Linear Regression

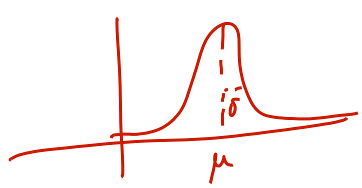
Machine Learning – CSE546
Carlos Guestrin
University of Washington

September 30, 2014

What about continuous variables?

- Billionaire says: If I am measuring a continuous variable, what can you do for me?
- **You say: Let me tell you about Gaussians…**

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



Some properties of Gaussians

- affine transformation (multiplying by scalar and adding a constant)
  - X ~ $N(\mu,\sigma^2)$
  - Y = aX + b ➔ Y ~ $N(a\mu+b,a^2\sigma^2)$
- Sum of Gaussians
  - X ~ $N(\mu_X,\sigma^2_X)$
  - Y ~ $N(\mu_Y,\sigma^2_Y)$
  - Z = X+Y ➔ Z ~ $N(\mu_X+\mu_Y, \sigma^2_X+\sigma^2_Y)$

Learning a Gaussian

HW scores

x_1 98
96
85
x_N

- Collect a bunch of data
  - Hopefully, i.i.d. samples
  - e.g., exam scores
- Learn parameters
  - Mean
  - Variance

$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i$

why?

MLE

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

MLE for Gaussian

- Prob. of i.i.d. samples $D=\{x_1,\ldots,x_N\}$:

$$P(\mathcal{D} \mid \mu, \sigma) \overset{iid}{=} \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^N \prod_{i=1}^N e^{\frac{-(x_i-\mu)^2}{2\sigma^2}}$$

$$\hat{\mu}_{MLE}, \hat{\sigma}^2_{MLE} = \arg\max_{\mu,\sigma} P(D|\mu,\sigma) = \arg\max_{\mu,\sigma} \ln P(D|\mu,\sigma)$$

- Log-likelihood of data:

$$\begin{aligned} \ln P(\mathcal{D} \mid \mu, \sigma) &= \ln\left[\left(\frac{1}{\sigma\sqrt{2\pi}}\right)^N \prod_{i=1}^N e^{\frac{-(x_i-\mu)^2}{2\sigma^2}}\right] \\ \max_{\mu,\sigma} \quad &= -N \ln \sigma\sqrt{2\pi} - \sum_{i=1}^N \frac{(x_i-\mu)^2}{2\sigma^2} \end{aligned}$$

Your second learning algorithm: MLE for mean of a Gaussian

- What's MLE for mean?

$$0 = \frac{d}{d\mu} \ln P(\mathcal{D} \mid \mu, \sigma) = \frac{d}{d\mu}\left[-N \cancel{\ln \sigma\sqrt{2\pi}}^{\,0} - \sum_{i=1}^N \frac{(x_i-\mu)^2}{2\sigma^2}\right]$$

$$\to -\sum_{i=1}^N \frac{d}{d\mu} \frac{(x_i-\mu)^2}{2\sigma^2} = -\sum_{i=1}^N \frac{-x_i+\mu}{\cancel{\sigma^2}} = 0$$

$$N\mu = \sum_{i=1}^N x_i \Rightarrow \hat{\mu} = \frac{\sum_{i=1}^N x_i}{N}$$

MLE for variance

- Again, set derivative to zero:

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

$$= \frac{d}{d\sigma}\left[-N\ln\sigma\sqrt{2\pi}\right] - \sum_{i=1}^{N}\frac{d}{d\sigma}\left[\frac{(x_i-\mu)^2}{2\sigma^2}\right]$$

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

Learning Gaussian parameters

- MLE:

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

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

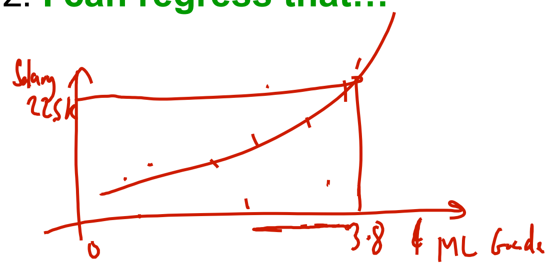
Now you can justify

- BTW. MLE for the variance of a Gaussian is **biased**
  - Expected result of estimation is **not** true parameter!
  - Unbiased variance estimator:

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

Prediction of continuous variables

- Billionaire sayz: Wait, that's not what I meant!
- You sayz: Chill out, dude.
- He sayz: I want to predict a continuous variable for continuous inputs: I want to predict salaries from GPA.
- You sayz: **I can regress that…**



linear

The regression problem

$\langle 3.9, 255k \rangle$

input target

- **Instances:** $\langle \mathbf{x}_j, t_j \rangle$ $f: X \rightarrow \mathbb{R}$
- **Learn:** Mapping from x to t(**x**)
- **Hypothesis space:** polynomials
  - Given, basis functions $H = \{h_1, \ldots, h_K\}$ non-linear
  - Find coeffs **w**=$\{w_1, \ldots, w_k\}$ $\underbrace{t(\mathbf{x})}_{\text{data}} \approx \widehat{f}(\mathbf{x}) = \sum_{i=1}^{K} w_i h_i(\mathbf{x})$
  - Why is this called linear regression???
    - model is linear in the parameters

loss fn

- Precisely, minimize the residual squared error: squared

target estimate

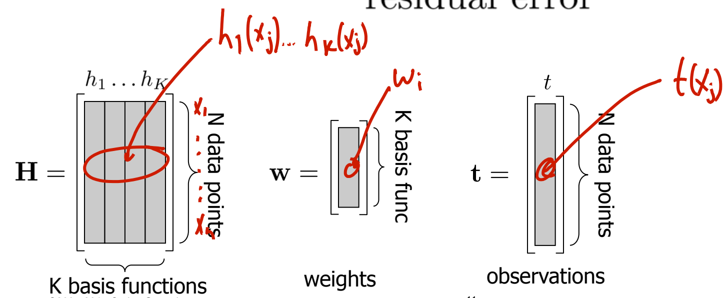
$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{j=1}^{N} \left(t(\mathbf{x}_j) - \sum_{i=1}^{K} w_i h_i(\mathbf{x}_j) \right)^2$$

The regression problem in matrix notation

$$\underbrace{\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j)\right)^2}_{\text{same}}$$

how do we min?

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \underbrace{(\mathbf{H}\mathbf{w} - \mathbf{t})^T(\mathbf{H}\mathbf{w} - \mathbf{t})}_{\text{residual error}}$$



Minimizing the Residual

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \overbrace{\underbrace{(\mathbf{H}\mathbf{w} - \mathbf{t})^T(\mathbf{H}\mathbf{w} - \mathbf{t})}_{\text{residual error}}}^{F(w)}$$

$$\nabla_w F(w) = 0 \Rightarrow \cancel{2} H^T(Hw - t) = 0$$

$$\Rightarrow H^T H w - H^T t = 0$$

$$\Rightarrow w^* = (H^T H)^{-1} H^T t$$

in scalar calculus.

$$\frac{d}{dw}\left[(\alpha w - t)(\alpha w - t)\right] = 2\alpha(\alpha w - t)$$

in matrix:

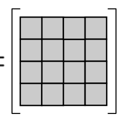
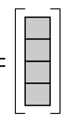
$$\nabla_w\left[(Hw-t)^T(Hw-t)\right] = 2H^T(Hw-t)$$

!!

Regression solution = simple matrix operations

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \underbrace{(\mathbf{H}\mathbf{w}-\mathbf{t})^T(\mathbf{H}\mathbf{w}-\mathbf{t})}_{\text{residual error}}$$

$$\text{solution: } \mathbf{w}^* = \underbrace{(\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}}_{\mathbf{A}^{-1}}\underbrace{\mathbf{H}^{\mathrm{T}}\mathbf{t}}_{\mathbf{b}} = \underline{\mathbf{A}^{-1}\mathbf{b}}$$

where $\mathbf{A} = \mathbf{H}^{\mathrm{T}}\mathbf{H} =$  $\mathbf{b} = \mathbf{H}^{\mathrm{T}}\mathbf{t} =$ 

k×k matrix for k basis functions

k×1 vector

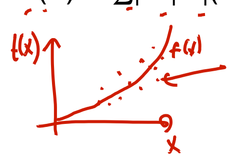
But, why?

