Bayesian Methods

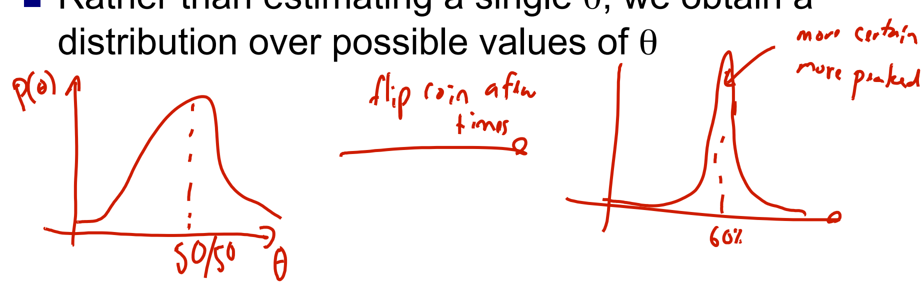
Machine Learning – CSE546

Carlos Guestrin
University of Washington

September 30, 2013

What about prior

- Billionaire says: Wait, I know that the thumbtack is "close" to 50-50. What can you do for me now?
- **You say: I can learn it the Bayesian way…**
- Rather than estimating a single θ, we obtain a distribution over possible values of θ



Bayesian Learning

data likelihood
D=(3H, 2T)
prior

- Use Bayes rule:

posterior

$$P(\theta \mid \mathcal{D}) = \frac{P(\mathcal{D} \mid \theta)P(\theta)}{P(\mathcal{D})}$$

normalizer

- Or equivalently:

unnormalized version

$$P(\theta \mid \mathcal{D}) \propto P(\mathcal{D} \mid \theta)P(\theta)$$

MLE:
$\max_{\theta} P(D|\theta)$
$\Rightarrow \hat{\theta}_{MLE}$

Bayesian: distribution $P(\theta|D)$
when $P(\theta)$ is uniform
$P(\theta|D) \propto P(D|\theta)$

Bayesian Learning for Thumbtack

beta binomial beta

$$P(\theta \mid \mathcal{D}) \propto P(\mathcal{D} \mid \theta)P(\theta)$$

- Likelihood function is simply Binomial:

$$P(\mathcal{D} \mid \theta) = \theta^{\alpha_H}(1-\theta)^{\alpha_T}$$

- What about prior? philosophy
  - Represent expert knowledge
  - Simple posterior form
- Conjugate priors:
  - Closed-form representation of posterior
  - **For Binomial, conjugate prior is Beta distribution**

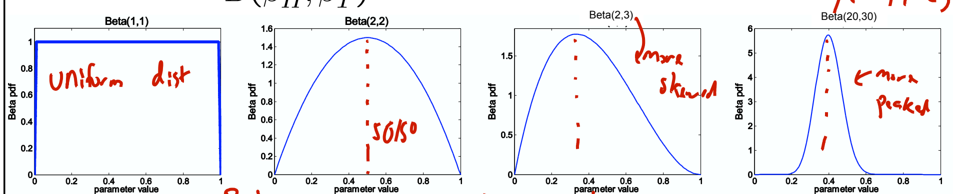
$B(\cdot,\cdot) \leftarrow$ is the beta function $\equiv$ normalizer

Beta prior distribution – P(θ)

$$P(\theta) = \frac{\theta^{\beta_H-1}(1-\theta)^{\beta_T-1}}{B(\beta_H,\beta_T)} \sim Beta(\beta_H,\beta_T)$$

Mean: $\beta_H/(\beta_H+\beta_T)$

Mode: $(\beta_H-1)/(\beta_H+\beta_T-2)$



Beta is conjugate to binomial

- Likelihood function: $P(\mathcal{D} \mid \theta) = \theta^{\alpha_H}(1-\theta)^{\alpha_T}$
- Posterior: $P(\theta \mid \mathcal{D}) \propto P(\mathcal{D} \mid \theta)P(\theta)$ ← $Beta(\beta_H,\beta_T)$

$$P(\theta \mid D=(\alpha_H,\alpha_T)) \propto P(D\mid\theta)\,P(\theta) = \theta^{\alpha_H}(1-\theta)^{\alpha_T}\,\frac{\theta^{\beta_H-1}(1-\theta)^{\beta_T-1}}{B(\beta_H,\beta_T)}$$

