

UCB - CS189
Introduction to Machine Learning
Fall 2015

Lecture 15: Gaussian Processes

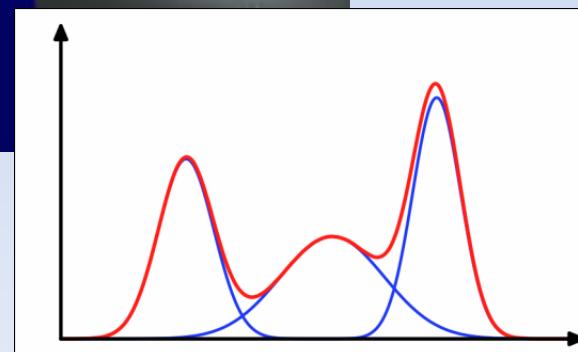
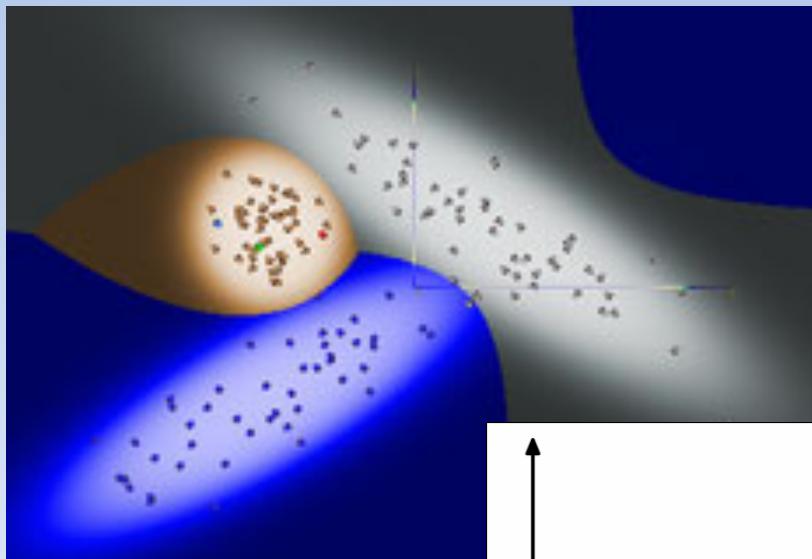
Isabelle Guyon



ChaLearn

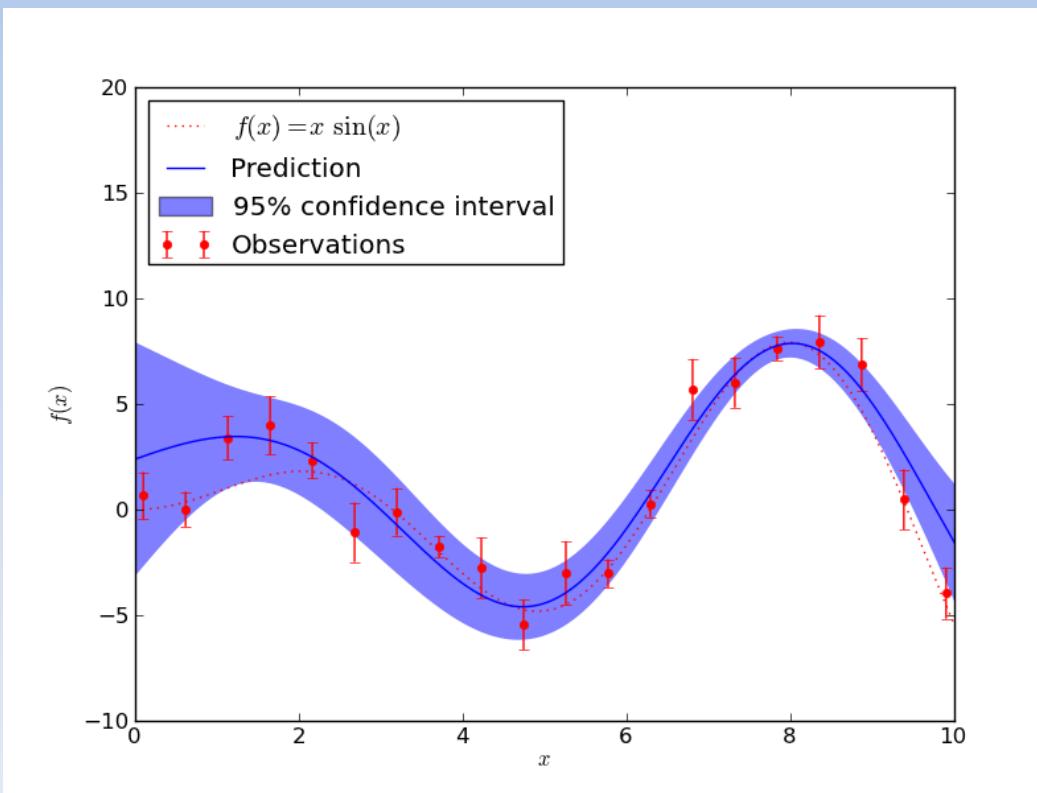
Come to my office hours...
Wed 2:30-4:30 Soda 329

Last time: Mixture models



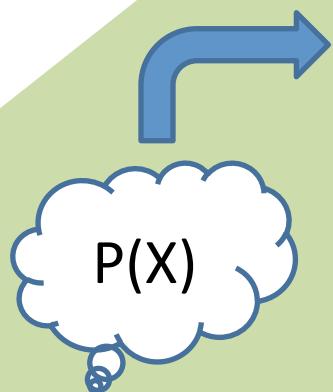
Come to my office hours...
Wed 2:30-4:30 Soda 329

Today: Gaussian processes

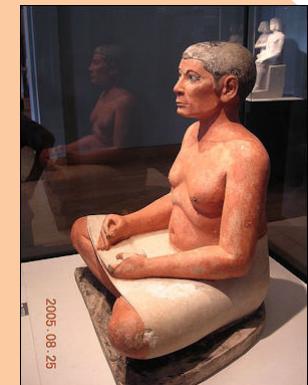
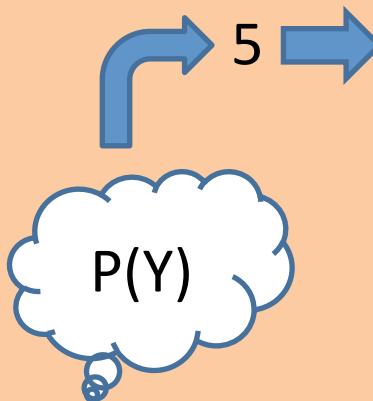


Data generating models

$$P(X, Y) = P(X)P(Y|X) = P(Y)P(X|Y)$$



86	98	78	6	63	78	78	14	16	1
0	16	12	74	8	85	85	30	30	87
2	86	1	0	23	80	22	67	75	75
40	40	76	76	60	29	29	1	81	30
32	32	99	69	76	56	6	56	52	2
8	23	83	57	57	21	21	70	1	8
76	3	72	82	80	21	51	74	54	4
91	16	00	10	86	98	78	63	78	34



$\downarrow P(Y|X)$

96 98 78 6 63 79...

