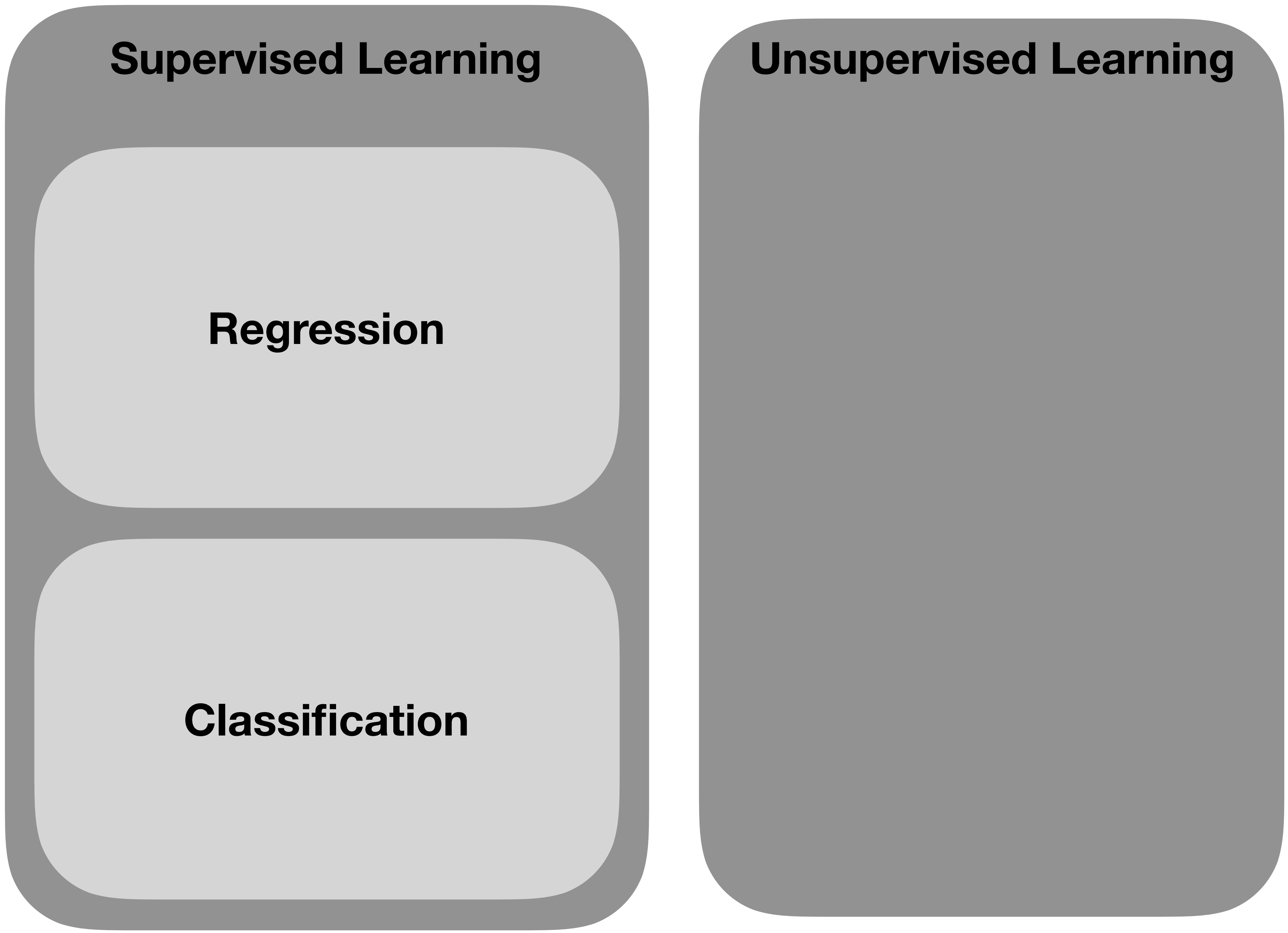
Regression

Sewoong Oh

CSE446
University of Washington

Machine Learning



Predictors

Data fitting

- goal: predicting “How much is my house worth?”

- data

$$
\begin{aligned}
(x_1, y_1) &= (\,2318\, sq.ft.\,,\ \$\,315k) \\
(x_2, y_2) &= (\,1985\, sq.ft.\,,\ \$\,295k) \\
(x_3, y_3) &= (\,2861\, sq.ft.\,,\ \$\,370k) \\
&\ \ \vdots \qquad\qquad \vdots \\
(x_n, y_n) &= (\,2055\, sq.ft.\,,\ \$\,320k)
\end{aligned}
$$

← data pair or example

- hope/belief: We think $y \in \mathbf{R}$ and $x \in \mathbf{R}^d$ are approximately related by

$$y \approx f_0(x)$$

- x is called the **input data**
  y is called the outcome, response, target, label, or dependent variable

- y is what we want to predict

Features

- often, the input data needs to be pre-processed to be applied to a machine learning algorithm
- these predefined processed representation of the input data is called **features**
- we use x to denote raw data input, and $h: \mathbf{R}^d \rightarrow \mathbf{R}^k$ to denote corresponding **feature vector**

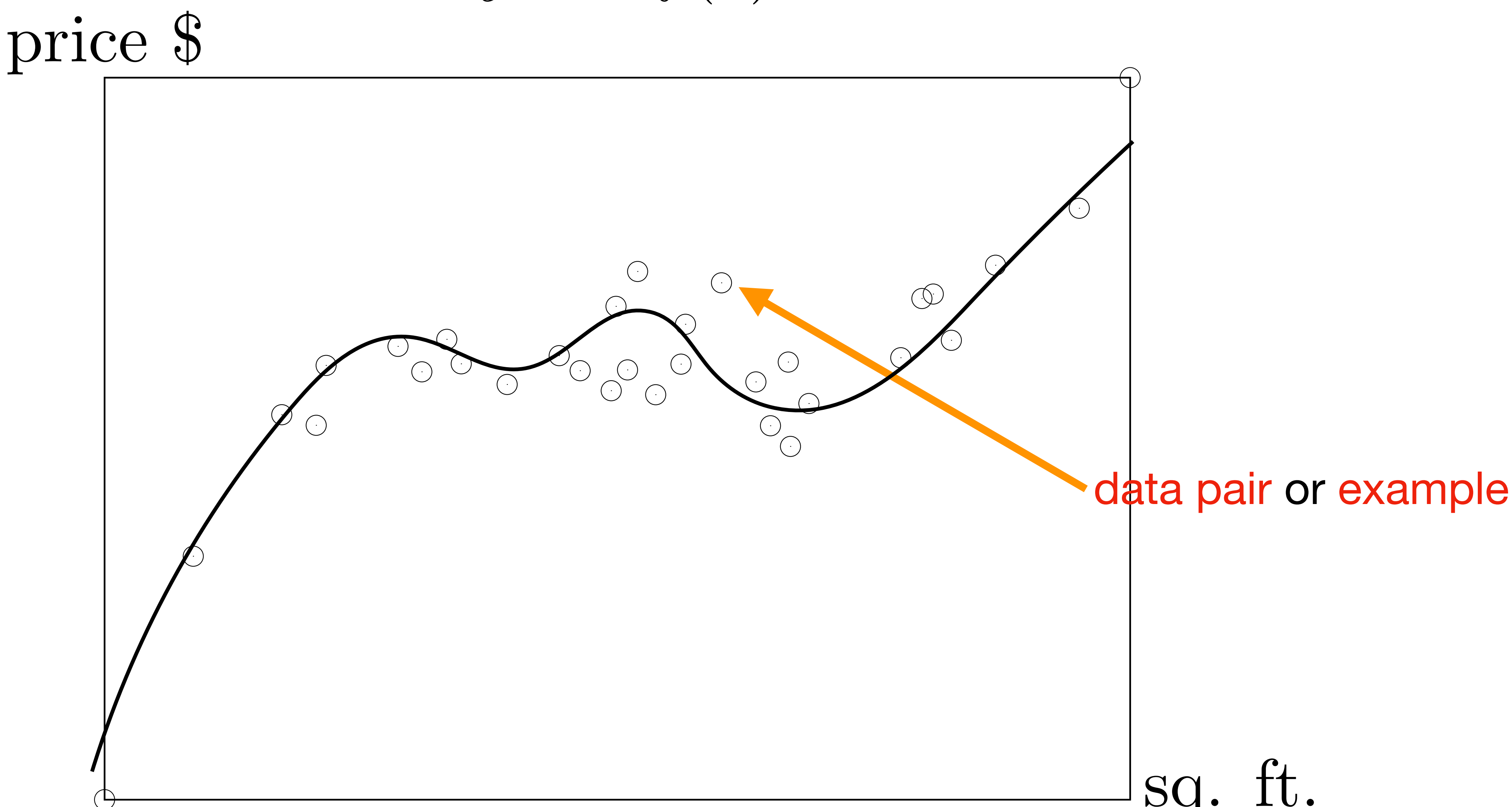
$$x = \begin{bmatrix} x[1] \\ x[2] \\ \vdots \\ x[d] \end{bmatrix} \qquad h(x) = \begin{bmatrix} h_0(x) \\ h_1(x) \\ \vdots \\ h_k(x) \end{bmatrix}$$

