2/23/15

(FINISH) REGRESSION

(I) Bayesian Linear Regression (aka Ridge Regression)

Likelihood $\left[\; Y \mid X = x \sim N(\vec{w}^T \vec{x}, \sigma^2) \;\right]$ Same as last time

Prior $\left[\; \vec{w} \sim N(\vec{0}, t^2 I) \Leftrightarrow w_i \sim N(0, t^2) \right.$ IID components of $\vec{w}$

$$\Rightarrow P(\vec{w}) = \prod_{i=1}^{d} \frac{1}{\sqrt{2\pi t^2}} e^{-\frac{(w_i - 0)^2}{2t^2}}$$

Now $\hat{w}_{MAP} = \underset{\vec{w}}{\operatorname{argmax}} \; \log P(\vec{w} \mid D)$

$$= \underset{\vec{w}}{\operatorname{argmax}} \; \log \left[\frac{P(D \mid \vec{w}) P(\vec{w})}{P(D)} \right]$$

$$= \underset{\vec{w}}{\operatorname{argmax}} \left[\log P(D \mid \vec{w}) + \log P(\vec{w}) \right]$$

$$= \underset{\vec{w}}{\operatorname{argmax}} \left[-\sum_{i=1}^{n} \frac{(y_i - \vec{w}^T \vec{x}_i)^2}{2\sigma^2} - \frac{n}{2} \log 2\pi\sigma^2 - \sum_{j=0}^{d} \frac{w_j^2}{2t^2} - \frac{d}{2} \log 2\pi t^2 \right]$$

$$= \underset{\vec{w}}{\operatorname{argmin}} \left[\frac{1}{n} \sum_{i=1}^{n} (y_i - w^T x_i)^2 + \frac{1}{n} \underbrace{\left(\frac{2\sigma^2}{2t^2}\right)}_{\lambda} \|\vec{w}\|_2^2 \right]$$

$$= \underset{\vec{w}}{\operatorname{argmin}} \left[\underbrace{\frac{1}{n} \sum_{i=1}^{n} (y_i - \vec{w}^T \vec{x}_i)^2}_{\text{Avg. LOSS}} + \underbrace{\frac{\lambda}{n}}_{\text{trade-off}} \underbrace{\|w\|_2^2}_{\text{Regularization!}} \right]$$

Bayesian Loss $\equiv$ Avg-Fitting-Error + Norm-Square-of $\vec{w}$

Smell right? Let's check.

$\lambda \to 0$ $\left(\frac{\sigma^2}{t^2} \to 0 \right.$ or relatively uninformative prior$)$

$\Rightarrow$ $\hat{w}_{MAP} = \hat{w}_{MLE}$

$\lambda \to \infty$

$\hat{w}_{MAP} \to \vec{0}$

$\Rightarrow$ $\hat{y}_i = \vec{w}^T \vec{x}_i \to \vec{0}$ $\forall i$

Hmm. that's a bit odd. Why always zero?

replace $\sum_{j=0}^{d} w_j^2$ with $\sum_{j=1}^{d} w_j^2$

don't penalize norm of w_0

Now as $\lambda \to \infty$

$w_j \to 0$ $\forall j \neq 0$

$w_0 \to \frac{1}{n}\sum_{i=1}^{n} y_i$ [Avg of dataset. Why? HW2!]

So $\hat{y}_i \to y_{mean}$ (Makes sense)

Okay So $\hat{w}_{MAP} = \underset{\vec{w}}{\operatorname{argmin}} \; \frac{1}{n}\|Y - Xw\|_2^2 + \frac{\lambda}{n}\|w\|_2^2$

So $\frac{\partial L(\vec{w})}{\partial \vec{w}} = \frac{1}{n}\left[-2Y^T X + 2w^T X^T X\right] + 2\frac{\lambda}{n} w^T = 0$

$$\Rightarrow \frac{1}{n} X^T Y = \frac{1}{n} X^T X \vec{w} + \frac{\lambda}{n} \vec{w}$$

$$\Rightarrow \left(X^T X + \lambda I \right) \cdot \vec{w} = X^T Y$$

$$\Rightarrow \boxed{\hat{w}_{MAP} = \underbrace{(X^T X + \lambda I)}^{-1} X^T Y}$$

Compare with $\hat{w}_{MLE} = X^{\dagger} Y = \underbrace{(X^T X)^{-1}} X^T Y$

$$\underbrace{(X^T X)^{-1}}_{\text{might not be invertible}} \longrightarrow \underbrace{(X^T X + \lambda I)^{-1}}$$

might not be invertible

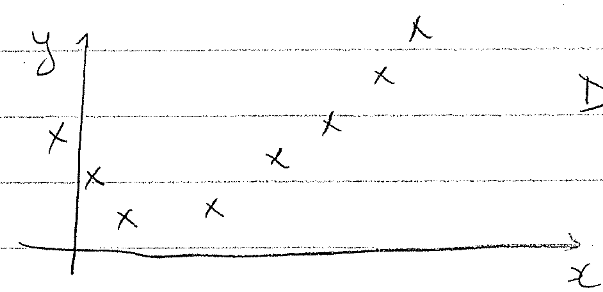
Shifts eigenvalues of $X^T X$ up ↑ by λ

$$\left[\because \quad (A + \lambda I) \cdot \vec{x} = A\vec{x} + \lambda \vec{x} \atop \qquad\qquad = (\text{eigenvalue} + \lambda) \cdot \vec{x} \right]$$

This is a Bayesian Interpretation of a Linear Algebra Procedure!