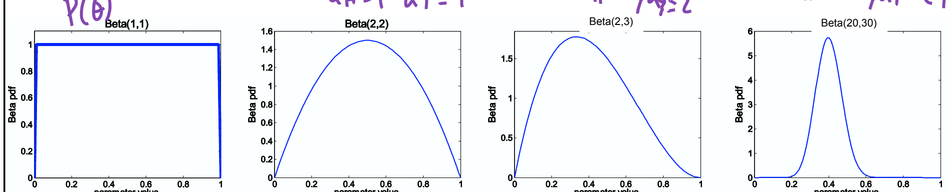
← doesn't depend on θ

$$\propto \theta^{\alpha_H+\beta_H-1}(1-\theta)^{\alpha_T+\beta_T-1} = Beta(\alpha_H+\beta_H, \alpha_T+\beta_T)$$

Posterior distribution

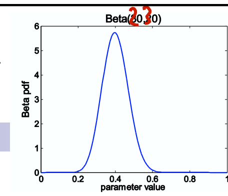
- Prior: $Beta(\beta_H,\beta_T)$
- Data: α_H heads and α_T tails
- Posterior distribution:

$$P(\theta \mid \mathcal{D}) \sim Beta(\beta_H+\alpha_H, \beta_T+\alpha_T)$$



Bayesian updates

Using Bayesian posterior



- Posterior distribution:

$$P(\theta \mid \mathcal{D}) \sim Beta(\beta_H + \alpha_H, \beta_T + \alpha_T)$$

- Bayesian inference:
  - No longer single parameter:

integrals can be hard to compute

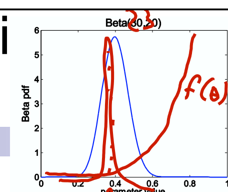
← how much money you'll make

$$E[f(\theta)] = \int_0^1 f(\theta) P(\theta \mid \mathcal{D}) d\theta$$

average over possible parameters

  - Integral is often hard to compute

MAP: Maximum a posteriori approximation



MAP can be arbitrarily bad approx.

$$P(\theta \mid \mathcal{D}) \sim Beta(\beta_H + \alpha_H, \beta_T + \alpha_T)$$

$$E[f(\theta)] = \int_0^1 f(\theta) P(\theta \mid \mathcal{D}) d\theta$$

- As more data is observed, Beta is more certain

- MAP: use most likely parameter:

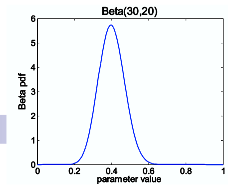
$$\hat{\theta}_{MAP} = \arg\max_{\theta} P(\theta \mid \mathcal{D}) \qquad E[f(\theta)] \approx f(\hat{\theta}_{MAP})$$

possible f:

simplest $f(\theta) = \theta$ ⇒ $E[f(\theta)] = E[\theta]$ = mean θ

$f(\theta)$ = returns on a bet if thumb tack = θ

MAP for Beta distribution



$$P(\theta \mid \mathcal{D}) = \frac{\theta^{\beta_H+\alpha_H-1}(1-\theta)^{\beta_T+\alpha_T-1}}{B(\beta_H+\alpha_H, \beta_T+\alpha_T)} \sim Beta(\beta_H+\alpha_H, \beta_T+\alpha_T)$$

$(\beta_H-1, \beta_T-1) \leftarrow$ fake coin flips over MLE

- MAP: use most likely parameter:

$$\widehat{\theta}_{MAP} = \arg\max_\theta P(\theta \mid \mathcal{D}) = \frac{\beta_H+\alpha_H-1}{\beta_H+\alpha_H+\beta_T+\alpha_T-2}$$

$\widehat{\theta}_{MLE} = \alpha_H / (\alpha_H+\alpha_T)$

- Beta prior equivalent to extra thumbtack flips
- As $N \to +\infty$, prior is "forgotten" α_H, α_T dominate β_H, β_T
- **But, for small sample size, prior is important!**