- Billionaire (again) says: Why sum squared error???
- You say: Gaussians, Dr. Gateson, Gaussians…

- Model: prediction is linear function plus Gaussian noise
  - t(**x**) = $\sum_i w_i h_i(\mathbf{x}) + \varepsilon_{\mathbf{x}}$

$\varepsilon_x \overset{iid}{\sim} N(0, \sigma^2)$

$t(x) \overset{iid}{\sim} N(\sum w_i h_i(x), \sigma^2)$



- Learn **w** using MLE

params of mean fn.

noise variance

target

$$P(t \mid \mathbf{x}, \mathbf{w}, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-[t-\overbrace{\sum_i w_i h_i(\mathbf{x})}^{\text{mean}}]^2}{2\sigma^2}}$$

input

$\arg\min_w \text{const} + f(w) = \arg\min_w f(w)$

Maximizing log-likelihood

$\ln \prod = \sum \ln$

$\ln e^{\text{smth}} = \text{smth}$

Maximize:

$$\arg\max_{w} \ln P(\mathcal{D} \mid \mathbf{w}, \sigma) = \underbrace{\ln \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^N}_{\text{doesn't depend on } w} \prod_{j=1}^{N} e^{\frac{-[\overset{f(x_j)}{t_j} - \sum_i w_i h_i(\mathbf{x}_j)]^2}{2\sigma^2}}$$

$$= \arg\max_w \left[\ln(\cdot) - \sum_{j=1}^{N} \frac{[t(x_j) - \sum_i w_i h_i(x_j)]^2}{2\sigma^2} \right]$$

$$= \arg\max_w - \sum_{j=1}^{N} \frac{[t(x_j) - \sum_i w_i h_i(x_j)]^2}{2\sigma^2}$$

$$= \arg\min_w \sum_{j=1}^{N} [t(x_j) - \sum_i w_i h_i(x_j)]^2$$

$\arg\max_w -f(w) = \arg\min_w f(w)$

$\arg\min_w \frac{f(w)}{2\sigma^2} = \arg\min_w f(w)$

Least-squares Linear Regression is MLE for Gaussians!!!

Announcements

- Go to recitation!! ☺
  - Wednesday, 5pm in EEB 045
- First homework will go out today
  - Due on October 14
  - Start early!!

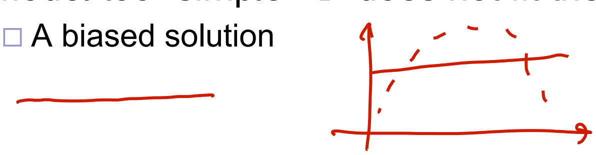
Bias-Variance Tradeoff

Machine Learning – CSE546
Carlos Guestrin
University of Washington

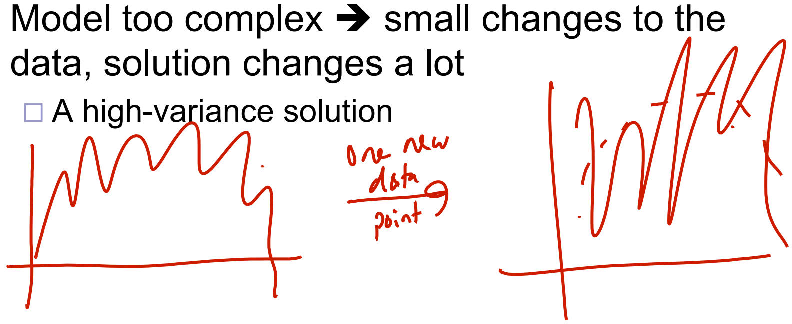
September 30, 2014

Bias-Variance tradeoff – Intuition

- Model too "simple" ➔ does not fit the data well
  - A biased solution



- Model too complex ➔ small changes to the data, solution changes a lot
  - A high-variance solution



