}(x))^2] \big| X = x]$$

Bias-Variance Tradeoff

$$\eta(x) = \mathbb{E}_{Y|X}[Y|X = x] \qquad \widehat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

$$\mathbb{E}_{Y|X}[\mathbb{E}_{\mathcal{D}}[(Y - \widehat{f}_{\mathcal{D}}(x))^2]\big|X = x] = \mathbb{E}_{Y|X}[\mathbb{E}_{\mathcal{D}}[(Y - \eta(x) + \eta(x) - \widehat{f}_{\mathcal{D}}(x))^2]\big|X = x]$$

$$= \mathbb{E}_{Y|X}\Big[\mathbb{E}_{\mathcal{D}}[(Y - \eta(x))^2 + 2(Y - \eta(x))(\eta(x) - \widehat{f}_{\mathcal{D}}(x)) + (\eta(x) - \widehat{f}_{\mathcal{D}}(x))^2]\big|X = x\Big]$$

$$= \mathbb{E}_{Y|X}[(Y - \eta(x))^2\big|X = x] + \mathbb{E}_{\mathcal{D}}[(\eta(x) - \widehat{f}_{\mathcal{D}}(x))^2]$$

irreducible error
Caused by stochastic label noise

learning error
Caused by either using too "simple" of a model or not enough data to learn the model accurately

Bias-Variance Tradeoff

$$\eta(x) = \mathbb{E}_{Y|X}[Y|X = x] \qquad \widehat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

$$\mathbb{E}_{\mathcal{D}}[(\eta(x) - \widehat{f}_{\mathcal{D}}(x))^2] = \mathbb{E}_{\mathcal{D}}[(\eta(x) - \mathbb{E}_{\mathcal{D}}[\widehat{f}_{\mathcal{D}}(x)] + \mathbb{E}_{\mathcal{D}}[\widehat{f}_{\mathcal{D}}(x)] - \widehat{f}_{\mathcal{D}}(x))^2]$$

Bias-Variance Tradeoff

$$\eta(x) = \mathbb{E}_{Y|X}[Y|X = x] \qquad \widehat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

$$\mathbb{E}_{\mathcal{D}}[(\eta(x) - \widehat{f}_{\mathcal{D}}(x))^2] = \mathbb{E}_{\mathcal{D}}[(\eta(x) - \mathbb{E}_{\mathcal{D}}[\widehat{f}_{\mathcal{D}}(x)] + \mathbb{E}_{\mathcal{D}}[\widehat{f}_{\mathcal{D}}(x)] - \widehat{f}_{\mathcal{D}}(x))^2]$$