Discriminative models



$$f(x) = \left(\sum_k \alpha_k [\Phi(x^k)] \cdot \Phi(x) \right)$$

w **k(x^k, x; θ)**

Algorithm:

Hebb's rule

$$\mathbf{w} = \sum_{k=1:N} y^k \mathbf{x}^k$$

$$\mathbf{w} = \mathbf{X}^T \mathbf{y}$$



"Perceptrons"

- α_k fitted to R_{train}
- $k(x^k, x) = x^k \cdot x$
- No θ
- $f(x) = w \cdot x$

Algorithms:

SVM

Logistic regression

Ridge regression

$$\mathbf{w} = \mathbf{X}^T \mathbf{y} = \Sigma^{-1} \mathbf{X}^T \mathbf{y}$$



Parzen windows

- $\alpha_k = y_k$
- Any $k(x^k, x; \theta)$
- θ fitted to R_{CV}
- $f(x) = \sum_k \alpha_k k(x^k, x; \theta)$

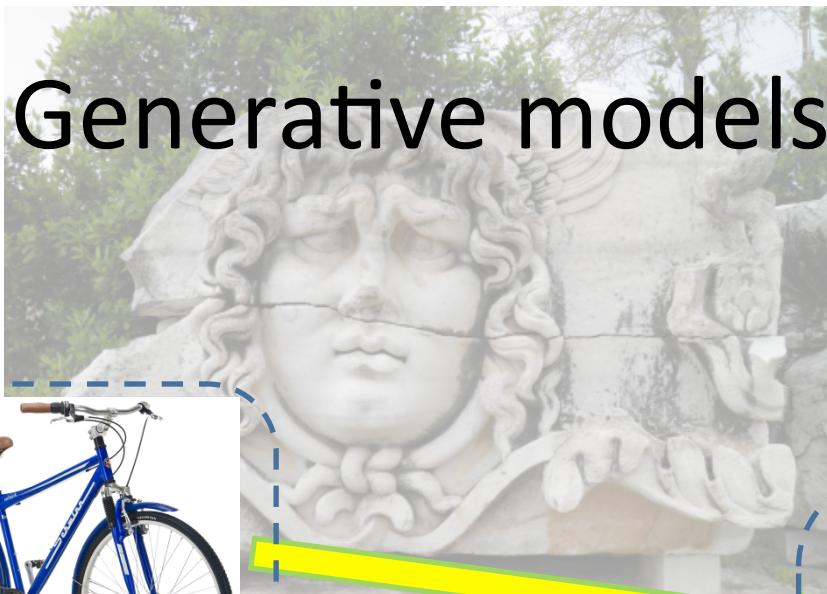
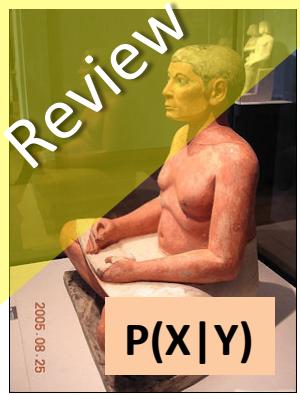


Kernel methods

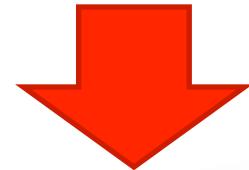
- α_k fitted to R_{train}
- Any $k(x^k, x; \theta)$
- θ fitted to R_{CV}
- $f(x) = \sum_k \alpha_k k(x^k, x; \theta)$

Hebb's method

- $\alpha_k = y_k$
- $k(x^k, x) = x^k \cdot x$
- No θ
- $f(x) = w \cdot x$



Today



Algorithm:

Centroid

$$\mathbf{w} = \sum_k y^k \mathbf{x}^k = \mathbf{X}^T \mathbf{y}$$

$$y^k \in \{+1/N_1, -1/N_0\}$$

$$\mathbf{w} = \boldsymbol{\mu}^{[1]} - \boldsymbol{\mu}^{[0]}$$



**Naïve
Bayes**

LDA



**Mixture
models**



**Gaussian
processes**

$$\text{Algorithm:}$$

$$\mathbf{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}^{[1]} - \boldsymbol{\mu}^{[0]})$$

$$\boldsymbol{\Sigma} = \mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}, \lambda > 0$$

$$\mathbf{w} = \boldsymbol{\Sigma}^{-1} \mathbf{X}^T \mathbf{y}$$

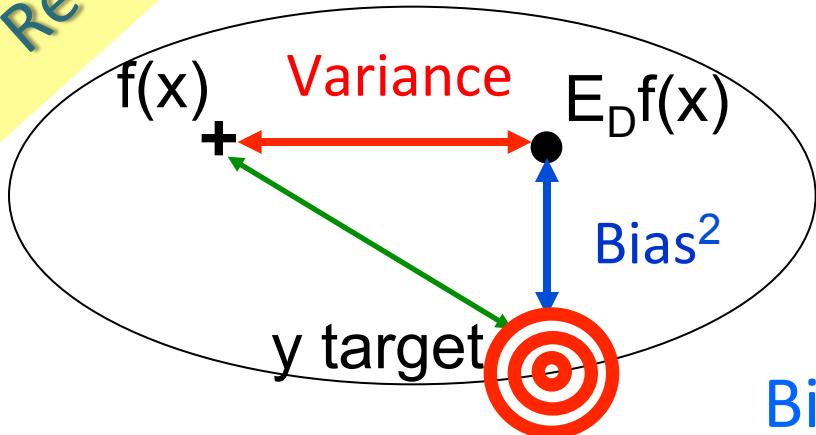
$$y^k \in \{+1/N_1, -1/N_0\}$$

Last
time



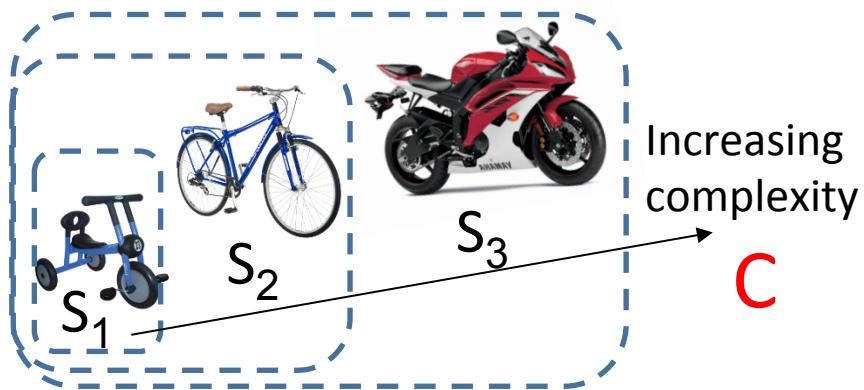
Review

Wrong assumptions don't really matter!