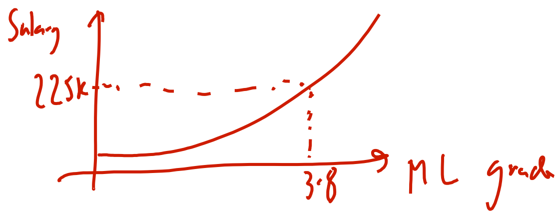
Linear Regression

Machine Learning – CSE546
Carlos Guestrin
University of Washington

September 30, 2013

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. and ML grade
- You sayz: **I can regress that…**



The regression problem <3.9, 225k>

- **Instances:** <$\mathbf{x}_j$, t_j>
- **Learn:** Mapping from x to t($\mathbf{x}$) $\hat{f}(x) \to f(x)$
- **Hypothesis space:** non-linear
  - Given, basis functions features $H = \{h_1, \ldots, h_K\}$
  - Find coeffs $\mathbf{w}=\{w_1,\ldots,w_k\}$ $\underbrace{t(\mathbf{x})}_{\text{data}} \approx \widehat{f}(\mathbf{x}) = \sum_i w_i h_i(\mathbf{x})$??
  - Why is this called linear regression???
    - model is linear in the parameters

loss function / the risk

- Precisely, minimize the ~~residual~~ sum squared error: estimate

target

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

squared

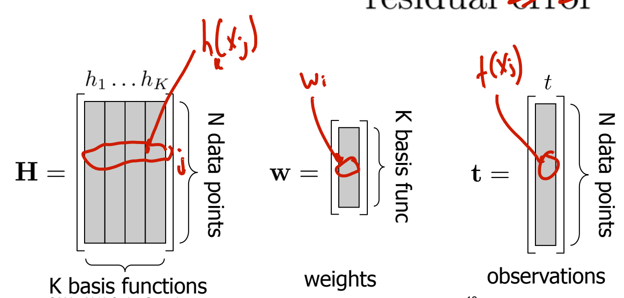
The regression problem in matrix notation

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

⇓ same

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

how do we min?



Minimizing the Residual

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

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

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

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

yed!! ☺

in scalar calculus

$$\frac{\partial}{\partial w} (\alpha w - t)(\alpha w - t)$$

$$= 2\alpha(\alpha w - t)$$

in matrix

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

$$= 2 H^T (Hw - t)$$

Regression solution = simple matrix operations

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

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

where $\mathbf{A} = \mathbf{H}^{\mathrm{T}}\mathbf{H} = \underbrace{\left[\begin{array}{cccc} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{array}\right]}_{}$ $\quad \mathbf{b} = \mathbf{H}^{\mathrm{T}}\mathbf{t} = \underbrace{\left[\begin{array}{c} \square \\ \square \\ \square \\ \square \end{array}\right]}_{}$

$N < k$

$k(H^T H)_k$ is non invertible

k×k matrix
for k basis functions

k×1 vector

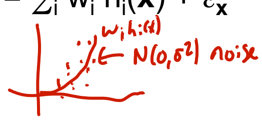
But, why?

$N(\mu, \sigma^2) \leftarrow$ Gaussian mean μ
variance σ^2

- Billionaire (again) says: Why sum squared error???
- You say: Gaussians, Dr. Gateson, Gaussians…
- Model: prediction is linear function plus Gaussian noise
  - $t(\mathbf{x}) = \sum_i w_i h_i(\mathbf{x}) + \varepsilon_{\mathbf{x}}$

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

$t(x) \overset{iid}{\sim} N\left(\sum_{i=1}^{k} w_i h_i(x), \sigma^2\right)$



- Learn **w** using MLE

target, input, params, noise variance

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

Maximizing log-likelihood

$\ln \prod = \sum \ln$

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

Maximize:

$$\underset{w}{\text{argmax}} \ln P(\mathcal{D} \mid \mathbf{w}, \sigma) = \ln \left[\left(\frac{1}{\sigma\sqrt{2\pi}}\right)^N \prod_{j=1}^{N} e^{\frac{-[t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j)]^2}{2\sigma^2}} \right]$$

$$= \underset{w}{\text{argmax}} \ln \Big(\cancel{\quad} \Big) - \sum_{j=1}^{N} \frac{\left[t(x_j) - \sum_i w_i h_i(x_j)\right]^2}{2\sigma^2}$$

const wrt w

$$= \underset{w}{\text{argmax}} - \sum_{j=1}^{N} \frac{\left[t(x_j) - \sum_{i=1}^{k} w_i h_i(x_j)\right]^2}{2\sigma^2}$$

$2\sigma^2 > 0$. doesn't affect argmax

$$= \underset{w}{\text{argmin}} \sum_{j=1}^{N} \left[t(x_j) - \sum_{i=1}^{k} w_i h_i(x_j)\right]^2$$

$\text{argmax} -f = \text{argmin } f$

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

Announcements

- Go to recitation!! ☺
  - Tuesday, 5:30pm in LOW 101
- First homework will go out today
  - Due on October 14
  - Start early!!