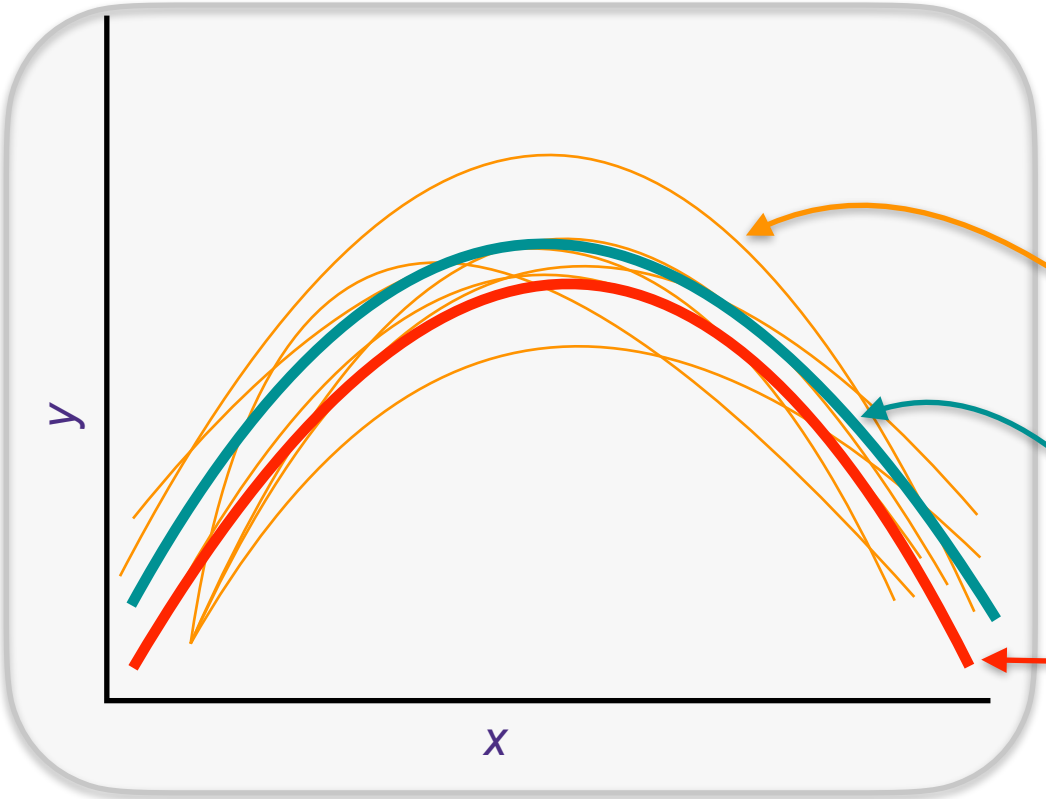
$$= \mathbb{E}_{\mathcal{D}}[(\eta(x) - \mathbb{E}_{\mathcal{D}}[\widehat{f}_{\mathcal{D}}(x)])^2 + 2(\eta(x) - \mathbb{E}_{\mathcal{D}}[\widehat{f}_{\mathcal{D}}(x)])(\mathbb{E}_{\mathcal{D}}[\widehat{f}_{\mathcal{D}}(x)] - \widehat{f}_{\mathcal{D}}(x))$$
$$+ (\mathbb{E}_{\mathcal{D}}[\widehat{f}_{\mathcal{D}}(x)] - \widehat{f}_{\mathcal{D}}(x))^2]$$

$$= (\eta(x) - \mathbb{E}_{\mathcal{D}}[\widehat{f}_{\mathcal{D}}(x)])^2 + \mathbb{E}_{\mathcal{D}}[(\mathbb{E}_{\mathcal{D}}[\widehat{f}_{\mathcal{D}}(x)] - \widehat{f}_{\mathcal{D}}(x))^2]$$

biased squared **variance**

Statistical Learning

$$\underbrace{\mathbb{E}_{\mathcal{D}}[(\eta(x) - \widehat{f}_{\mathcal{D}}(x))^2]}_{\text{learning error}} = \underbrace{(\eta(x) - \mathbb{E}_{\mathcal{D}}[\widehat{f}_{\mathcal{D}}(x)])^2}_{\text{biased squared}} + \underbrace{\mathbb{E}_{\mathcal{D}}[(\mathbb{E}_{\mathcal{D}}[\widehat{f}_{\mathcal{D}}(x)] - \widehat{f}_{\mathcal{D}}(x))^2]}_{\text{variance}}$$