Bias vs. variance tradeoff

$$E_D[f(x)-y]^2 = [E_D f(x)-y]^2 + E_D[f(x)-E_D f(x)]^2$$

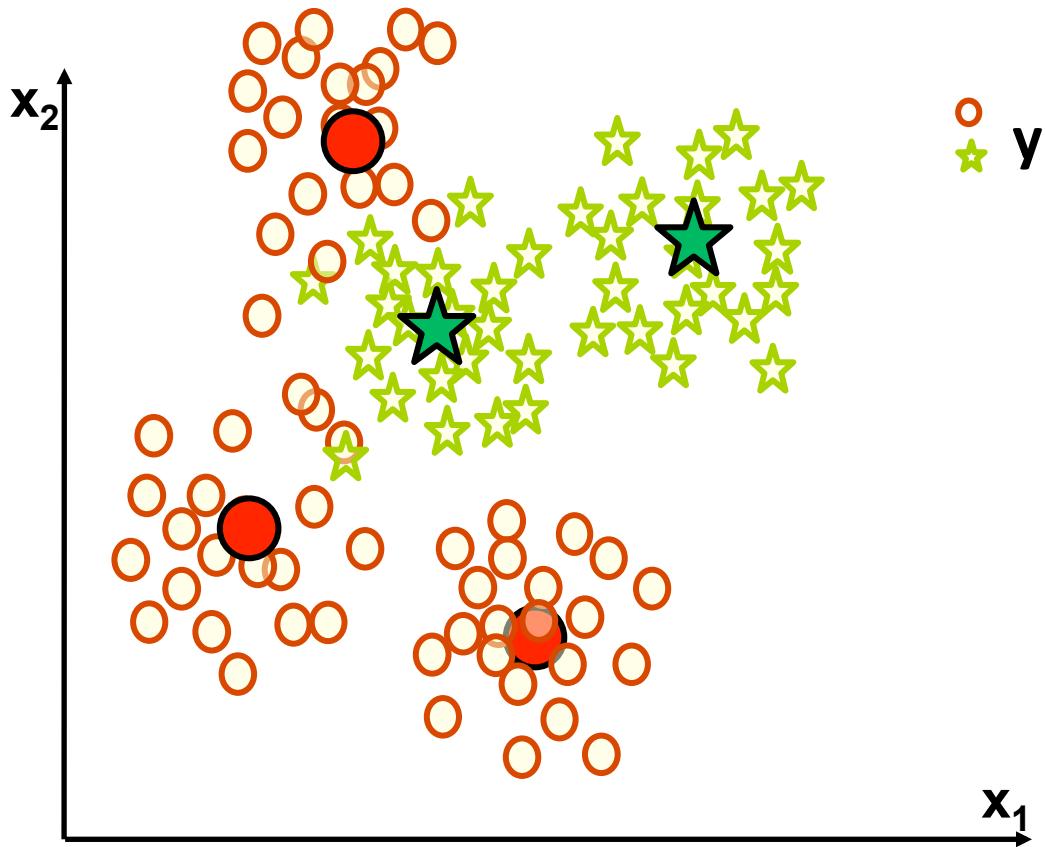


Structural risk minimization
 $R[f] \leq R_{\text{emp}}[f] + \epsilon(\delta, C/N)$

Bias is “good”,

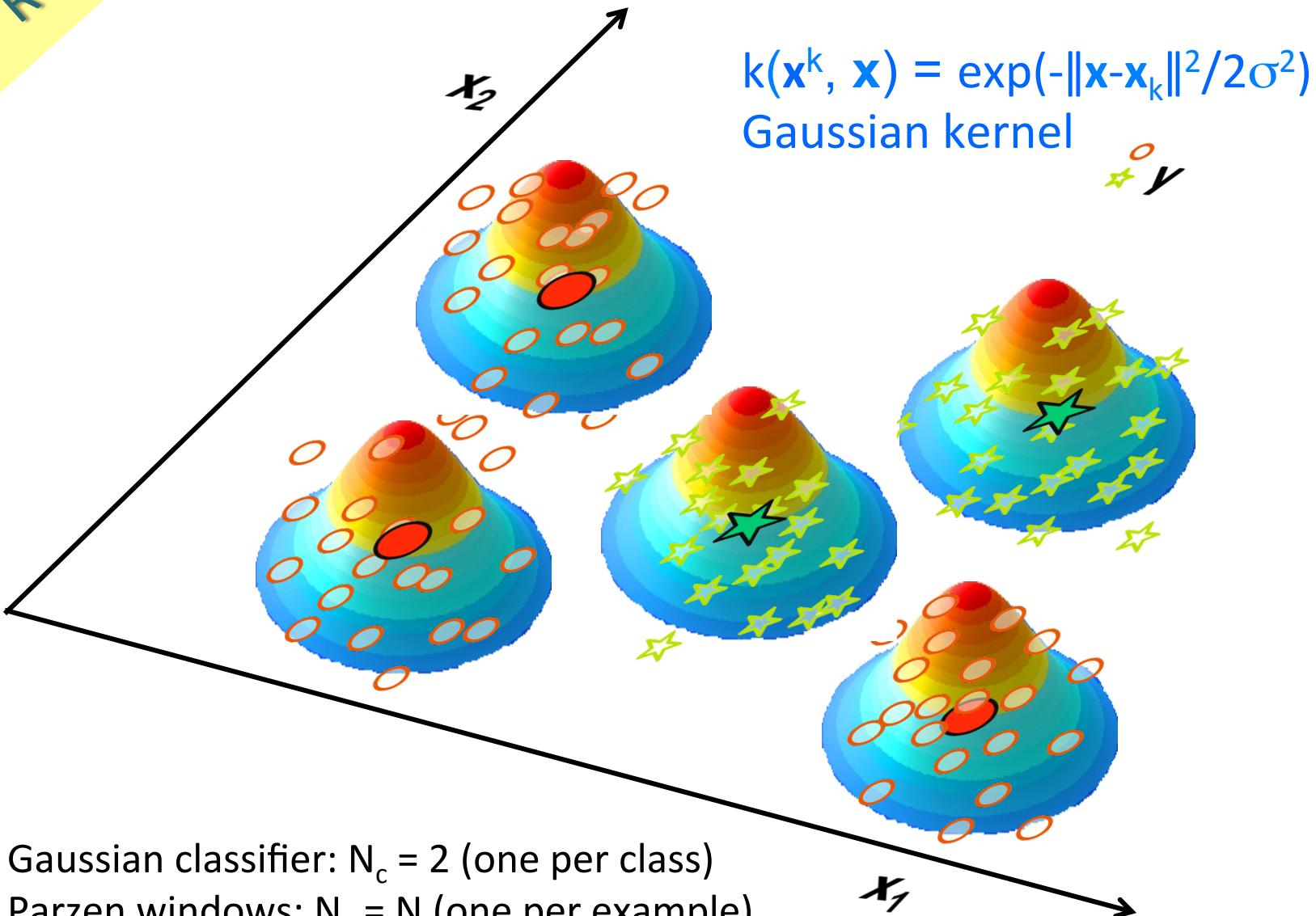
if lowering C helps you get better generalization error