(Squared) Bias of learner

from $D \to$ learn h_D

- Given dataset *D* with *N* samples, learn function $h_D(x)$
- If you sample a different dataset *D'* with *N* samples, you will learn different $h_{D'}(x)$
- **Expected hypothesis**: $E_D[h_D(x)] = \bar{h}_N$

  what I expect to learn

- **Bias:** difference between what you expect to learn and truth
  - Measures how well you expect to represent true solution
  - Decreases with more complex model
  - Bias^2 at one point *x*: $(f(x) - \bar{h}_N(x))^2$
  - Average Bias^2: $E_x[(f(x) - \bar{h}_N(x))^2]$

Variance of learner

$D \to h_D$

- Given dataset *D* with *N* samples, learn function $h_D(x)$
- If you sample a different dataset *D'* with *N* samples, you will learn different $h_{D'}(x)$
- **Variance:** difference between what you expect to learn and what you learn from a particular dataset
  - Measures how sensitive learner is to specific dataset
  - Decreases with simpler model
  - Variance at one point *x*: $E_{D,x}(h_D(x) - \bar{h}_N(x))^2$
  - Average variance:

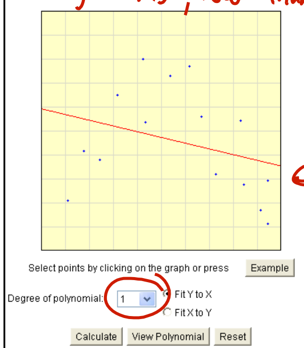
actually learn

expect to learn

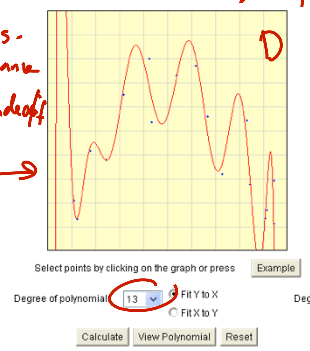
Bias-Variance Tradeoff

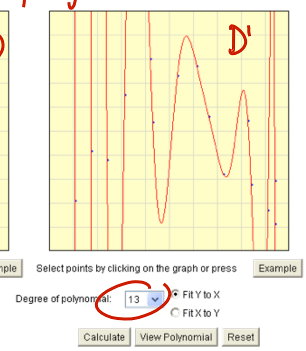
- Choice of hypothesis class introduces learning bias
  - More complex class → less bias
  - More complex class → more variance

high bias, low variance | low bias, high variance



bias-variance tradeoff





variance $= E[(x-\mu)^2]$

Bias-Variance Decomposition of Error

$$\bar{h}_N(x) = E_D[h_D(x)]$$

- Expected mean squared error: $\text{MSE} = E_D\left[E_x\left[(\underset{\text{truth}}{t(x)} - \underset{\text{estimate}}{h_D(x)})^2\right]\right]$ (S fixed)
- To simplify derivation, drop x: $E_D[(t-h_D)^2]$
- Expanding the square: Add & subtract $\bar{h}_N$

$\text{MSE} = E_D[(t - \bar{h}_N + \bar{h}_N - h_D)^2]$

$= \underbrace{E_D[(t-\bar{h}_N)^2]}_{\text{bias}^2} + \underbrace{E_D[(\bar{h}_N - h_D)^2]}_{\text{variance}} + 2E_D[(t-\bar{h}_N)(\bar{h}_N - h_D)]$ 0

play with this at home, hint: $\bar{h}_N = E_D[h_D]$ all other terms const.

Moral of the Story: Bias-Variance Tradeoff Key in ML

- Error can be decomposed:

$$\underbrace{\text{MSE}}_{\text{I want to minimize}} = E_D\left[E_x\left[(t(x) - h_D(x))^2\right]\right]$$

$$= \underbrace{E_x\left[\left(t(x) - \bar{h}_N(x)\right)^2\right]}_{\text{bias}} + \underbrace{E_D\left[E_x\left[\left(\bar{h}(x) - h_D(x)\right)^2\right]\right]}_{\text{variance}}$$

- Choice of hypothesis class introduces learning bias
  - More complex class → less bias
  - More complex class → more variance

What you need to know

- Regression
  - Basis function = features
  - Optimizing sum squared error
  - Relationship between regression and Gaussians
- Bias-variance trade-off
- Play with Applet