$$\widehat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

$$\mathbb{E}_{\mathcal{D}}[\widehat{f}(x)]$$

$$\eta(x) = \mathbb{E}_{Y|X}[Y|X = x]$$

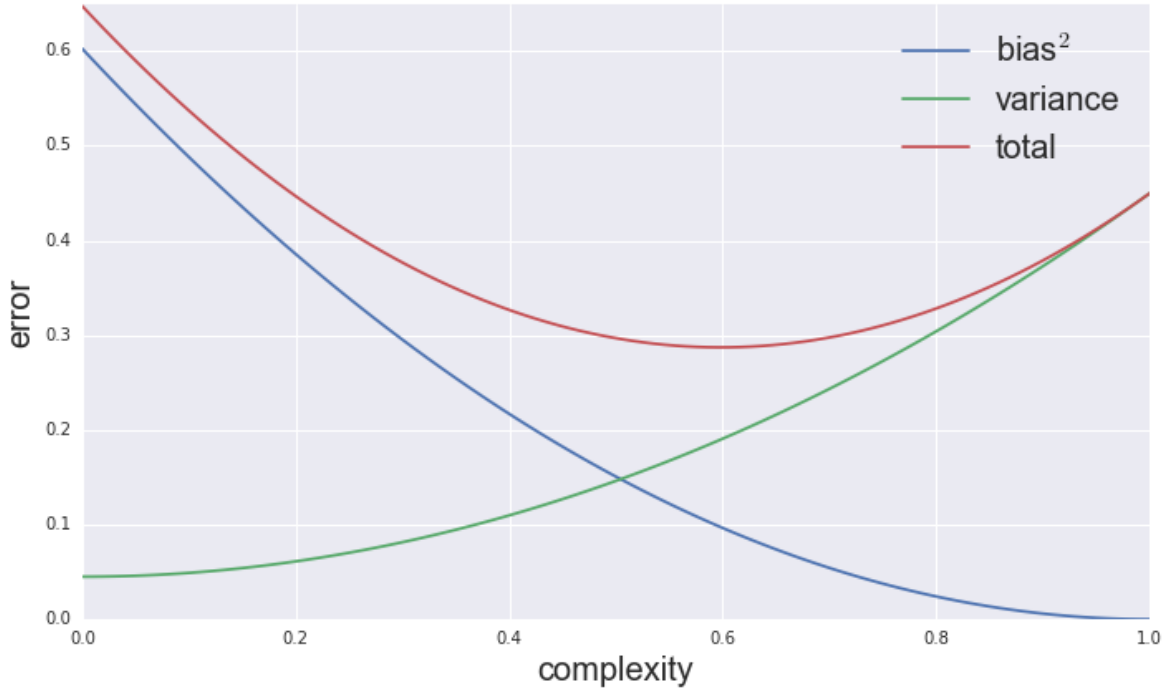
Bias-Variance Tradeoff

$$\mathbb{E}_{Y|X}[\mathbb{E}_{\mathcal{D}}[(Y - \widehat{f}_{\mathcal{D}}(x))^2]\big|X = x] = \mathbb{E}_{Y|X}[(Y - \eta(x))^2\big|X = x]$$

irreducible error

$$+(\eta(x) - \mathbb{E}_{\mathcal{D}}[\widehat{f}_{\mathcal{D}}(x)])^2 + \mathbb{E}_{\mathcal{D}}[(\mathbb{E}_{\mathcal{D}}[\widehat{f}_{\mathcal{D}}(x)] - \widehat{f}_{\mathcal{D}}(x))^2]$$

biased squared **variance**



Bias-Variance Demo

See colab notebook

Bias-variance tradeoff

Let's define a ground truth $\eta(x) = E[Y|X = x]$ to be a degree-5 polynomial.
Then, define $P(Y|X = x) = \mathcal{N}(x, \sigma^2)$