Mixture models

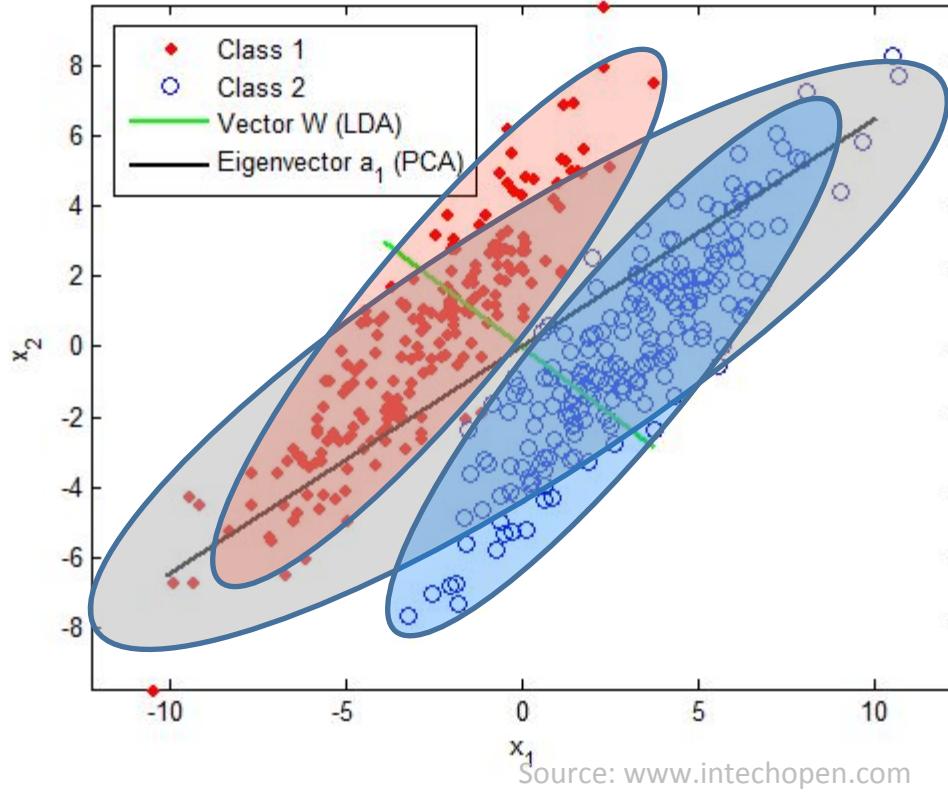


$$f(\mathbf{x}) = \sum_{k=1:N_c} \alpha_k k(\boldsymbol{\mu}^k, \mathbf{x}; \boldsymbol{\sigma}) + b$$

Gaussian mixtures



How to “kernelize” PCA and LDA?



Apply whitening in both case to get $\Xi = X \Sigma^{-1/2}$

PCA: Σ is the TOTAL covariance matrix.

LDA: Σ is the POOLED WITHIN CLASS covariance matrix.

Σ^{-1} for BIG data



$X^T X$ dim(d,d)
 XX^T dim(N,N)
 which one do I
 rather invert?

Singular value decomposition:

$X = VSU^T$, with $U^T U = 1$ and $V^T V = 1$

S diagonal dim(r, r): singular values, $r = \text{rank}(X) \leq \min(d, N)$

$X^T X = US^2 U^T$ and $XX^T = VS^2 V^T$ $\text{dim}(U) = (d, r)$ $\text{dim}(V) = (N, r)$

cheap $\Xi = XU$, $\Sigma^{1/2} = USU^T$, $\Sigma^{-1} = US^{-2}U^T$, etc.

r is small + keep only largest singular values.

Kernel trick:

$d \gg N \rightarrow XX^T = VS^2 V^T$ $XX^T \rightarrow K$ (kernel trick)

Instead of $\Xi = XU$ use $\Xi = VS$, much cheaper!

$\Xi_{\text{new}} = X_{\text{new}} U?$

$\Xi_{\text{new}} = X_{\text{new}} X^T VS^{-1}$

$X = VSU^T$ so $U = X^T VS^{-1}$

$X_{\text{new}} X^T \rightarrow [k(x^h, x^k)]$ (kernel trick)

Parameter estimation

- Maximum likelihood (ML):

Param = $\text{argmax}(\underbrace{P(\text{data} | \text{model})}_{\text{likelihood}})$

Examples:

$$\mu^{[y]} = \sum_{x^k \in y} x^k \quad \text{likelihood}$$

$$\sigma_{ij} = (1/N) \sum_{k=1:N} (x_i^k - \mu_i)(x_j^k - \mu_j)$$

- Maximum A Posteriori (MAP):

Param = $\text{argmax}(\underbrace{P(\text{model} | \text{data})}_{\text{posterior}})$

$$= \text{argmax}(\underbrace{P(\text{data} | \text{model})}_{\text{likelihood}} \underbrace{P(\text{model})}_{\text{prior}})$$

Bayesian MAP \cong SRM

- Maximum A Posteriori (MAP):

$$f = \operatorname{argmax} P(f | D)$$

$$= \operatorname{argmax} P(D | f) P(f)$$

$$= \operatorname{argmin} \underbrace{-\log P(D | f)}_{\text{Negative log likelihood}} + \underbrace{-\log P(f)}_{\text{Negative log prior}}$$

$= \text{Empirical risk } R_{\text{emp}}[f]$ $= \text{Regularizer } \Omega[f]$

- Structural Risk Minimization (SRM):

$$f = \operatorname{argmin} R_{\text{emp}}[f] + \Omega[f]$$

Example: Ridge regression a.k.a. Gaussian processes

- Structural Risk Minimization (SRM):

$$\mathbf{w} = \operatorname{argmin}_{\mathbf{w}} \underbrace{\|\mathbf{y} - \mathbf{X} \mathbf{w}^T\|^2}_\text{Negative log likelihood} + \underbrace{\lambda \|\mathbf{w}\|^2}_\text{Negative log prior}$$

- Maximum A Posteriori (MAP):

$$\mathbf{w} = \operatorname{argmin}_{\mathbf{w}} -\log P(D|f) -\log P(f)$$

$$= \operatorname{argmax}_{\mathbf{w}} P(D|f) P(f)$$

$$= \operatorname{argmax}_{\mathbf{w}} \exp(-\|\mathbf{y} - \mathbf{X} \mathbf{w}^T\|^2) \exp(-\lambda \|\mathbf{w}\|^2)$$

$$= \operatorname{argmax}_{\mathbf{w}} \exp(-\|\mathbf{y} - \mathbf{X} \mathbf{w}^T\|^2) \boxed{\exp(-\|\mathbf{w}\|^2 / 2\sigma^2)}$$

Bayesian “learning”

- Remember the idea of ensembles:
Do not select a model, “vote” on all possible models according to their “performance”.

Replace:

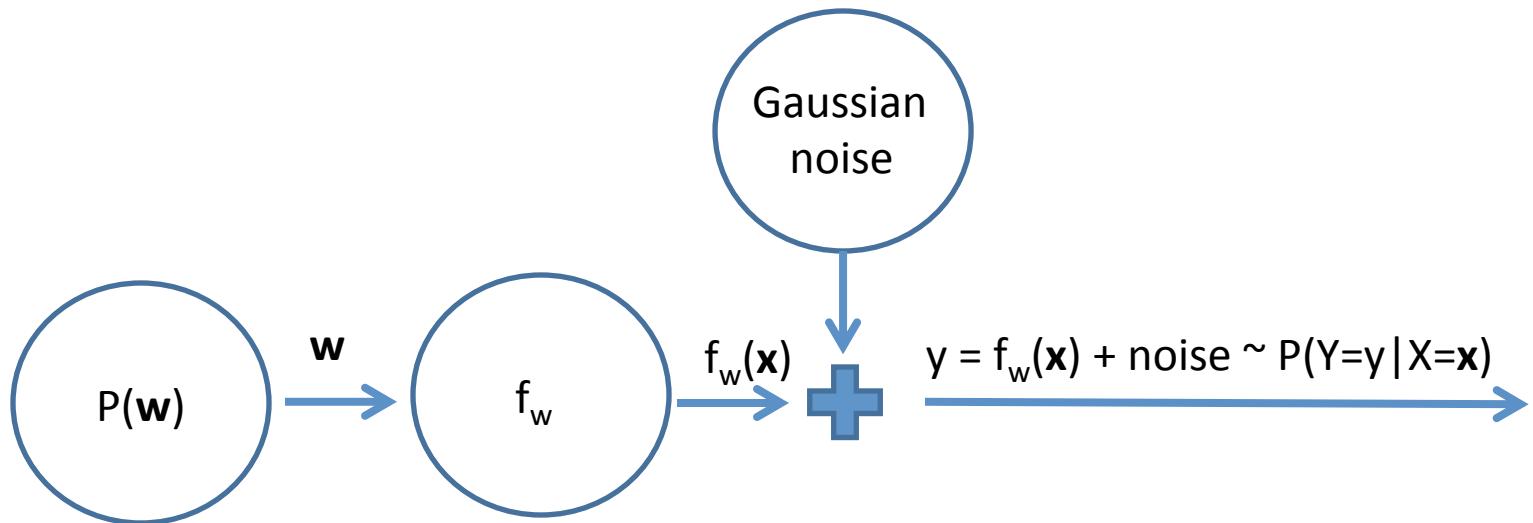
$$\mathbf{w} = \operatorname{argmax}_{\mathbf{w}} \underbrace{\exp(-\|\mathbf{y} - \mathbf{X} \mathbf{w}^T\|^2)}_{f(\mathbf{x}) = \mathbf{w} \mathbf{x}^T} \underbrace{\exp(-\|\mathbf{w}\|^2 / 2\sigma^2)}_{\sim P(\mathbf{w} | \text{data})}$$

$$f(\mathbf{x}) = \mathbf{w} \mathbf{x}^T \quad \sim P(\mathbf{w} | \text{data})$$

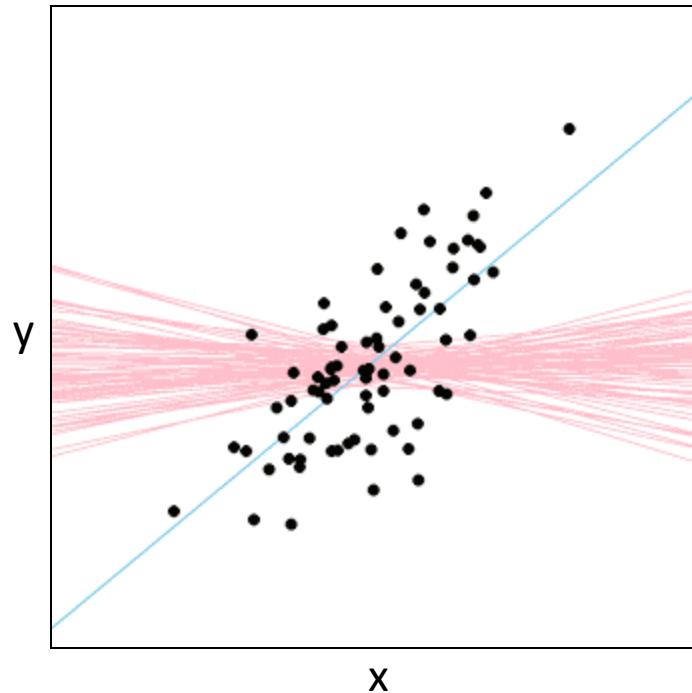
by

$$P(y|x) = \int_w P(y|x, w) P(w|data) dw$$

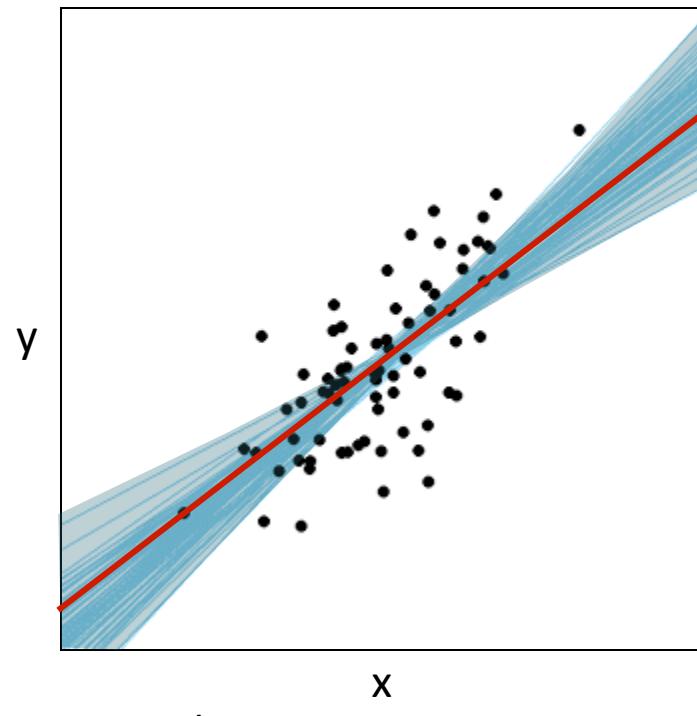
Generative model



Linear Gaussian Processes

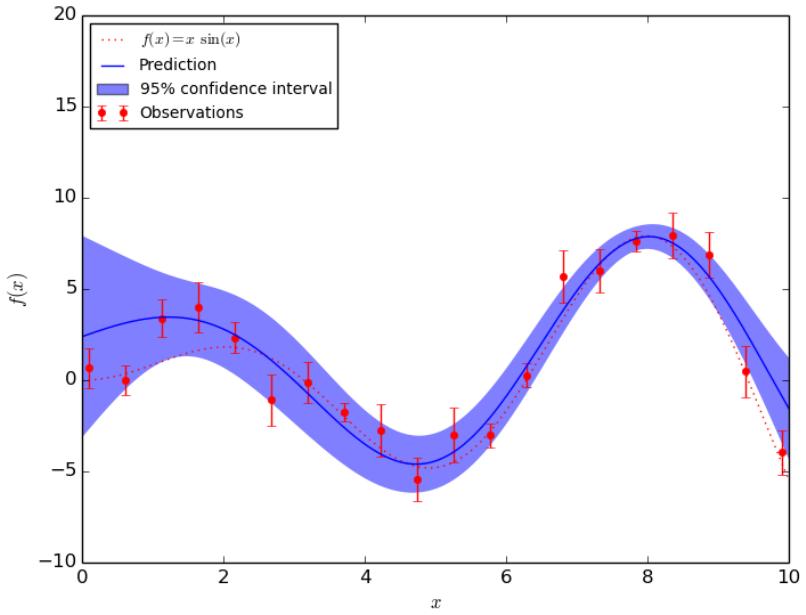
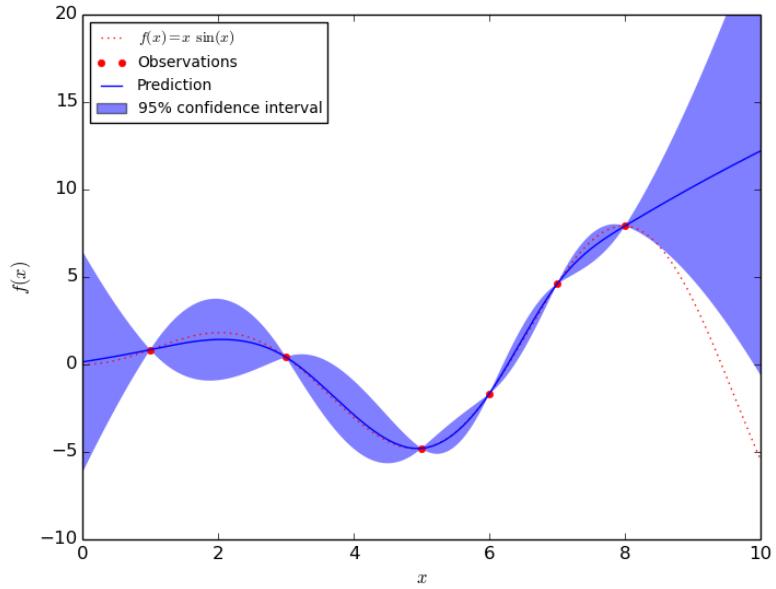