- for example,
  - *x* is a document, then h(x) is word count histogram (k=273,000 for English or 106,230 for Chinese)
  - x is an email, then h(x) is the count of trigger words
  - x is a facial image, then h(x) is hair color, beard, skin tone

Predictor

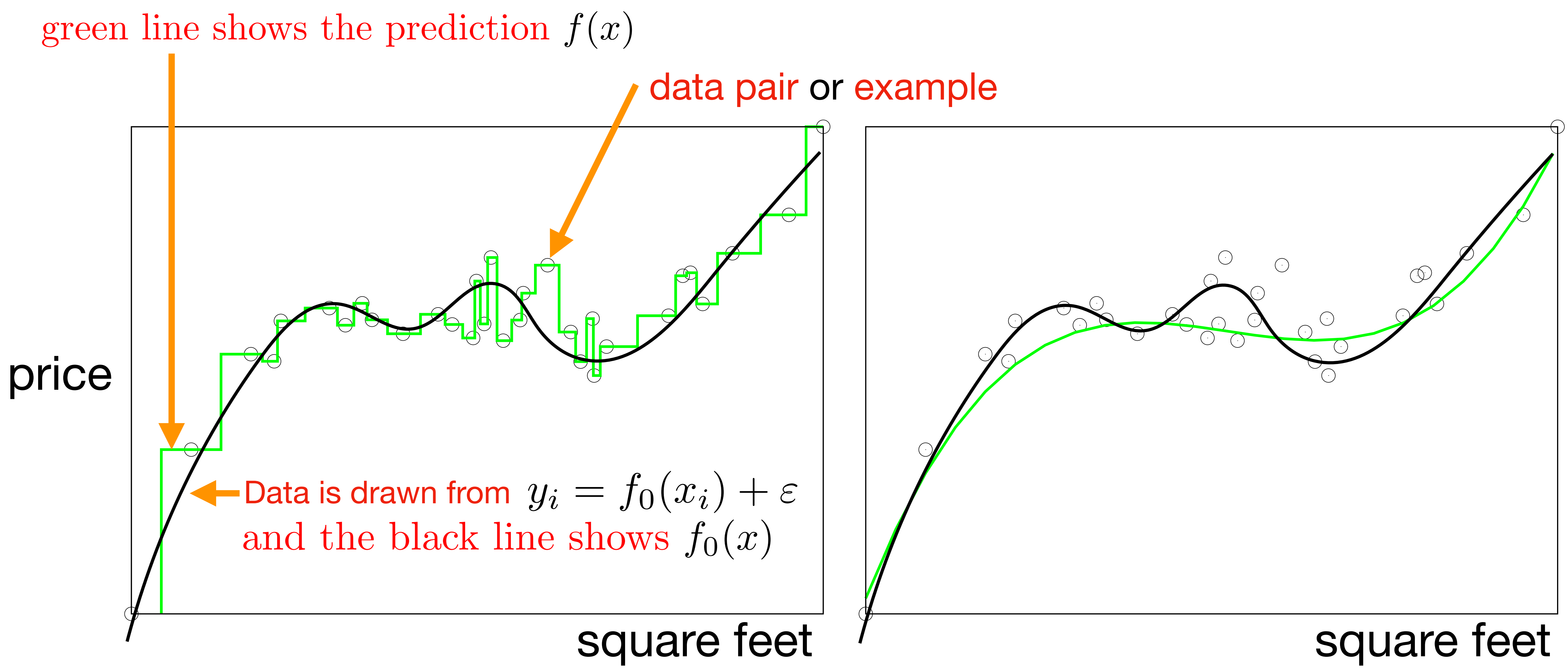
- we seek a **predictor** or **model** $f : \mathbf{R}^d \to \mathbf{R}$
- for an input data x, our prediction of the label y is

$$\hat{y} \;=\; f(x)$$



- small error on an example, $f(x_i) \approx y_i$, implies that we have a good prediction on the ith pair (x_i, y_i)

a machine learning algorithm is
a principled recipe for producing a predictor, given data



- left plot shows nearest neighbor prediction
- right plot shows cubic polynomial fit $f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$
- we want a good prediction on pairs we have not seen

demo1_1nn_lin.ipynb

Two schools of thoughts

- machine learning

Belongs to a set of of functions (to be defined by the statistician)

$$\text{given } \{(x_1, y_1), \ldots, (x_n, y_n)\}, \text{ find a predictor } \boxed{f \in \mathcal{F}}$$

- any machine learning algorithm can be derived from

Empirical Risk Minimization

$$y \simeq f_0(x)$$

with a given loss function $\mathcal{L}$

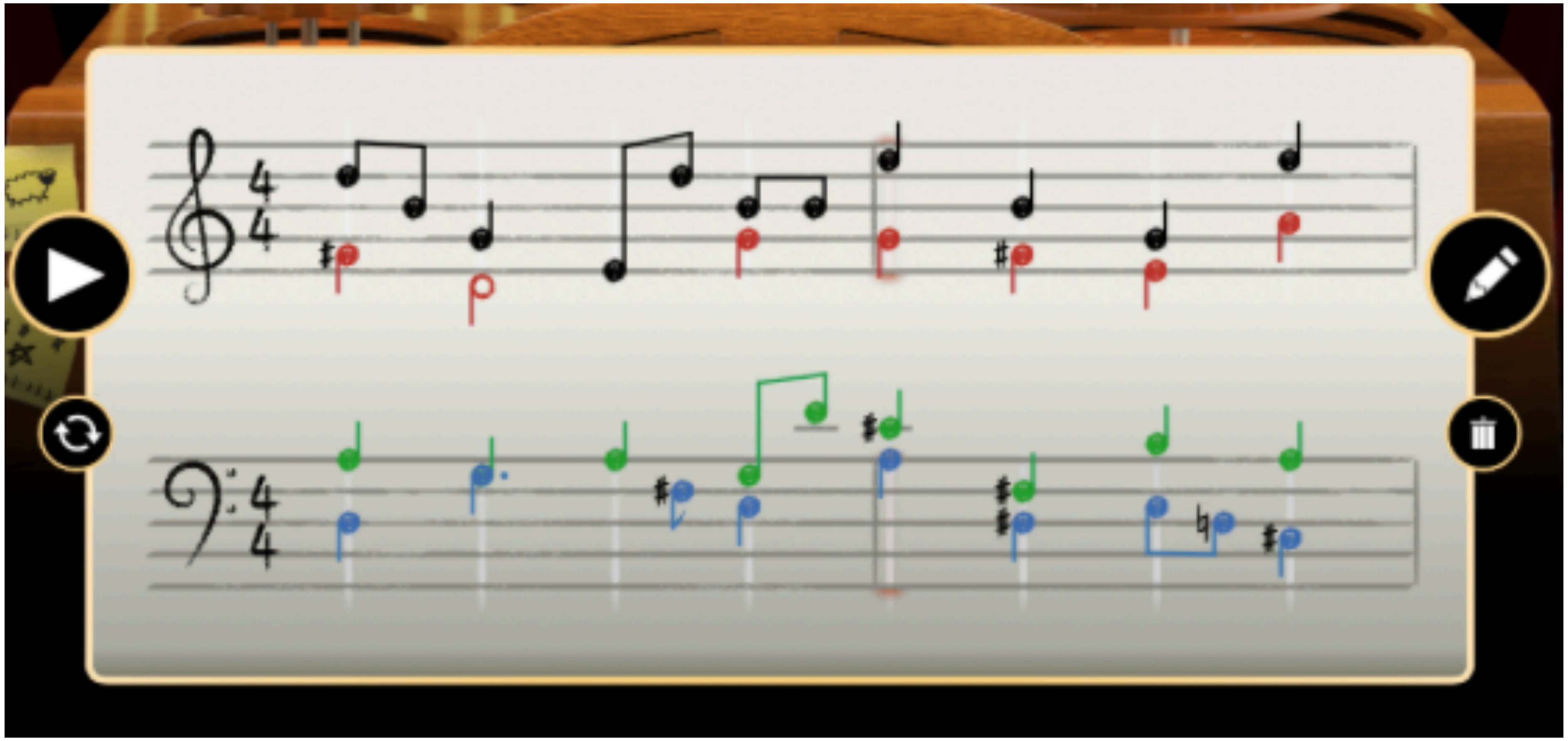
$$\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \mathcal{L}(f(x_i), y_i)$$