In [1]:

```python
import numpy as np
import matplotlib.pyplot as plt

# Define \eta = E[Y|X] as a 5-th order polynomial
# (this is the same function we used in demo_polynomial.ipynb)
eta = lambda x: (x-.99)*(x-.4)*(x-.25)*(x+.6)*(x+.8)

# Generate samples from P(X,Y)
def sample_Pxy(n, noise_sigma=0.03):

    # Sample inputs x from P(x).
    x = np.random.uniform(-1,1,n)
    x = np.sort(x)
    x[0]=-1
    x[n-1]=1

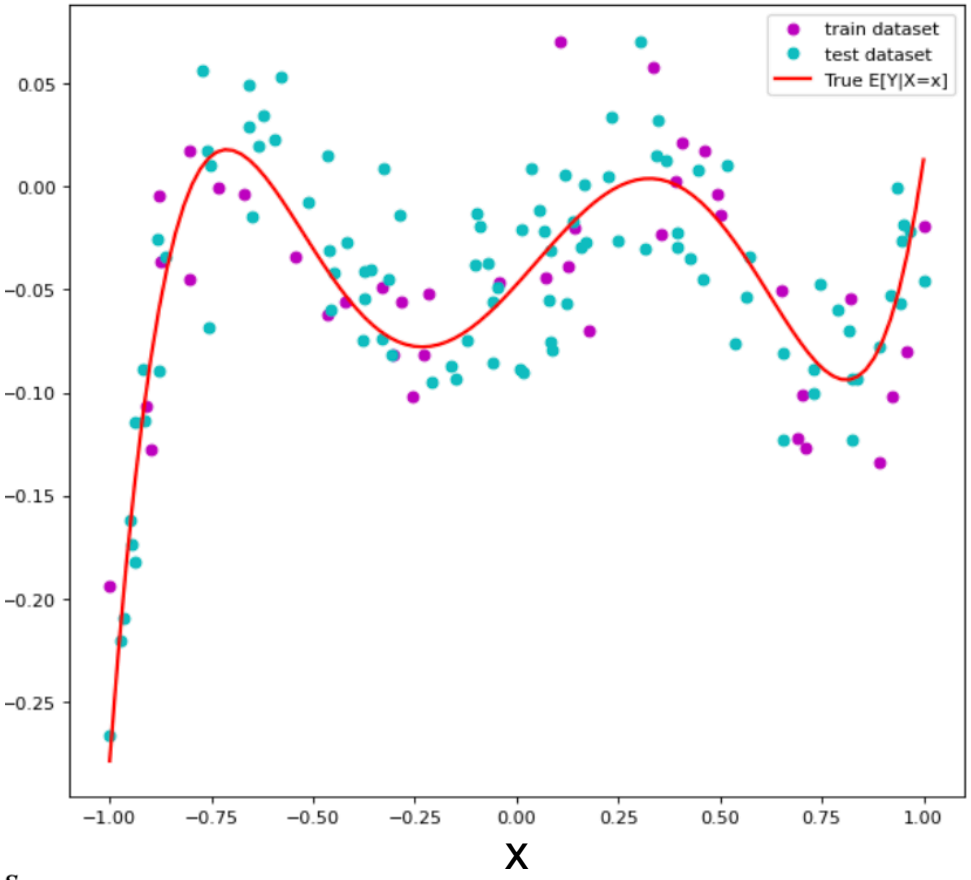
    # Draw samples from P(Y|X)
    y = eta(x) + noise_sigma*np.random.randn(n)

    return x,y

# Generate training data.
n_train = 40 # sample size
x, y = sample_Pxy(n=n_train)

# Generate test data.
n_test = 100
x_, y_ = sample_Pxy(n=n_test)

# Plot the samples and ground truth.
t = np.linspace(-1,1,100)
y0 = eta(t)
fig=plt.figure(figsize=(9, 8), dpi= 80, facecolor='w', edgecolor='k')
plt.plot(x,y,'mo',label='train dataset')
plt.plot(x_,y_,'co',label='test dataset')
plt.plot(t,y0,'r-', linewidth=2, label='True E[Y|X=x]')
plt.legend()
plt.show()
```



In typical scenarios, we only have one set of samples, which we separate into S_{test} and S_{train}

- it is critical that those two sets do not overlap and the sets are chosen randomly to ensure that the test error is independent of the training error, and also the test samples are coming frmo the same distribution as the new samples that will come in the future

However, in order to understand how the test error behaves (theoretically), we consider the expected test error, and call it true error: i.e. $\mathcal{L}_{\text{true}} = \mathbb{E}[\mathcal{L}_{\text{test}}]$

- test error is an **unbiased** estimate of the true error
- true error is unobservable (we canot copmute it, given finite samples)
- but we care the most about the true error
- so we use test error as a surrogate or an approximation

In order to compute the true error, we simulate a process where we get many fresh samples, and train new predictor each time with the fresh set of samples. It is important to understand that the resulting predictor $f_{S_{\text{train}}}(\cdot)$ is a random function, where the randomness comes from the fresh random training data set S_{train}. We will draw many such random functions, plot them, and see how test, train, true errors behave.

