Announcements

- Homework 2 due <u>tonight</u>

$$f(w) = \frac{1}{n}\sum_{i=1}^{n} \ell_i(w) + \lambda \|w\|_2^2$$

$$\nabla f(w) = \frac{1}{n}\sum_{i=1}^{n} \nabla \ell_i(w) + 2\lambda w$$

$$I_t \sim \text{uniform}(\{1, \dots, n\})$$

$$g_t = \nabla \ell_{I_t}(w_t) + 2\lambda w_t$$

$$\mathbb{E}_{I_t}[g_t] = \nabla f(w_t)$$

Bayesian Methods

Machine Learning – CSE546
Kevin Jamieson
University of Washington

November 1, 2018

MLE Recap - coin flips

- **Data**: sequence *D= (HHTHT…),* **k heads** out of **n flips**
- **Hypothesis:** P(Heads) = θ, P(Tails) = 1-θ

$$P(\mathcal{D}|\theta) = \theta^k (1-\theta)^{n-k}$$

- Maximum likelihood estimation (MLE): Choose θ that maximizes the probability of observed data:

$$\begin{aligned}\widehat{\theta}_{MLE} &= \arg\max_{\theta} \; P(\mathcal{D}|\theta) \\ &= \arg\max_{\theta} \; \log P(\mathcal{D}|\theta)\end{aligned}$$

$$\widehat{\theta}_{MLE} = \frac{k}{n}$$

What about prior

- *Billionaire*: Wait, I know that the coin is “close” to 50-50. What can you do for me now?
- **You say: I can learn it the Bayesian way…**

Bayesian Learning

- Use Bayes rule:

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

- Or equivalently:

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

Bayesian Learning for Coins

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

- Likelihood function is simply Binomial:

$$P(\mathcal{D}|\theta) = \theta^k(1-\theta)^{n-k}$$

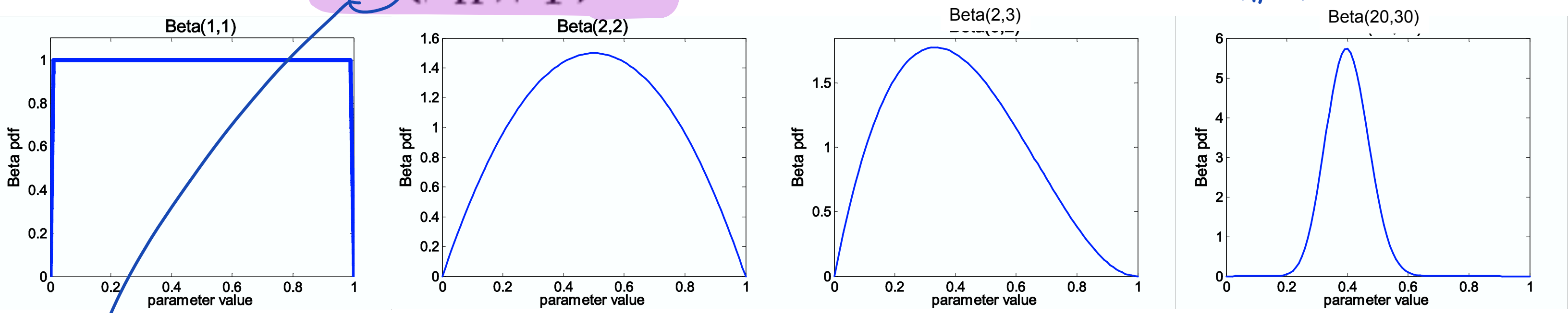
- What about prior?
  - Represent expert knowledge
- Conjugate priors:
  - Closed-form representation of posterior
  - **For Binomial, conjugate prior is Beta distribution**

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: $\frac{\beta_H}{\beta_H + \beta_T}$