Maximum Likelihood Estimator

$$y = f_0(x) + \varepsilon$$

with known pdf of ε

$$\max_{f \in \mathcal{F}} \prod_{i=1}^{n} P(y_i = f(x_i) + \varepsilon)$$



soprano (S)

alto (A)

tenor (T)

bass (B)

- Example: Google Harmonizer
- Training data
  - Input data x: soprano notes
  - Output data y: alto, tenor, bass notes
- Test data
  - Input data x: soprano notes

$$x = [\ \underbrace{0, 0, 1, 0, \cdots, 0}_{\text{pitch of the first note}}, \underbrace{1, 0, \cdots, 0}_{\text{duration}}, 0, 0, 0, 0, 1, 0, \cdots]$$

https://www.google.com/doodles/celebrating-johann-sebastian-bach

Supervised Learning

Regression | Classification

- **Linear regression**
- **Nearest neighbor**
- **Decision trees**
- **Bootstrap**
- **Deep Neural Networks**

Unsupervised Learning

Linear predictors (linear regression):

first class of predictors of interest

Linear predictor

- The models we choose are guided by our belief in the real world data
- (one dimensional) **linear regression model** assumes each data point comes from a linear model with an independent additive noise ε_i

$$y_i \quad = \quad w_0 + w_1 x_i + \varepsilon_i$$

Independent noise added to each sample

- (one dimensional) **linear predictor** makes predictions with a linear function of the input x

$$\hat{y} \quad = \quad f(x) \quad = \quad \hat{w}_0 + \hat{w}_1 x$$

- strictly speaking, this is an **affine** model
  - Linear function has the form $f(x) = w_1 x$
  - Affine function has the form $f(x) = w_0 + w_1 x$
- in this class, we use affine functions, but call them linear, and use those terms interchangeably

Linear predictor

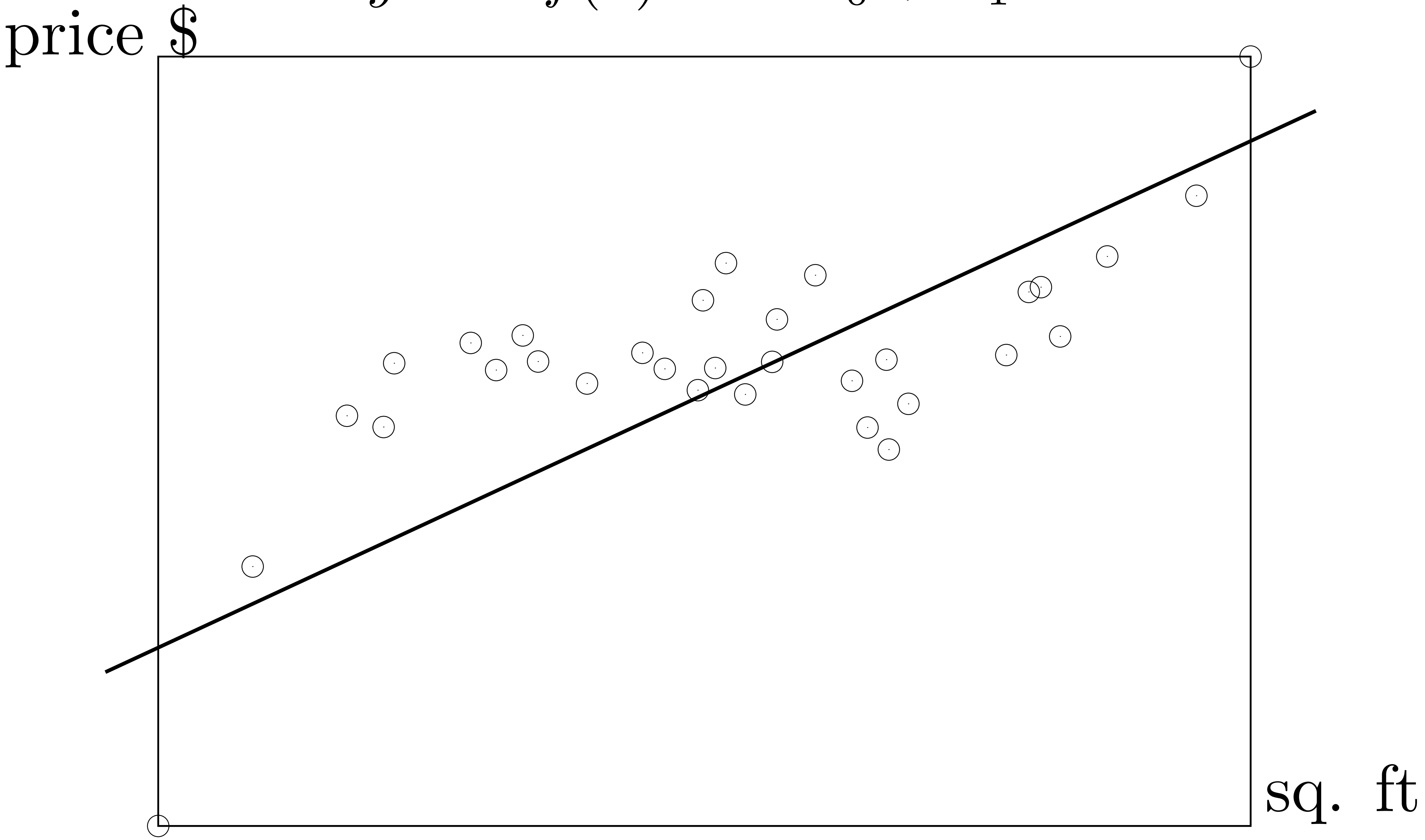
- in general, linear regression model can be multi-dimensional

$$f(x) = w_0 + w_1\, x[1] + w_2\, x[2] + \cdots + w_d\, x[d]$$
$$= \boxed{w^T x}$$

column vector $w = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix}$ row vector $w^T = \begin{bmatrix} w_0 & w_1 & \cdots & w_d \end{bmatrix}$ $x = \begin{bmatrix} 1 \\ x[1] \\ \vdots \\ x[d] \end{bmatrix}$

- $w_0, w_1, \ldots, w_d$ are the model parameters
- in this multi-dimensional case,
  - a **linear** function has the form
    $f(x) = w_1 x[1] + w_2 x[2] + \cdots + w_d x[d]$
  - and an **affine** function has the form
    $f(x) = w_0 + w_1 x[1] + w_2 x[2] + \cdots + w_d x[d]$

$$\hat{y} \;=\; f(x) \;=\; w_0 + w_1\, x$$



- once you fit a model to the data, e.g. $f(x) = 10,000 + 141\, x$
  - a seller with a house $x = 2511$ sq.ft. can predict the price
  - a buyer with money $y = \$364k$ can predict the size
- interpretation of the parameters
  - w_0 is the shift: price of land with no house
  - w_1 is the slope: how much price goes up per sq.ft.

Interpreting a linear model

- In general,

$$\hat{y} \;=\; f(x) \;=\; w_0 + w_1\, x[1] + w_2\, x[2] + \cdots + w_d\, x[d]$$

- w_3 is how much the (predicted) price increase when $x[3]$ increases by 1
- $w_7 = 0$ means the price does not depend on $x[7]$
- the constant term w_0 predicts when all features are zero
- for notational consistency, sometimes we say $x[0] = 1$ is a constant feature
- in general, w small implies the predictor is insensitive to changes in x

$$|f(x) - f(\tilde{x})| = |w^T x - w^T \tilde{x}| = |w^T(x - \tilde{x})| \leq \|w\|_2 \|x - \tilde{x}\|_2$$