Blue: frequentist regression
Red: prior on w

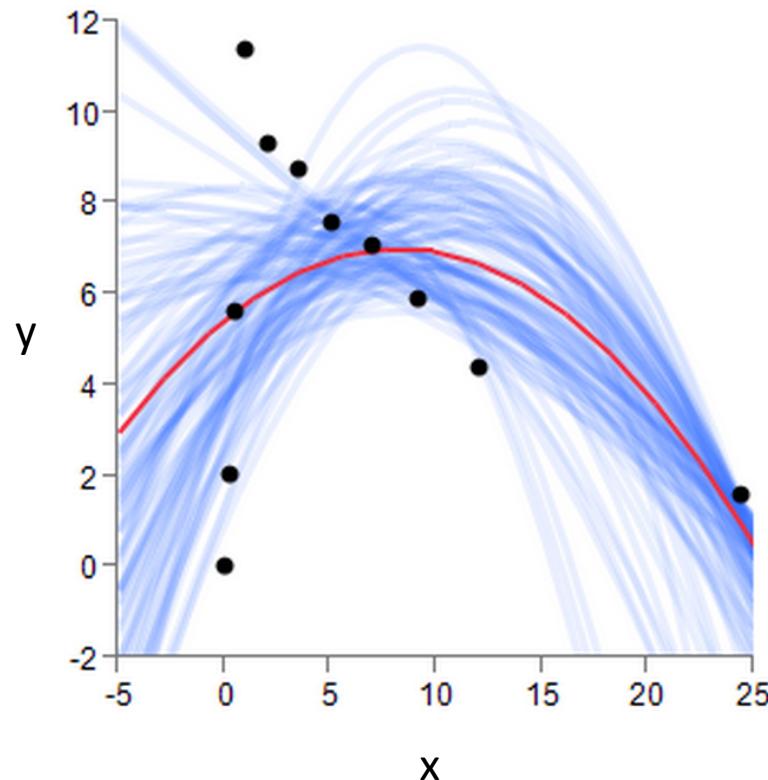


Blue: Bayesian regression

Kernel Gaussian Processes



Bootstrap



<http://blogs.sas.com/content/jmp/2012/12/04/visualizing-the-variability-of-a-curve-using-bootstrapping/>

Summary

- **Kernel trick for generative models:** When $d \gg N$ the covariance matrix $X^T X$ is too large, we can use XX^T instead in many algorithms. This lends itself to “kernelization”:
 - Kernel PCA,
 - Kernel LDA,
 - Kernel ridge regression a.k.a. “Gaussian processes”.
- **Parameter estimation:** From a data generating point of view, **Maximum Likelihood (ML)** makes sense as a parameter estimation paradigm:
 - The “classical” ways of estimating mean and covariance are ML estimators.
 - Regularization has also its equivalent in the data generating setting: the Maximum A Posteriori (MAP) framework.
- **Gaussian processes:** For regression, Gaussian processes allow you to estimate $P(Y|X)$, so compute error bars of predictions.



Thanks



Alexei (Alyosha) Efros



Nihar B. Shah



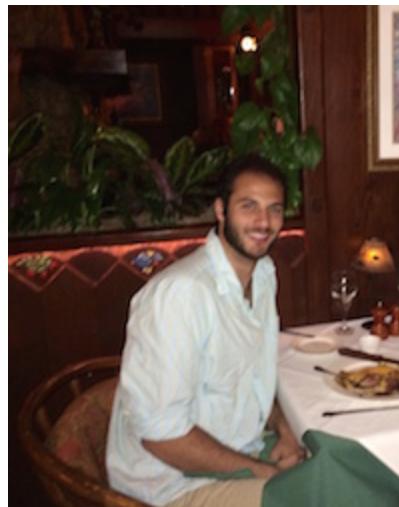
Brian Chu



Weicheng Kuo



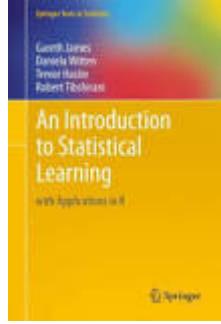
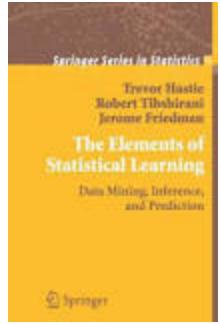
Deepak Pathak



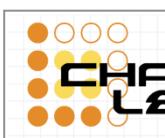
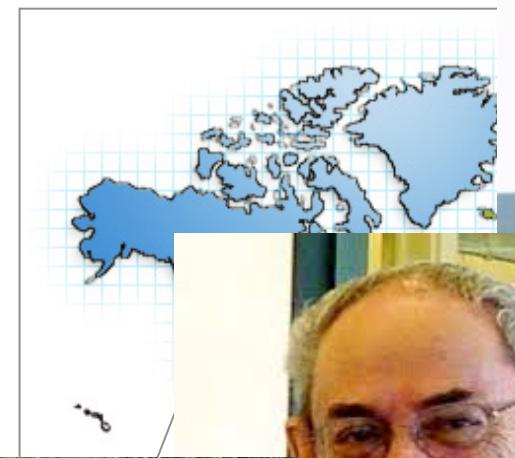
Shaun Singh



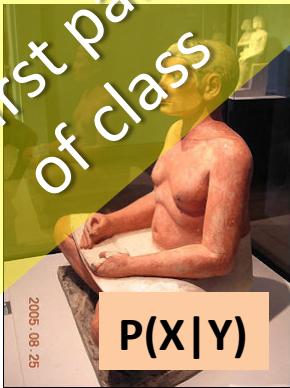
Faraz Tavakoli



Thanks



First part
of class



Algorithm:

Centroid

$$\mathbf{w} = \sum_k y^k \mathbf{x}^k = \mathbf{X}^T \mathbf{y}$$
$$y^k \in \{+1/N_1, -1/N_0\}$$
$$\mathbf{w} = \mu^{[1]} - \mu^{[0]}$$