Mode: $\frac{\beta_H - 1}{\beta_H + \beta_T - 2}$



- Likelihood function: $P(\mathcal{D}|\theta) = \theta^k (1-\theta)^{n-k}$
- Posterior: $P(\theta \mid \mathcal{D}) \propto P(\mathcal{D} \mid \theta) P(\theta)$

$$B(x,y) = \frac{\Gamma(x+y)}{\Gamma(x)\Gamma(y)}$$

$$\Gamma(x) = \int_0^\infty z^{x-1} e^{-z}\, dz$$

$$\Gamma(m) = (m-1)! = (m-1)(m-2)\cdots(2)(1)$$ m integer

$$\propto \theta^{k+\beta_H-1} (1-\theta)^{n-k+\beta_T-1}$$

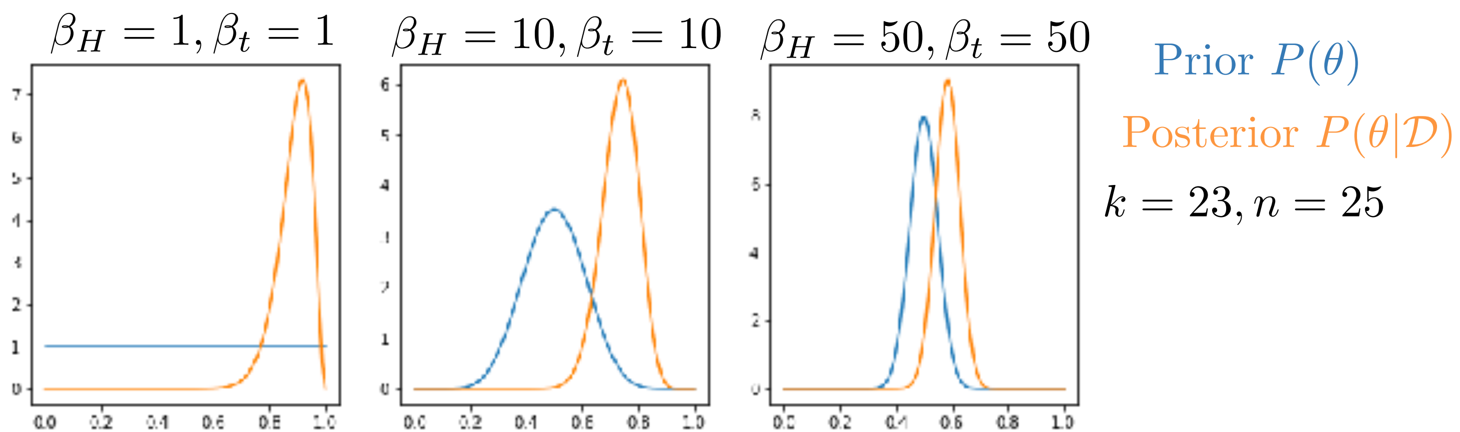
$$= Beta(k+\beta_H, n-k+\beta_T)$$

Posterior distribution

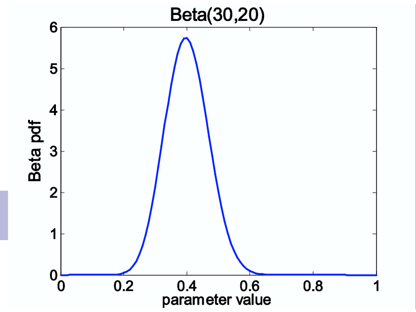
- Prior: $Beta(\beta_H, \beta_T)$
- Data: k heads and (n−k) tails

- Posterior distribution:

$$P(\theta|\mathcal{D}) = Beta(k+\beta_H, (n-k)+\beta_T)$$



Using Bayesian posterior



- Posterior distribution:

$$P(\theta|\mathcal{D}) = Beta(k+\beta_H, (n-k)+\beta_T)$$

- Bayesian inference:
  - Estimate mean

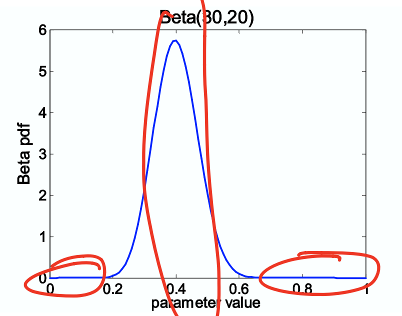
$$E[\theta] = \int_0^1 \theta P(\theta|\mathcal{D}) d\theta$$