This follows from Cauchy-Schwarz

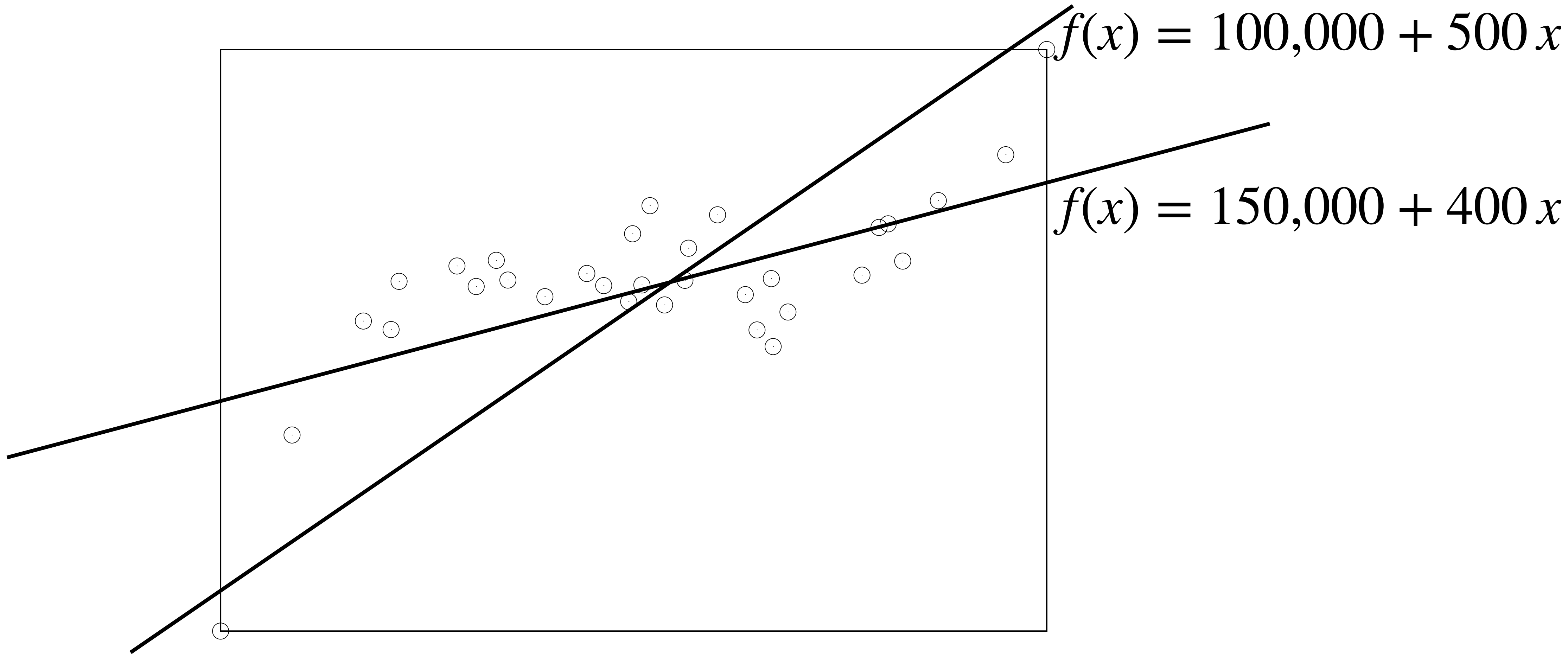
Cauchy-Schwarz inequality

- for any two vectors $x, y \in \mathbb{R}^d$, the following inequality holds

$$\underbrace{\sum_{i=1}^{d} x_i y_i}_{x^T y} \;\leq\; \underbrace{\sqrt{\sum_{i=1}^{d} x_i^2}\sqrt{\sum_{j=1}^{d} y_j^2}}_{\|x\|_2 \|y\|_2}$$

  where $\|x\|_2 = \sqrt{(x_1)^2 + (x_2)^2 + \cdots + (x_d)^2}$ is called the **Euclidean norm**, **2-norm**, or **L2-norm**, and measures the Euclidean distance from the origin to the point x

- hence, $|f(x) - f(\tilde{x})| \;\leq\; \|w\|_2 \|x - \tilde{x}\|_2$, implies that if the learned model parameter w has a small norm $\|w\|_2$, then the prediction $f(x)$ does not change too much as we change from a data point x to another data point $\tilde{x}$



Empirical risk minimization:

the process of finding a good linear predictor

Quality metric

- a **risk** or **loss function** $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ determines which model is a better fit
  - smaller $\ell(\hat{y}, y)$ indicates that $\hat{y}$ is a good approximation of y
  - typically (but not always) $\ell(y, y) = 0$ and $\ell(\hat{y}, y) \geq 0$ for all $\hat{y}, y$

- Typical choices
  - **quadratic loss:** $\ell(\hat{y}, y) = (\hat{y} - y)^2$
  - **absolute loss:** $\ell(\hat{y}, y) = |\hat{y} - y|$

Empirical risk

- How does the predictor $f(\cdot)$ fit a my data set $\{(x_i, y_i)\}_{i=1}^n$ with loss ℓ ?
  - **empirical risk** is the average loss over the data points
$$\mathscr{L} = \frac{1}{n}\sum_{i=1}^{n} \ell(\hat{y}_i, y_i) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i), y_i)$$
- if $\mathscr{L}$ is small, the predictor predicts the given data well
- When the predictor is parametrized by w, we write
$$\mathscr{L}(w) = \frac{1}{n}\sum_{i=1}^{n} \ell(f_w(x_i), y_i)$$

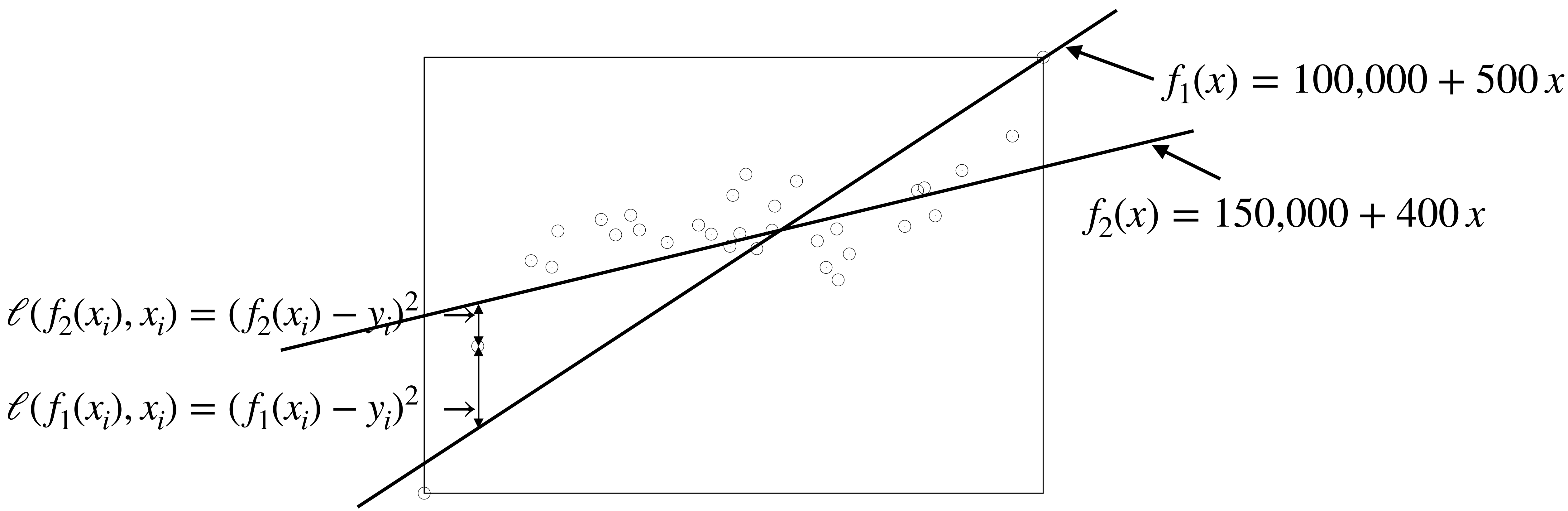
  to make the dependence on w explicit

Mean squared error

- with the most popular choice of squared loss $\ell(\hat{y}, y) = (\hat{y} - y)^2$, empirical risk is **mean-squared error (MSE)**

$$\mathscr{L}(w) \;=\; \text{MSE} \;=\; \frac{1}{n}\sum_{i=1}^{n}(f_w(x_i) - y_i)^2$$

- often we use root-mean-squared error, $\text{RMSE} = \sqrt{\text{MSE}}$, which has the same unit/scale as the outcomes y_i's



Empirical risk minimization

- Training:
    - choosing the parameter w in a parametrized predictor $f_w(x)$ is called **fitting the predictor** to data or **training the model**
    - **empirical risk minimization (ERM)** is a general method for fitting parametrized predictors
    - ERM: choose w that minimizes empirical risk $\mathscr{L}(w)$

$$\text{minimize}_w \ \ \mathscr{L}(w)$$

- algorithm: often there is no analytical solution to this minimization problem, so we use **numerical optimization**
- for the squared loss example, this is

$$\text{minimize}_w \ \ \frac{1}{n}\sum_{i=1}^{n}(f_w(x_i) - y_i)^2$$

- if loss is squared loss and $f_w(x)$ is a linear model, then it is special in the sense that it has a closed form (or analytical) solution
- This closed form solution is what we study in the rest of this chapter (the set of slides under the title regression)