Naïve Bayes

Generative models



LDA

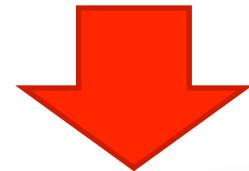
Algorithm:

$$\mathbf{w} = \Sigma^{-1}(\mu^{[1]} - \mu^{[0]})$$
$$\Sigma = \mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}, \lambda > 0$$
$$\mathbf{w} = \Sigma^{-1} \mathbf{X}^T \mathbf{y}$$
$$y^k \in \{+1/N_1, -1/N_0\}$$

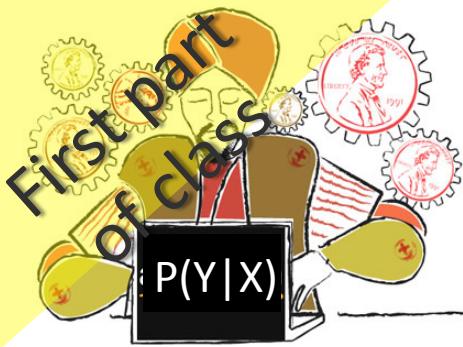
Mixture models



Today



Gaussian processes



Discriminative models

$$f(x) = \left(\sum_k \alpha_k [\Phi(x^k)] \cdot \Phi(x) \right)$$

w $k(x^k, x; \theta)$

Algorithm:

Hebb's rule

$$w = \sum_{k=1:N} y^k x^k$$

$$w = X^T y$$



"Perceptrons"

- α_k fitted to R_{train}
- $k(x^k, x) = x^k \cdot x$
- No θ
- $f(x) = w \cdot x$

Algorithms:

SVM

Logistic regression

Ridge regression

$$w = X^T y = \Sigma^{-1} X^T y$$



Parzen windows

- $\alpha_k = y_k$
- Any $k(x^k, x; \theta)$
- θ fitted to R_{CV}
- $f(x) = \sum_k \alpha_k k(x^k, x; \theta)$



Kernel methods

- α_k fitted to R_{train}
- Any $k(x^k, x; \theta)$
- θ fitted to R_{CV}
- $f(x) = \sum_k \alpha_k k(x^k, x; \theta)$

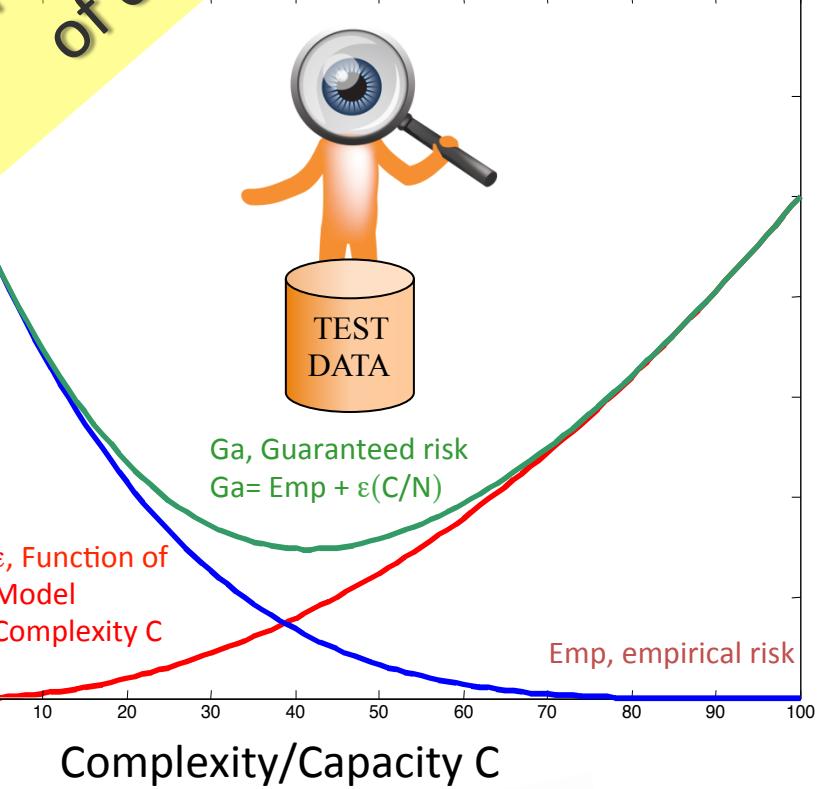
Hebb's method

- $\alpha_k = y_k$
- $k(x^k, x) = x^k \cdot x$
- No θ
- $f(x) = w \cdot x$

First part
of class

Structural Risk Minimization

Vapnik, 1974



With *high probability* ($1-\delta$),
 $0 < \delta \ll 1$

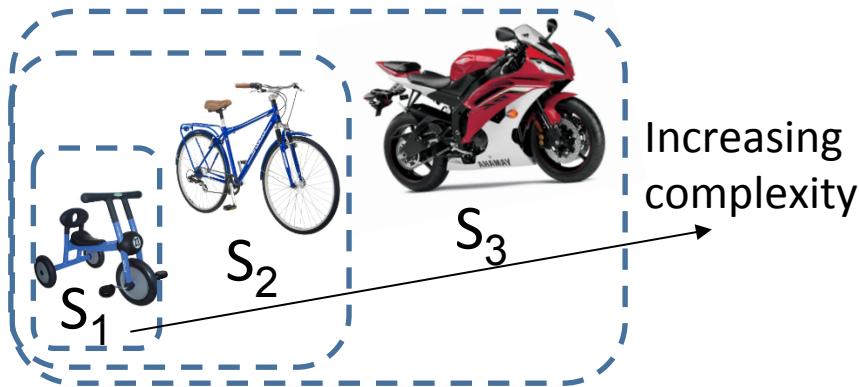
$$R[f] \leq R_{\text{emp}}[f] + \epsilon(\delta, C/N)$$