In [2]:

```
num_runs = 100
yhat_simple = np.zeros((num_runs,len(t)))
yhat_complex = np.zeros((num_runs,len(t)))
yhat_justright = np.zeros((num_runs,len(t)))

def poly_features(x, degree):
    """
    Generates a matrix where each column corresponds to x raised to a power from 0 to 'degree'.

    Parameters:
    x (numpy array): The input data.
    degree (int): The maximum degree of the polynomial features.

    Returns:
    numpy array: A matrix where the columns are [1, x, x^2, ..., x^degree].
    """
    return np.vstack([x**i for i in range(degree + 1)]).T

run=0
while run < num_runs:
    n = 40 # sample size
    x,y = sample_Pxy(n)

    # degree-3 polynomial linear regression
    p=3
    X = poly_features(x, degree=p)
    w = np.matmul(np.matmul(np.linalg.inv(np.matmul(X.T,X)),X.T),y)
    T_ = poly_features(t, degree=p)
    yhat_simple[int(run)] = np.matmul(T_,w)
    yh = np.matmul(X,w)
    X_ = poly_features(x_, degree=p)
    y_h= np.matmul(X_,w)
    w_ = w

    # degree-5 polynomial linear regression
    p=5
    X = poly_features(x, degree=p)
    w = np.matmul(np.matmul(np.linalg.inv(np.matmul(X.T,X)),X.T),y)
    T_ = poly_features(t, degree=p)
    yhat_justright[int(run)] = np.matmul(T_,w)
    yh = np.matmul(X,w)
    X_ = poly_features(x_, degree=p)
    y_h= np.matmul(X_,w)
    w_ = w

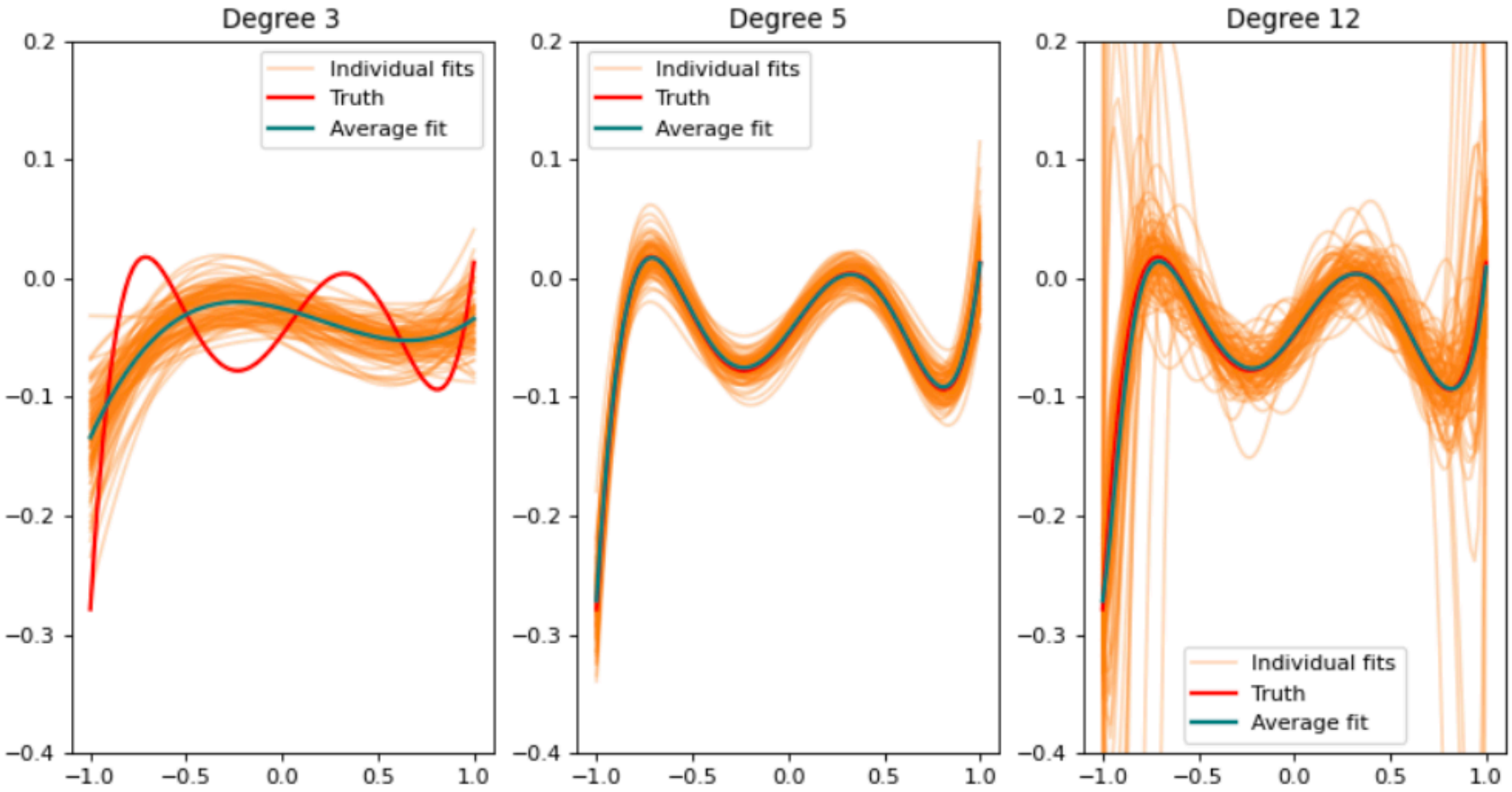
    # degree-12 polynomial linear regression
    p=12
    X = poly_features(x, degree=p)
    w = np.array(p+1)
    w = np.matmul(np.matmul(np.linalg.inv(np.matmul(X.T,X)),X.T),y)
    T_ = poly_features(t, degree=p)
    yhat_complex[int(run)] = np.matmul(T_,w)
    yh1 = np.matmul(X,w)
    X_ = poly_features(x_, degree=p)
    y_h1= np.matmul(X_,w)