(2) Polynomial Regression

Consider 1D data $x \in \mathbb{R}^1$ for now (for simplicity)



Doesn't look like a linear relationship.

Can we do better?

Linear Regression: $\hat{y} = w_0 + w_1 x$

Polynomial Regression: $\hat{y} = w_0 + w_1 x^1 + w_2 x^2 + \cdots + w_d x^d$
(of degree-d)

$$= [w_0 \; w_1 \; \cdots \; w_d] \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^d \end{bmatrix}$$

But we can easily rewrite this as.

$$= \vec{w}^T \underbrace{\phi(x)}_{\text{higher-dim feature space "embedding"}}$$

$$\phi : \mathbb{R}^1 \to \mathbb{R}^{(d+1)}$$

But $\hat{y} = \vec{w}^T \phi(x)$ is still linear in $\vec{w}$!

That's all that matters!
Linearity in params $(\vec{w})$ not input $(\vec{x})$

so same steps apply!

In general $\vec{x} \in \mathbb{R}^d \longrightarrow \phi(\vec{x}) \in \mathbb{R}^D \quad D \neq d$

say $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \phi(\vec{x}) = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_1^2 \\ x_2^2 \\ x_3^2 \\ x_1 x_2 \\ x_2 x_3 \\ x_3 x_1 \end{bmatrix}$ } Quadratic Feature Augmentation / Quadratic "Kernel Map" [We will see Kernels later]

$\hat{y} = \vec{w}^T \phi(\vec{x})$ still linear in $\vec{w}$!

$[\equiv \vec{x}^T A \vec{x} + \vec{c}^T \vec{x}]$ Quadratic in $\vec{x}$

NEW TOPIC: MODEL SELECTION

③ How do we pick d in 1-dim polynomial regression? ($x \in \mathbb{R}^1$)

Alternatively,

Model 0: $y^{(0)} = w_0$

Model 1: $y^{(1)} = w_0 + w_1 x$

⋮

Model D: $y^{(D)} = w_0 + w_1 x + w_2 x^2 + \cdots + w_D x^D$

Model Selection: Which model class is better?

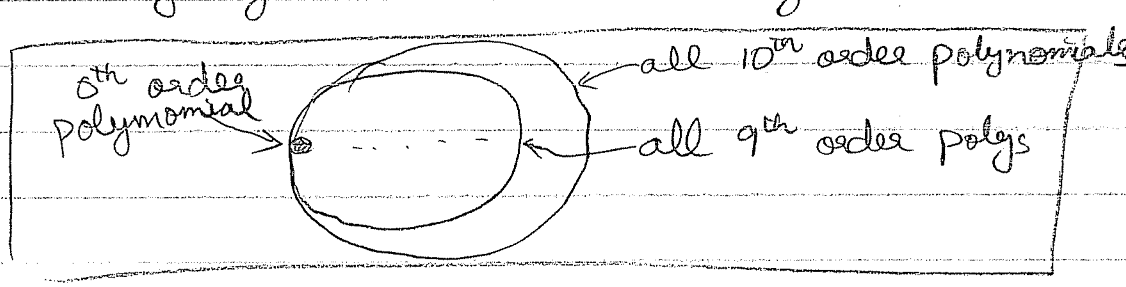
Notice that $\vec{w}^{(d)}_{OLS} = \underset{\vec{w} \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n} \sum_{\substack{\text{training} \\ \text{data}}} (y_i - w^{(d)T} \vec{x}_i)^2$

So how about?

$$d = \underset{\{1, \ldots, 100\}}{\operatorname{argmin}} \frac{1}{n} \sum_{\substack{\text{training} \\ \text{data}}} (y_i - \vec{w}^{(d)T}_{OLS} \vec{x}_i)^2$$

Won't work. Why?

Problem: Model classes are nested, $d=100$ will always give lowest training error (or E_{train})



Large Model Classes ($\equiv$ Strong ML models) will always do better in terms of E_{train}.

But why is low train error not a good thing?

Because that's not what we care about

④ What do we care about? Expected Error/Loss

$$X, Y \sim P(X, Y) \quad [\text{Unknown}]$$

What we want $= \min\limits_{w} \underbrace{E_{P(X,Y)}}_{\text{reality}} \Big[L\big(y, \underbrace{\hat{y}}_{\text{Model Class}}(x; \underbrace{w}_{\text{Parameters}})\big) \Big]$

$$= \min_{w} \int_{\underbrace{x}_{\substack{\text{Space of } X \\ \text{Say } \mathbb{R}^d}}} \int_{\underbrace{y}_{\substack{\text{Space of } Y \\ \text{Say } \mathbb{R}^1}}} L(y, \hat{y}(x; w))\, p(x, y)\, dx\, dy$$

Problems

→ $p(x, y)$ unknown

→ Integral hard to solve

→ Min of Integral

"Empirical Risk Minimization (ERM)"

$\equiv$ Let's approximate integral with samples $(\bar{x}_i, y_i) \sim P(X, Y)$

$$E_{approx} = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{y}(x_i; \vec{w}))$$

5 Model Selection in Practice

ALL DATA

~60% (say)	~20%	~20%
TRAIN	VAL	TEST
E_{train}	E_{val}	E_{test}

Might try multiple model classes

Used to estimate model parameters

$\hat{w} = \underset{\vec{w}}{\arg\min}\ E_{train}(\vec{w}, d)$

Used to choose model class

$\hat{d} = \underset{\{1,\dots,100\}}{\arg\min}\ E_{val}(\hat{w}, d)$

Used to estimate Expected Error/Loss

$E_{test}(\hat{w}, \hat{d})$

No Tweaking/learning on test set otherwise a biased estimator