Supervised Learning

Regression | Classification

- **Linear regression**
- **Nearest neighbor**
- **Decision trees**
- **Bootstrap**
- **Deep Neural Networks**

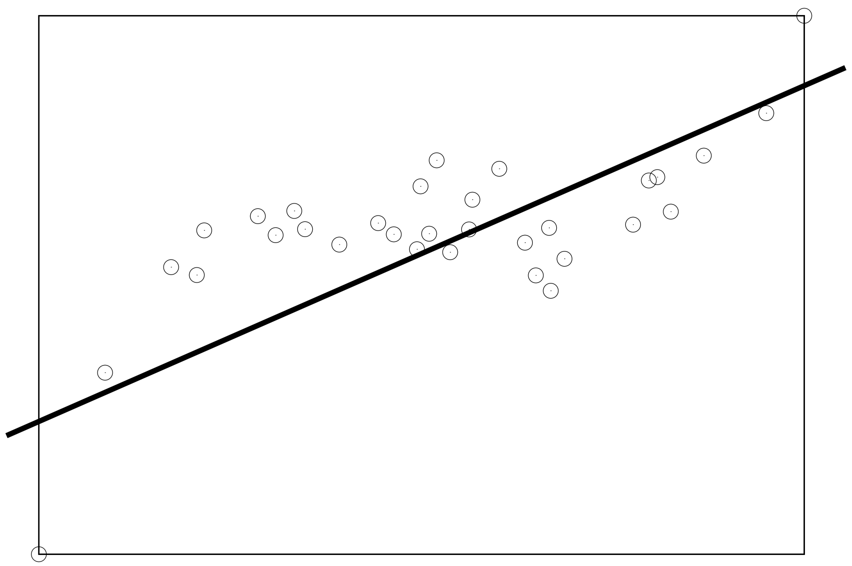
Unsupervised Learning

Least squares linear regression

with a choice of squared loss

Training a model is finding the best parameters

- Least squares linear regression
  - predictor: linear with parameter $w \in \mathbb{R}^d$
    $$\hat{y} = f_w(x) = w^T x$$
  - loss: squared loss
    $$\ell(\hat{y}, y) = (\hat{y} - y)^2$$
  - empirical risk is MSE
    $$\mathcal{L}(w) = \frac{1}{n} \sum_{i=1}^{n} (w^T x_i - y_i)^2$$



- ERM: choose model parameter w to minimize MSE
- called **linear least squares fitting** or **linear regression**
- linear regression is particularly sensitive to outliers

Least squares formulation

- express MSE in **matrix notation** as

$$\frac{1}{n}\sum_{i=1}^{n}\Big(\underbrace{(x_i)^T w}_{=w^T x_i} - y_i\Big)^2 = \boxed{\frac{1}{n}\|\mathbf{X}w - \mathbf{y}\|_2^2}$$

where **data matrix** $\mathbf{X} \in \mathbb{R}^{n\times d}$ and **outcome vector** $\mathbf{y} \in \mathbb{R}^n$ are

$$\mathbf{X} = \begin{bmatrix} (x_1)^T \\ \vdots \\ (x_n)^T \end{bmatrix}, \quad \mathbf{X}w = \begin{bmatrix} (x_1)^T w \\ \vdots \\ (x_n)^T w \end{bmatrix}$$, and $$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

and $\|y\|_2$ is a **2-norm, L_2-norm** or **Euclidean norm** of a vector such that

$$\|y\|_2 = \sqrt{\sum_{i=1}^{n}(y_i)^2}$$ and $$\|y\|_2^2 = \sum_{i=1}^{n}(y_i)^2$$

Least-squares solution

- The best parameter w is the solution to

$$\text{minmize}_w \ \|\mathbf{X}w - \mathbf{y}\|_2^2$$

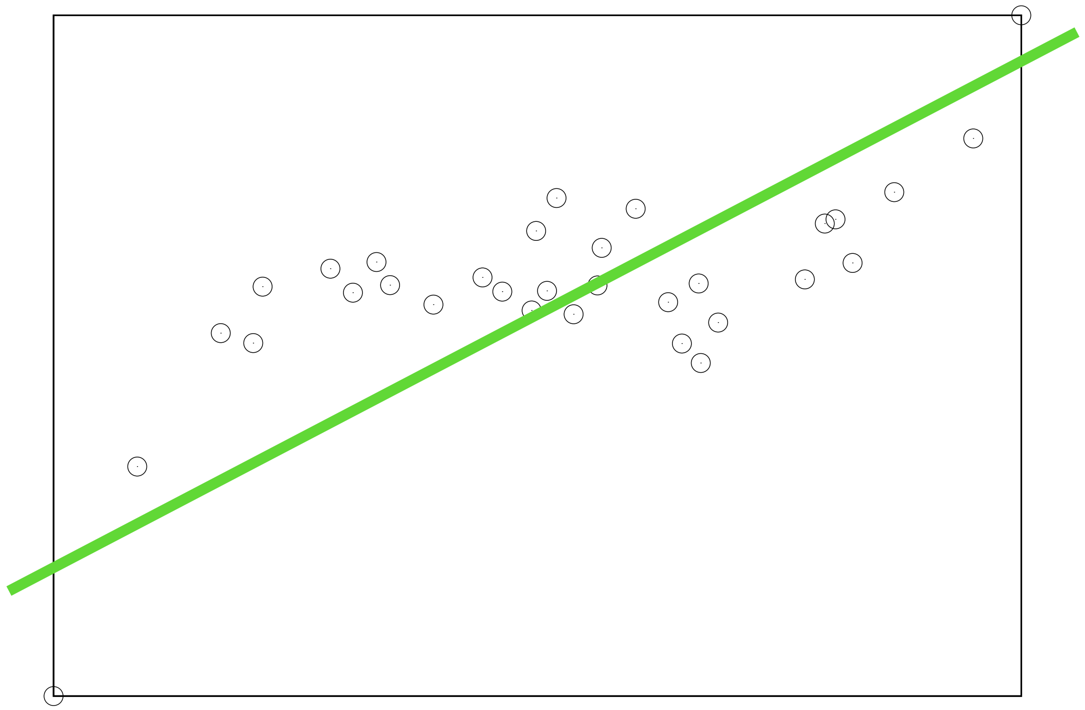
- When $\mathbf{X}$ has linearly independent columns (which implies that $\mathbf{X}$ is a tall matrix and $n \geq d$), there is a unique optimal solution

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

- optimal prediction is

$$f_{\hat{w}_{\text{LS}}}(x) = \hat{w}_{\text{LS}}^T x = \mathbf{y}^T\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-T} x$$

demo2_lin.ipynb
demo3_diabetes.ipynb



Linear models with higher order features

with human-engineered features

Linear regression with polynomial features

- polynomial feature vector $h : \mathbb{R}^d \to \mathbb{R}^k$
  for example with d=1, each feature is a monomial of the form