  - Estimate arbitrary function f

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

  - For arbitrary f integral is often hard to compute

MAP: Maximum a posteriori approximation



$$P(\theta|\mathcal{D}) = Beta(k + \beta_H, (n-k) + \beta_T)$$

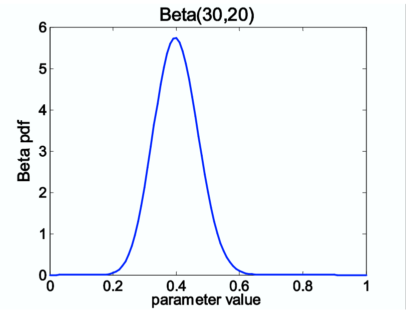
$$E[f(\theta)] = \int_0^1 f(\theta) P(\theta \mid \mathcal{D}) d\theta$$

- As more data is observed, Beta is more certain

- MAP: use most likely parameter:

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

MAP for Beta distribution

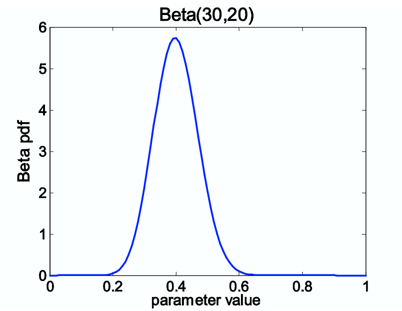


$$P(\theta|\mathcal{D}) \propto \theta^{k+\beta_H-1}(1-\theta)^{n-k+\beta_T-1}$$

- MAP: use most likely parameter:

$$\widehat{\theta} = \arg\max_{\theta} P(\theta \mid \mathcal{D}) = \frac{k+\beta_H-1}{n+\beta_H+\beta_T-2}$$

MAP for Beta distribution



$$P(\theta|\mathcal{D}) \propto \theta^{k+\beta_H-1}(1-\theta)^{n-k+\beta_T-1}$$

- MAP: use most likely parameter:

$$\widehat{\theta} = \arg\max_{\theta} P(\theta \mid \mathcal{D}) = \frac{k+\beta_H-1}{n \qquad +\beta_T-1}$$

- Beta prior equivalent to extra coin flips
- As $N \to 1$, prior is "forgotten"
- **But, for small sample size, prior is important!**

Bayesian vs Frequentist

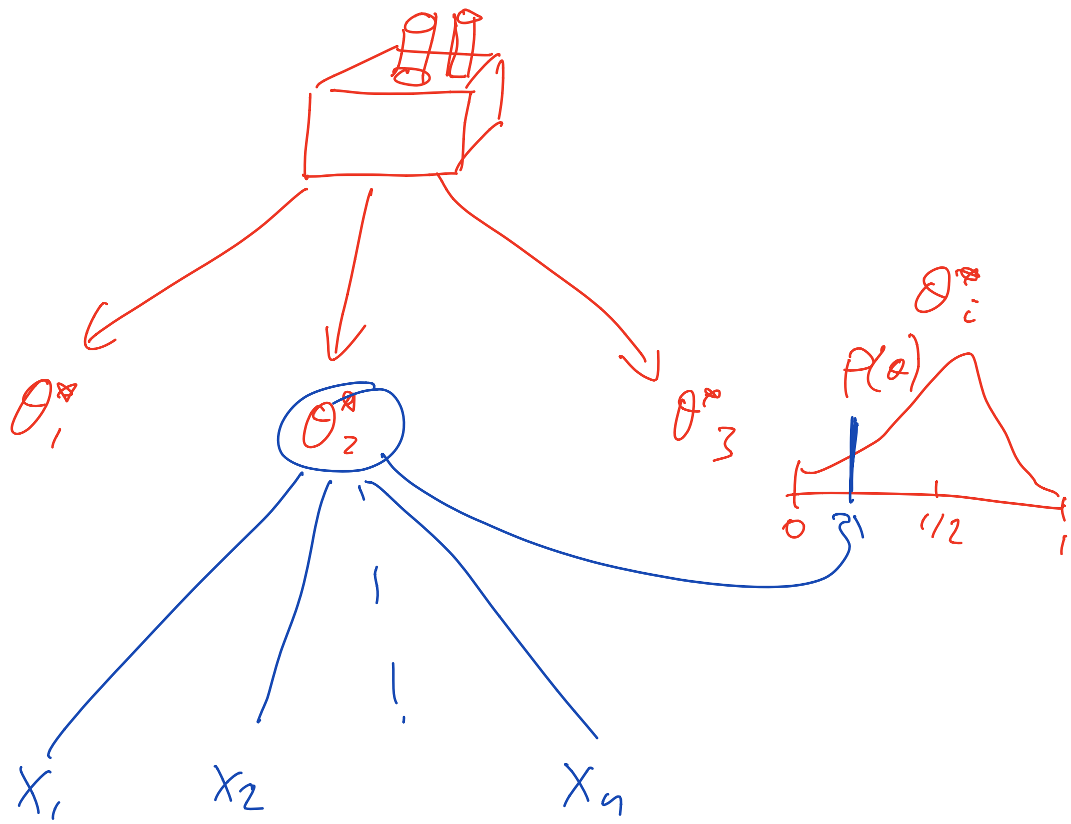
- Data: $\mathcal{D}$
- Frequentists treat unknown θ ***as fixed*** and the data *D* ***as random.*** $\widehat{\theta} = t(\mathcal{D})$