$R_{\text{gua}}[f]$

$$-\log P(D|f) \sim -\log P(D|f) - \log P(f)$$

Negative log likelihood
= Empirical risk $R[f]$

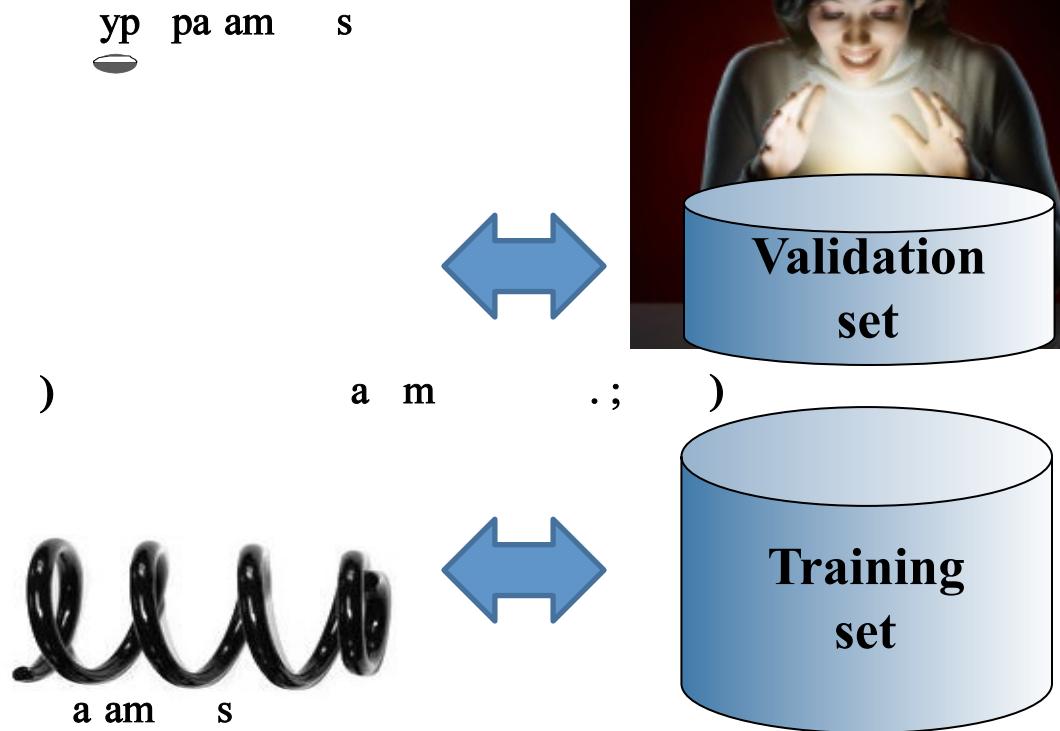
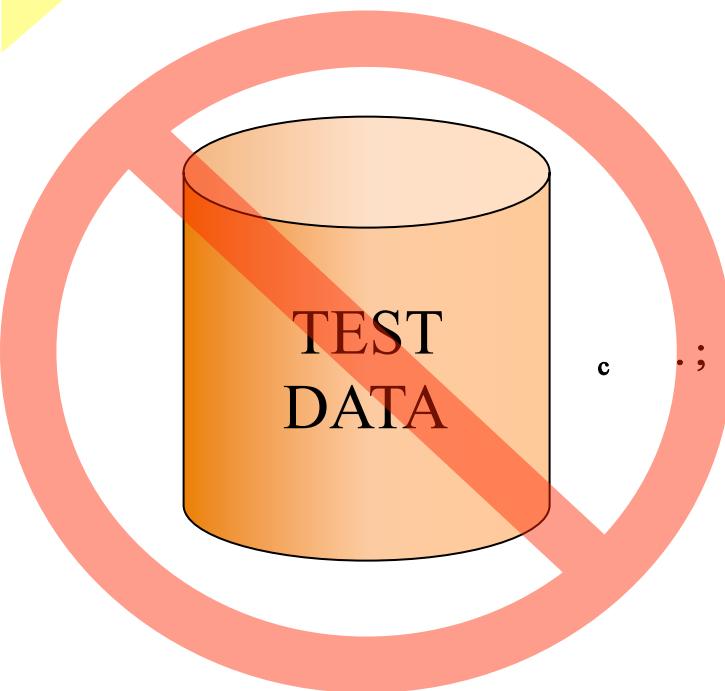
Negative log prior
= Regularizer $\Omega[f]$



Make nested subsets of models,
increasing complexity/capacity:
 $S_1 \subset S_2 \subset \dots \subset S_N$

First part
of class

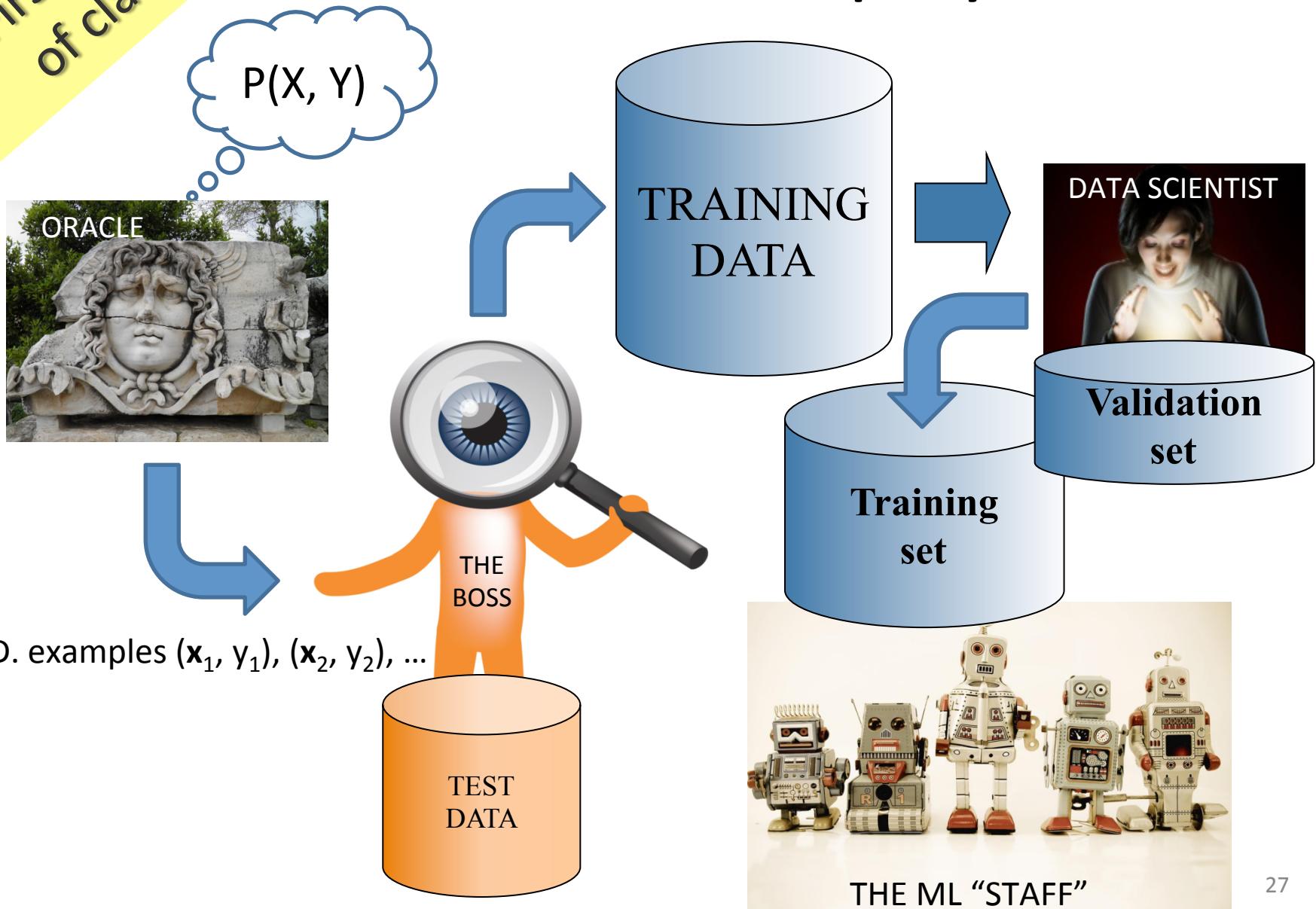
Model selection / hyper-parameter search



$$f^{**} = \operatorname{argmin}_{\theta} R_2[f^*, D], \text{ such that } f^* = \operatorname{argmin}_{\alpha} R_1[f, D]$$

First part
of class

Your turn to play...



First part
of class

Drive safely!

Keep C/N (or d_{eff}/N) under control



Come to my office hours...
Wed 2:30-4:30 Soda 329

Next time: Prof. Efros!