$$h(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^{k-1} \end{bmatrix}$$

- and the predictor is a linear function fo the polynomial features

$$\hat{y} = w^T h(x) = w_0 + w_1 x + w_2 x^2 + \cdots + w_{k-1} x^{k-1}$$

- MSE with k-dimensional w is

$$\mathscr{L}(w) \; = \; \frac{1}{n} \sum_{i=1}^{n} (w^T h(x) - y)^2$$

- in the 1-dimensional example, it is

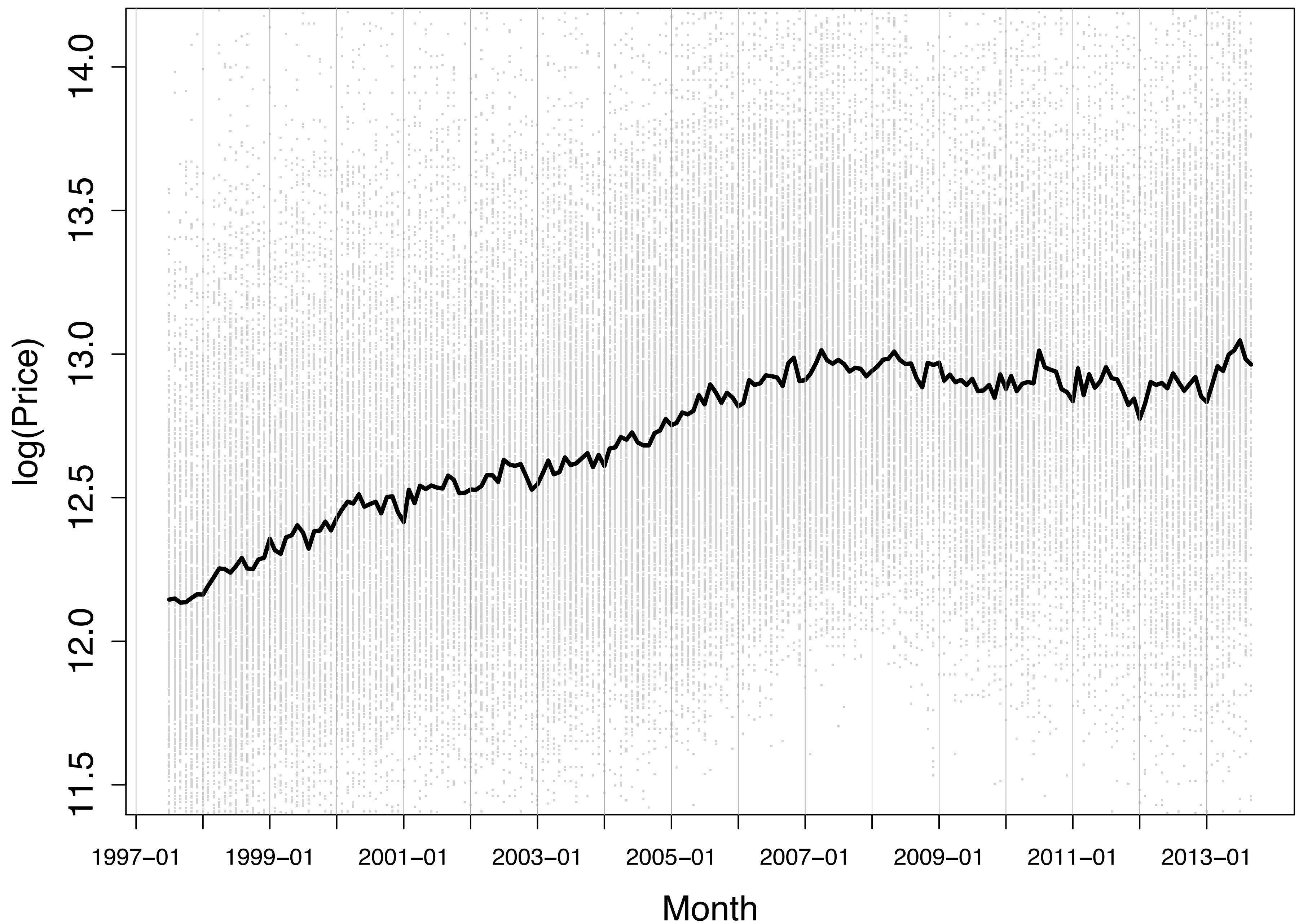
$$\mathscr{L}(w) \; = \; \frac{1}{n} \sum_{i=1}^{n} (w_0 + w_1 x + \cdots w_{k-1} x^{k-1} - y)^2$$

- but, low degree polynomials might not capture the true relations
  - domain knowledge help

demo2_lin.ipynb

Seasonal features

$$
\begin{aligned}
(x_i, y_i) &= (\text{month-year, average house price}) \\
& (\text{Jan } 2001, \$255k) \\
& (\text{Feb } 2001, \$268k) \\
& \vdots
\end{aligned}
$$