Diff. analysis for each estimator t.

- Bayesian treat the data *D* **as fixed** and the unknown θ ***as random*** $P(\theta|\mathcal{D})$

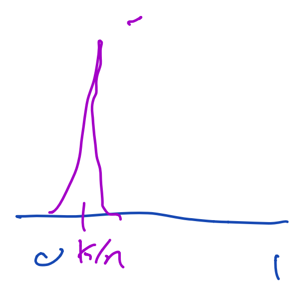
Given a "correct prior" we have $P(\theta|D)$

$\theta^* \sim P(\theta)$, $X \sim P(D|\theta^*)$, $P(\theta^*|D) \propto P(D|\theta^*)P(\theta^*)$



$k = \sum_{i=1}^{n} x_i$ heads of n flips

$$\mathbb{P}(\theta_2^* = \theta \mid D) \propto \underline{P(D \mid \theta)}\, P(\theta)$$



Recap for Bayesian learning

Bayesians are optimists:

- "If we model it correctly, we quantify uncertainty exactly"
- Answers all questions "simultaneously" with posterior probability
- Assumes one can accurately model:
  - Observations and link to unknown parameter *θ*: $p(x|\theta)$
  - Distribution, structure of unknown *θ*: $p(\theta)$

Frequentist are pessimists:

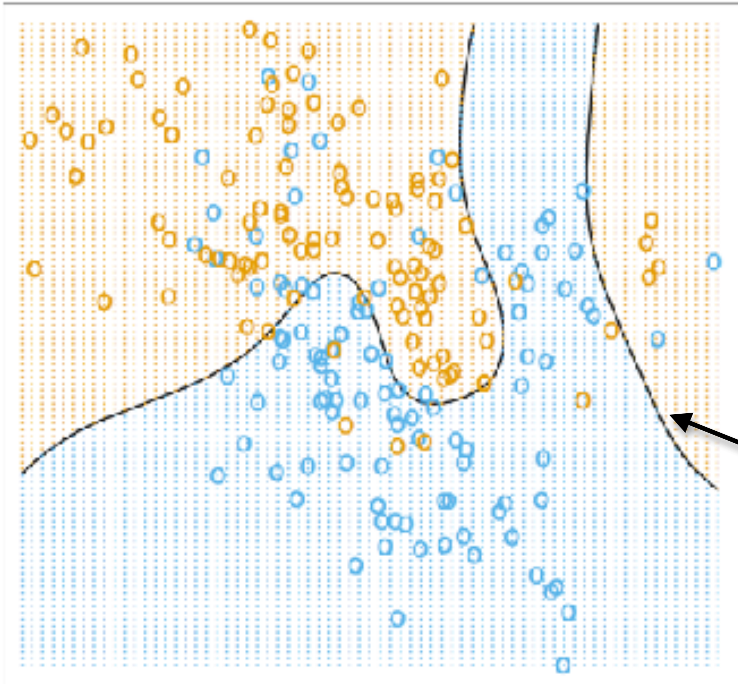
- "All models are wrong, prove to me your estimate is good"
- Answers each question with a separately analyzed estimator
- Makes very few assumptions, e.g. $\mathbb{E}[X^2] < \infty$ and constructs an estimator (e.g., median of means of disjoint subsets of data)