    run=run+1

def build_plot(yhat, title):
    ave_fit = np.zeros(np.size(t))
    for irun in range(1, num_runs):
        plt.plot(t,yhat[int(irun)], color='tab:orange',alpha=0.3)
        ave_fit = ave_fit + yhat[int(irun)]
    plt.plot(t,yhat[0],'tab:orange',alpha=0.3,label='Individual fits')
    plt.plot(t,y0,'r-', linewidth=2, label='Truth')
    plt.plot(t,(1/float(num_runs))*ave_fit, color='teal', linewidth=2,label='Average fit')
    plt.legend()
    plt.title(title)
    axes = plt.gca()
    axes.set_ylim([-0.4,0.2])

fig=plt.figure(figsize=(12, 6), dpi= 80, facecolor='w', edgecolor='k')
plt.subplot(1,3,1)
build_plot(yhat_simple, 'Degree 3')
plt.subplot(1,3,2)
build_plot(yhat_justright, 'Degree 5')
plt.subplot(1,3,3)
build_plot(yhat_complex, 'Degree 12')
plt.show()

```



Takeaways

- True function has degree 5, with additive noise
- Degree 3 fit has very high bias because the teal line, which is the average of the fits, is very different from true red line (because 3 < 5). Each individual fit (orange) varies about the average fit (teal) moderately indicating moderate variance. The overall error (bias^2 + variance) of each individual fit is high.
- Degree 12 has very low bias because the teal line is almost equal to the red (because 12 > 5). But each individual fit varies a lot about its average teal line indicating high variance. The overall error (bias^2 + variance) of each individual fit is high.
- Degree 5 has very low bias because the teal line is almost equal to the red (because 5 = 5). And each individual fit varies only a little about its average teal line indicating small variance. The overall error (bias^2 + variance) of each individual fit is low.