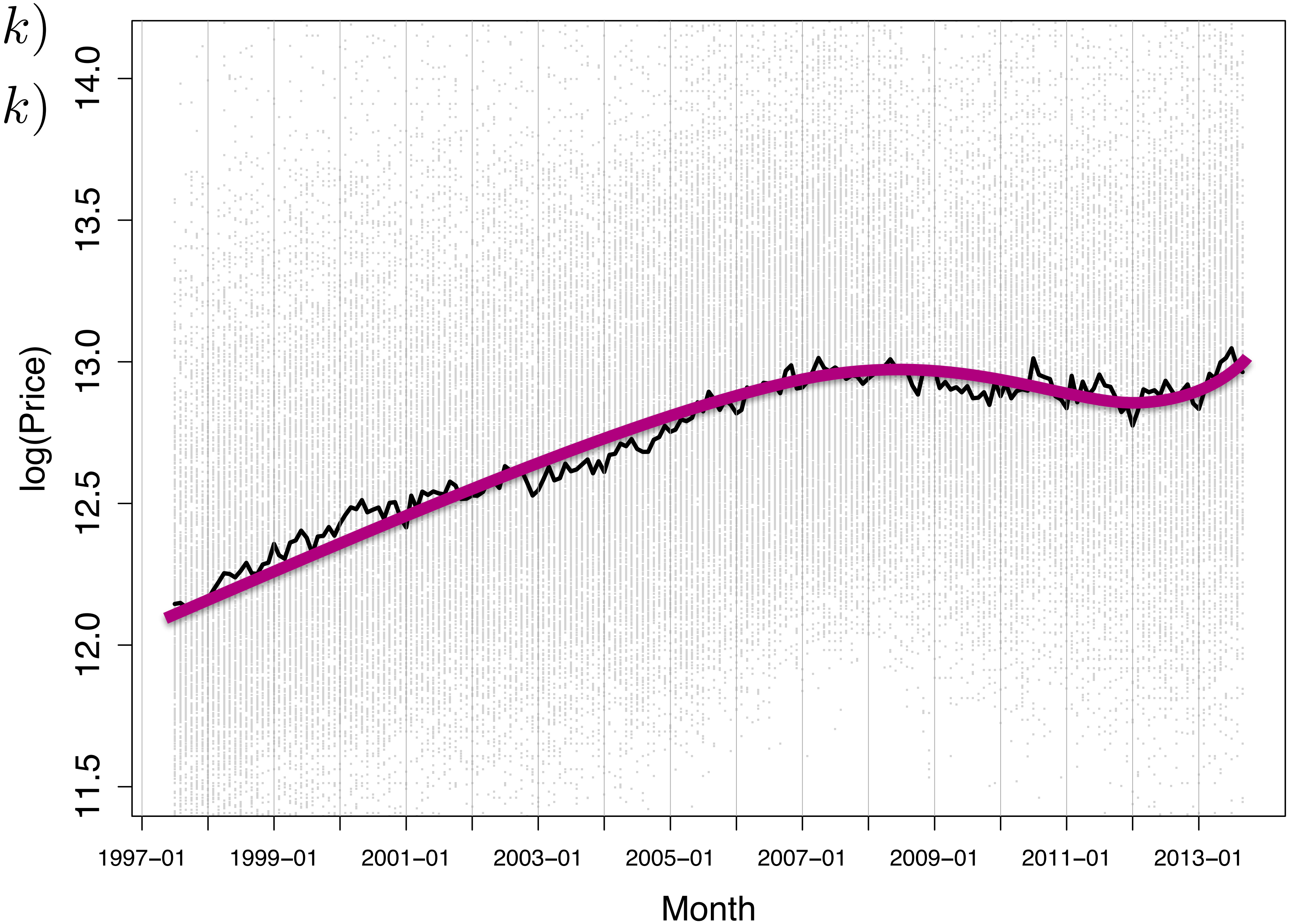
Seasonal features

$$(x_i, y_i) \ = \text{(month-year, average house price)}$$

$$(\text{Jan } 2001, \$255k)$$

$$(\text{Feb } 2001, \$268k)$$

$$\vdots$$



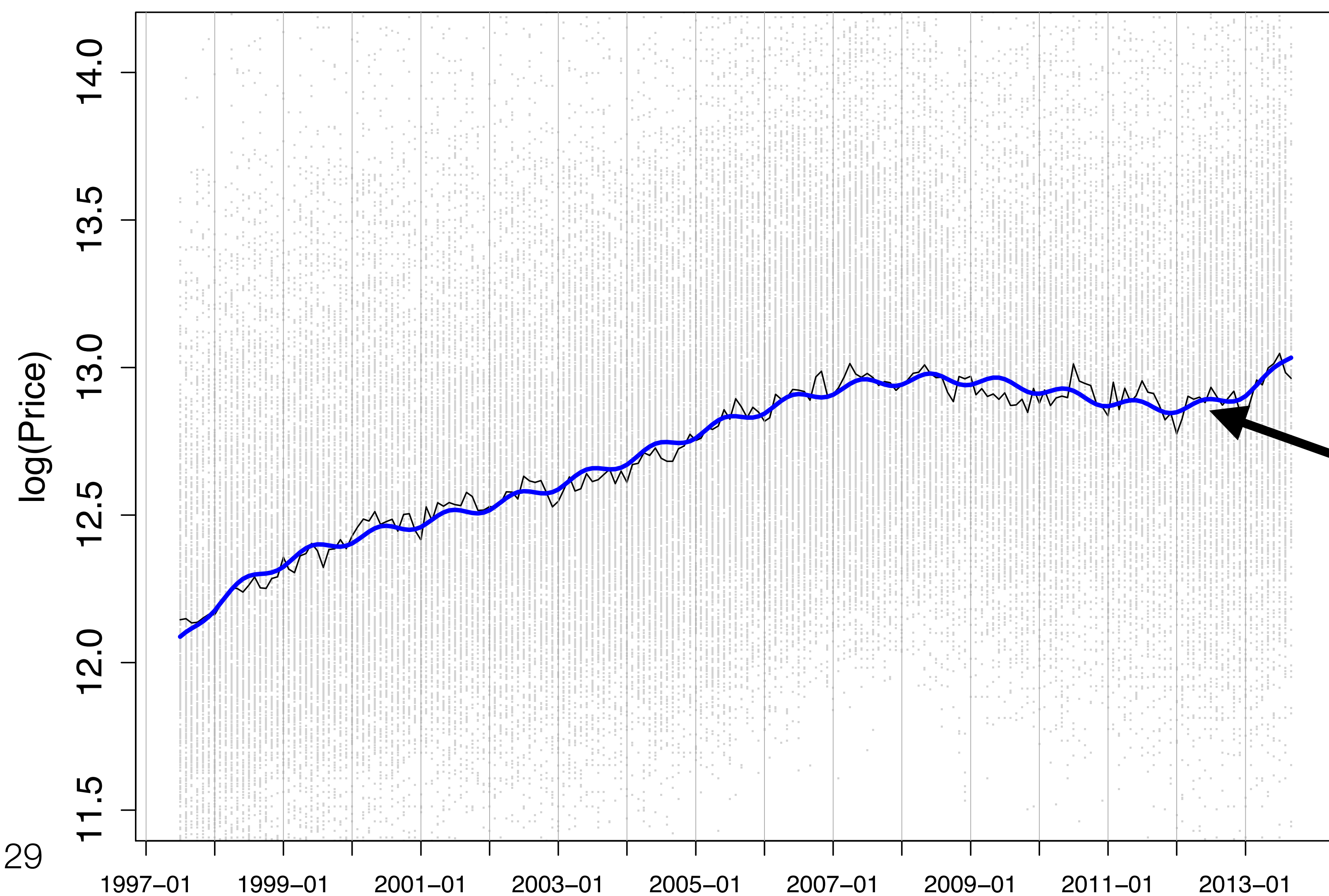
- more buyers in summer drive price higher
- but, best (low-degree) polynomial fit misses the seasonality

Seasonal features

- known relations like seasonality can be manually added as new features

magnitude → w_4, phase → w_5

$$f(x) \;=\; w_0 + w_1 x + w_2 x^2 + w_3 x^3 + {\color{red}w_4} \sin\left(\frac{2\pi x}{12} + {\color{red}w_5}\right)$$



best polynomial + sinusoidal fit
but, it is non-linear

Seasonal features

- reparametrization from a sinusoidal model to linear model

magnitude → w_4; phase → w_5

$$f(x) \;=\; w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 \sin\left(\frac{2\pi x}{12} + w_5\right)$$

$$\text{trigonometric identity} : \sin(a+b) = \sin(a)\cos(b) + \cos(a)\sin(b)$$

$$w_4 \sin\left(\frac{2\pi x}{12} + w_5\right) \;=\; \underbrace{w_4 \cos(w_5)}_{\tilde{w}_4} \sin\left(\frac{2\pi x}{12}\right) + \underbrace{w_4 \sin(w_5)}_{\tilde{w}_5} \cos\left(\frac{2\pi x}{12}\right)$$

$$f(x) \;=\; w_0 + w_1 x + w_2 x^2 + w_3 x^3 + \tilde{w}_4 \sin\left(\frac{2\pi x}{12}\right) + \tilde{w}_5 \cos\left(\frac{2\pi x}{12}\right)$$

feature 5 → $\sin\left(\frac{2\pi x}{12}\right)$; feature 6 → $\cos\left(\frac{2\pi x}{12}\right)$

- why use sinusoidal features?

Linear models with higher order features

- compact notation of the model

$$\begin{aligned} f(x) &= w_0 h_0(x) + w_1 h_1(x) + \cdots + w_D h_D(x) \\ &= w^T\, h(x) \end{aligned}$$

- vector notation of the model parameters *w* and features *h(x)*

$$w = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_D \end{bmatrix} \qquad h(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ x^3 \\ \sin(2\pi x/12) \\ \cos(2\pi x/12) \end{bmatrix}$$

- as the features are hard coded, human ingenuity/insight needed in feature engineering with domain knowledge

Modern machine learning tasks are complex

- predict “How old is this person?”

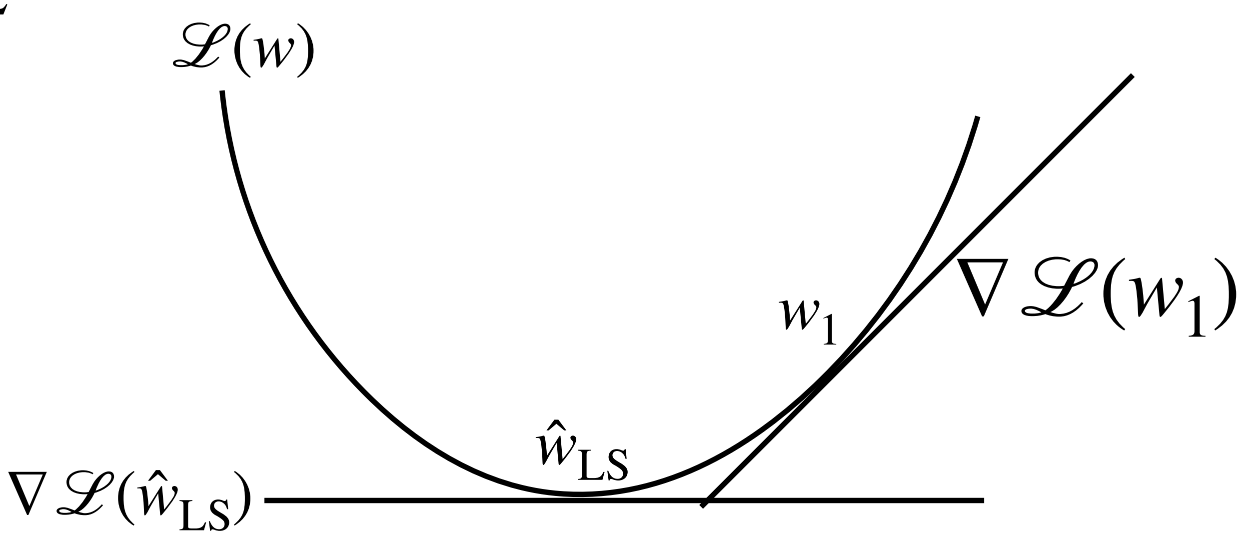
- how do we know which feature to use?

- study automated feature extraction using deep neural networks

Theoretical analysis

Least squares solution

- why is the solution of
$$\hat{w}_{\text{LS}} = \arg\min_{w} \|\mathbf{X}w - \mathbf{y}\|_2^2$$
equal to
$$\hat{w}_{\text{LS}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$



- note that $\|\mathbf{X}w - \mathbf{y}\|_2^2$ is a **strongly-convex function** when $\mathbf{X}$ has **linearly independent columns**

- minimizer of a strongly-convex function is unique, and can be found by taking the **gradient** $\nabla\mathscr{L}(w)$ and finding w such that the gradient is zero