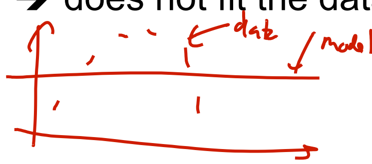
Bias-Variance Tradeoff

Machine Learning – CSE546
Carlos Guestrin
University of Washington

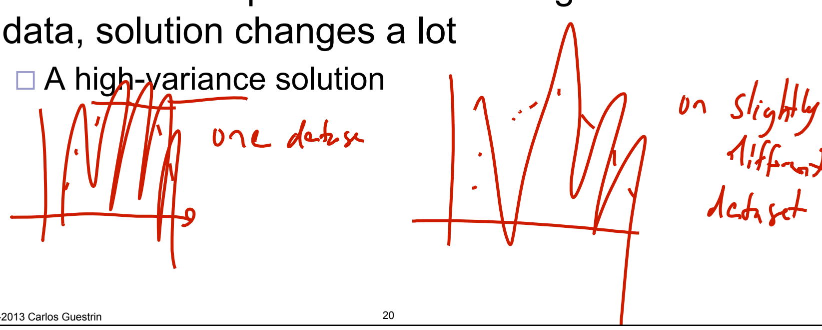
September 30, 2013

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

- Given dataset D with N samples, learn function $h_D(x)$ from $D \to$ learn $h_D(x)$
- If you sample a different dataset D' with N samples, you will learn different $h_D'(x)$
- **Expected hypothesis**: $E_D[h_D(x)] = \bar{h}_N(x)$
  ↑ 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: $\left(f(x) - \bar{h}_N(x)\right)^2$
  - Average Bias^2: $E_x\left[\left(f(x) - \bar{h}_N(x)\right)^2\right]$

$Var(x) = E_x[(x-\mu)^2]$

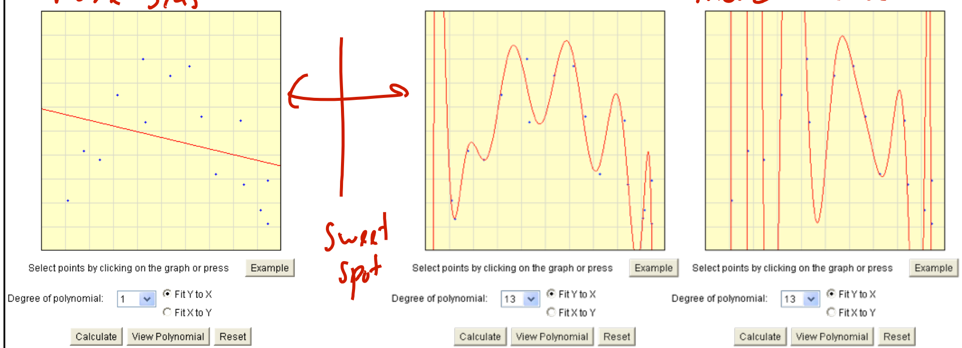
Variance of learner

- Given dataset D with N samples, learn function $h_D(x)$ $D \to h_D(x)$
- 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: $E_D\left[\left(\underbrace{h_D(x)}_{\text{what I learn with this dataset}} - \underbrace{\bar{h}_N(x)}_{\text{what I expect to learn}}\right)^2\right]$
  - Average variance: $E_x E_D\left[\left(h_D(x) - \bar{h}_N(x)\right)^2\right]$

Bias-Variance Tradeoff

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

more bias　　　　　　　　　　　　more variance



Bias-Variance Decomposition of Error

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

as with regression

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

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

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

play with this, hint:
$\bar{h}_N = E_D[h_D]$
all others are constants

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

- Error can be decomposed:

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

- 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