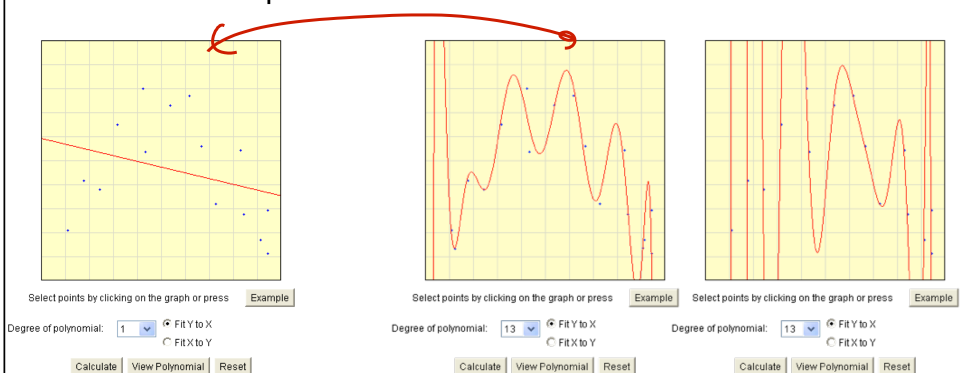
Overfitting

Machine Learning – CSE546
Carlos Guestrin
University of Washington

September 30, 2014

Bias-Variance Tradeoff

- Choice of hypothesis class introduces learning bias
  - More complex class → less bias
  - More complex class → more variance



Training set error

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$

- Given a dataset (Training data)
- Choose a loss function
  - e.g., squared error (L_2) for regression
- **Training set error:** For a particular set of parameters, loss function on training data:

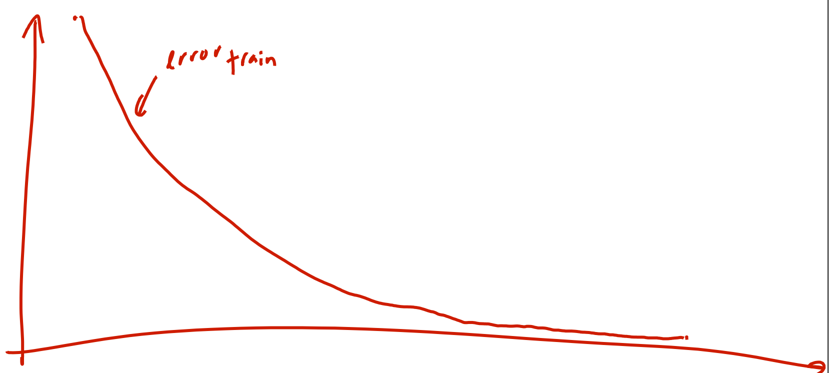
$$error_{train}(\mathbf{w}) = \frac{1}{N_{train}} \sum_{j=1}^{N_{train}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$

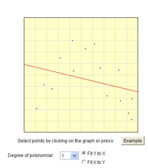
truth

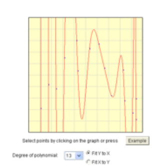
estimate

Training set error as a function of model complexity

$$error_{train}(\mathbf{w}) = \frac{1}{N_{train}} \sum_{j=1}^{N_{train}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$







Prediction error

- Training set error can be poor measure of "quality" of solution
- **Prediction error:** We really care about error over all possible input points, not just training data:

$$
\begin{aligned}
error_{true}(\mathbf{w}) &= E_{\mathbf{x}}\left[\left(t(\mathbf{x}) - \sum_i w_i h_i(\mathbf{x})\right)^2\right] \\
&= \int_{\mathbf{x}} \left(t(\mathbf{x}) - \sum_i w_i h_i(\mathbf{x})\right)^2 p(\mathbf{x}) d\mathbf{x}
\end{aligned}
$$

Prediction error as a function of model complexity

$$error_{train}(\mathbf{w}) = \frac{1}{N_{train}} \sum_{j=1}^{N_{train}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j)\right)^2$$

$$error_{true}(\mathbf{w}) = \int_{\mathbf{x}} \left(t(\mathbf{x}) - \sum_i w_i h_i(\mathbf{x})\right)^2 p(\mathbf{x}) d\mathbf{x}$$