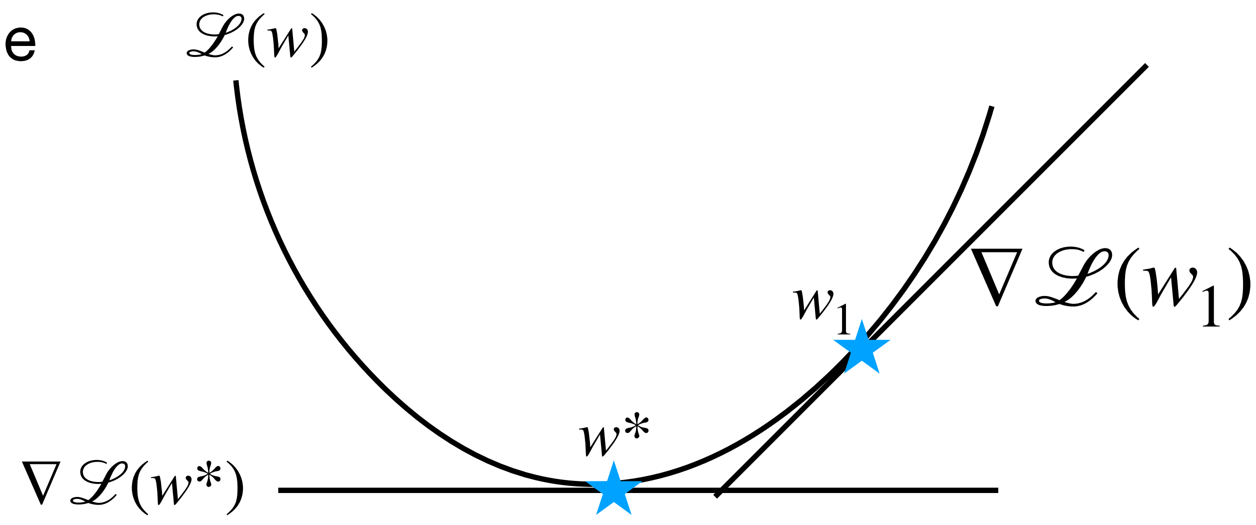
Simple example

- for 1-dimensional w, consider an example

  $$\mathcal{L}(w) = (2w - 4)^2$$

  $$\nabla \mathcal{L}(w) = \frac{\partial \mathcal{L}}{\partial w} = 4(2w - 4)$$

  setting derivative to zero, we get $w^* = 2$



- in general dimensions, let $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ be a multivariate function such that $w \mapsto \mathcal{L}(w)$
- its gradient is defined as a vector-valued function $\nabla \mathcal{L} : \mathbb{R}^d \to \mathbb{R}^d$ such that

$$\nabla \mathcal{L}(w) = \begin{bmatrix} \frac{\partial \mathcal{L}(w)}{\partial w_1} \\ \vdots \\ \frac{\partial \mathcal{L}(w)}{\partial w_d} \end{bmatrix}$$

- $\mathcal{L}(w) = 2w_1^2 - 4w_1w_2 + 3w_2^2 + 5w_1 - 3w_2 + 9$

- $\nabla \mathcal{L}(w) = \begin{bmatrix} \frac{\partial \mathcal{L}(w)}{\partial w_1} \\ \frac{\partial \mathcal{L}(w)}{\partial w_2} \end{bmatrix} = \begin{bmatrix} 4w_1 - 4w_2 + 5 \\ -4w_1 + 6w_2 - 3 \end{bmatrix}$

- setting the gradient to zero gives
  $w^* = (w_1^*, w_2^*) =$ (-9/4,-1)

Simple rules

- there is a set of simple rules that help compute the gradient of functions represented by matrix vector multiplications
- Rule 1: $\nabla(\|w\|_2^2) = 2w$
- Rule 2: $\nabla(b^T w) = b$
- Rule 3: $\nabla \mathscr{L}_w(Aw - b) = A^T \nabla_x \mathscr{L}(x)\,|_{x=Aw-b}$

- for $\mathscr{L}(w) = \|\mathbf{X}w - \mathbf{y}\|_2^2$, we claimed that $\nabla \mathscr{L}(w) = 2\mathbf{X}^T(\mathbf{X}w - \mathbf{y})$
- using above rules,
  - $\nabla \|\mathbf{X}w - \mathbf{y}\|_2^2 \underset{rule\ 3}{=} \mathbf{X}^T \nabla_z(\|z\|_2^2)\,|_{z=\mathbf{X}w-\mathbf{y}} \underset{rule\ 1}{=} \mathbf{X}^T 2(\mathbf{X}w - \mathbf{y})$
- this gives

$$\nabla \mathscr{L}(w) = 2\mathbf{X}^T(\mathbf{X}w - \mathbf{y})$$

Alternative derivation via summation notation

- let $\mathscr{L} : \mathbb{R}^d \to \mathbb{R}$ be a multivariate function $w \mapsto \mathscr{L}(w)$
- Its gradient is defined as a vector-valued function $\nabla \mathscr{L} : \mathbb{R}^d \to \mathbb{R}^d$ such that

$$\nabla \mathscr{L}(w) = \begin{bmatrix} \frac{\partial \mathscr{L}(w)}{\partial w_1} \\ \vdots \\ \frac{\partial \mathscr{L}(w)}{\partial w_d} \end{bmatrix}$$

- for $\mathscr{L}(w) = \|\mathbf{X}w - \mathbf{y}\|_2^2$, we have $\nabla \mathscr{L}(w) = 2\mathbf{X}^T(\mathbf{X}w - \mathbf{y})$

  which follows from $\|\mathbf{X}w - \mathbf{y}\|_2^2 = \sum_{i=1}^{n} (x_i^T w - y_i)^2 = \sum_{i=1}^{n} \Big(\{ \sum_{j=1}^{d} x_i[j] w_j \} - y_i \Big)^2$

- $$\frac{\mathscr{L}(w)}{\partial w_k} = \sum_{i=1}^{n} \frac{\partial \Big(\sum_{j=1}^{d} x_i[j] w_j - y_i \Big)^2}{\partial w_k} = \sum_{i=1}^{n} 2 x_i[k] \Big(\sum_{j=1}^{d} x_i[j] w_j - y_i \Big)$$

$$= 2 \sum_{i=1}^{n} x_i[k] (x_i^T w - y_i) = 2 \underbrace{\mathbf{X}[k, :]}_{\text{k-th row}} (\mathbf{X}w - \mathbf{y})$$

Once we have the gradient,

- for $\mathscr{L}(w) = \|\mathbf{X}w - \mathbf{y}\|_2^2$, we have
$$\nabla \mathscr{L}(w) = 2\mathbf{X}^T(\mathbf{X}w - \mathbf{y})$$

- hence, setting the gradient to zero, $2\mathbf{X}^T(\mathbf{X}w - \mathbf{y}) = 0$, we get
$$\mathbf{X}^T\mathbf{X}w = \mathbf{X}^T\mathbf{y}$$

- when $\mathbf{X}$ has a full row rank (i.e. when the columns of $\mathbf{X}$ are linearly independent, $\mathbf{X}^T\mathbf{X}$ is invertible

- this gives
$$w = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

- this is the optimal solution $\hat{w}_{\mathrm{LS}}$ we have been using

Two schools of thoughts

- machine learning

Belongs to a set of of functions
(to be defined by the statistician)

- given $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, find a predictor $f \in \mathcal{F}$

This could be the set of all degree-3 polynomial functions,
if we use degee-3 polynomial features and linear regression

- any machine learning algorithm can be derived from

Empirical Risk Minimization

$$y \simeq f_0(x)$$

with a given loss function ℓ

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \ell(f(x_i), y_i)$$

Maximum Likelihood Estimator

$$y = f_0(x) + \varepsilon$$

with known pdf of ε

$$\max_{f \in \mathcal{F}} \prod_{i=1}^{n} P(y_i = f(x_i) + \varepsilon)$$

Probabilistic interpretation of least squares

- given data $\{(x_i, y_i)\}_{i=1}^n$ and a **probabilistic model** with parameters w and σ^2,

$$y_i = w^T x_i + \varepsilon_i \ ,$$

with $\varepsilon_i \sim \mathcal{N}(0,\sigma^2)$ distributed as i.i.d. Gaussian with zero mean and variance σ^2

- recall pdf of Gaussian distribution is

$$\mathbf{P}(z) \ = \ \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{\sigma^2} x_2}$$

- the **log-likelihood** of a data point (x_i, y_i) is defined as

$$\log\big(\mathbf{P}(y_i - w^T x_i)\big) \ = -\frac{1}{\sigma^2}(w^T x_i - y_i)^2 - \frac{1}{2}\log(2\pi\sigma^2)$$

- the log-likelihood of the dataset is

$$\sum_{i=1}^{n} \log\Big(\mathbf{P}(y_i - w^T x_i)\Big) \;=\; \sum_{i=1}^{n} \Big\{ -\frac{1}{\sigma^2}\|w^T x_i - y_i\|_2^2 - \frac{d}{2}\log(2\pi\sigma^2) \Big\}$$

- **maximum likelihood estimation (MLE)** is an algorithm that outputs a model parameter $w \in \mathbb{R}^d$ such that it maximizes the log-likelihood:

$$\begin{aligned}
\hat{w}_{\text{MLE}} &= \arg\max_w \sum_{i=1}^{n} \Big\{ -\frac{1}{\sigma^2}(w^T x_i - y_i)^2 - \frac{1}{2}\log(2\pi\sigma^2) \Big\} \\
&= \arg\min_w \sum_{i=1}^{n} \Big\{ (w^T x_i - y_i)^2 \Big\} \\
&= \arg\min_w \|\mathbf{X}w - \mathbf{y}\|_2^2
\end{aligned}$$