Nearest Neighbor

Machine Learning – CSE546
Kevin Jamieson
University of Washington

November 1, 2018

Some data, Bayes Classifier



Training data:

True label: +1

True label: -1

Optimal "Bayes" classifier:

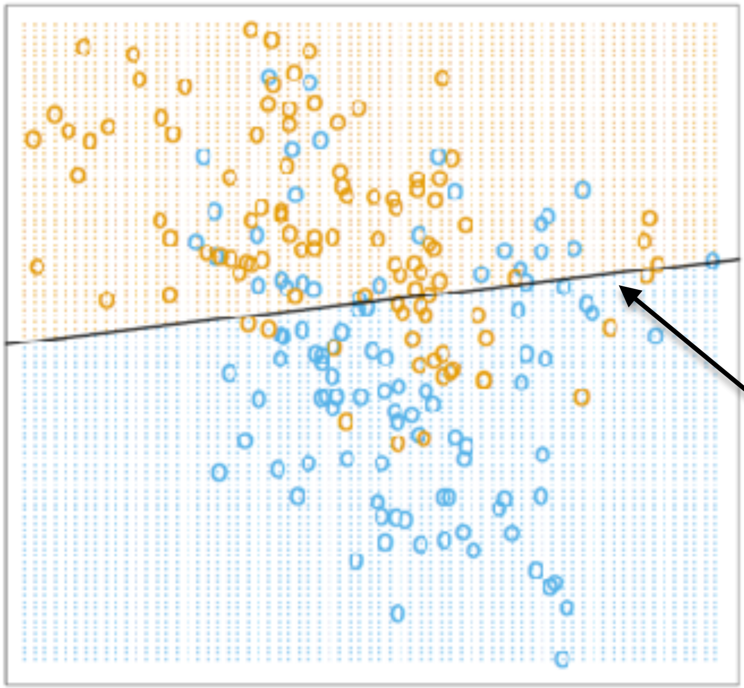
$$\mathbb{P}(Y = 1 | X = x) = \frac{1}{2}$$

Predicted label: +1

Predicted label: -1

Figures stolen from Hastie et al

Linear Decision Boundary



Training data:

True label: +1

True label: -1

Learned:

Linear Decision boundary

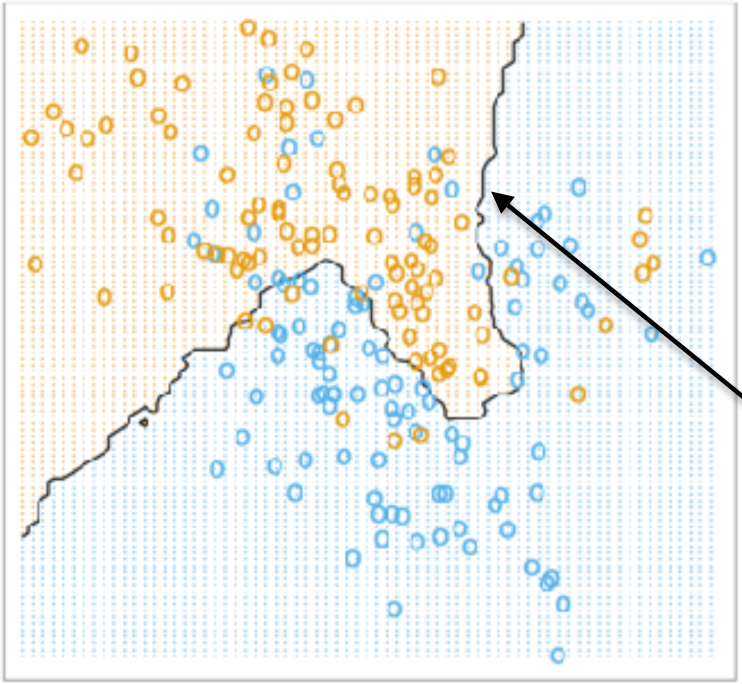
$$x^T w + b = 0$$

Predicted label: +1

Predicted label: -1

Figures stolen from Hastie et al

15 Nearest Neighbor Boundary



Training data:


True label: +1


True label: -1

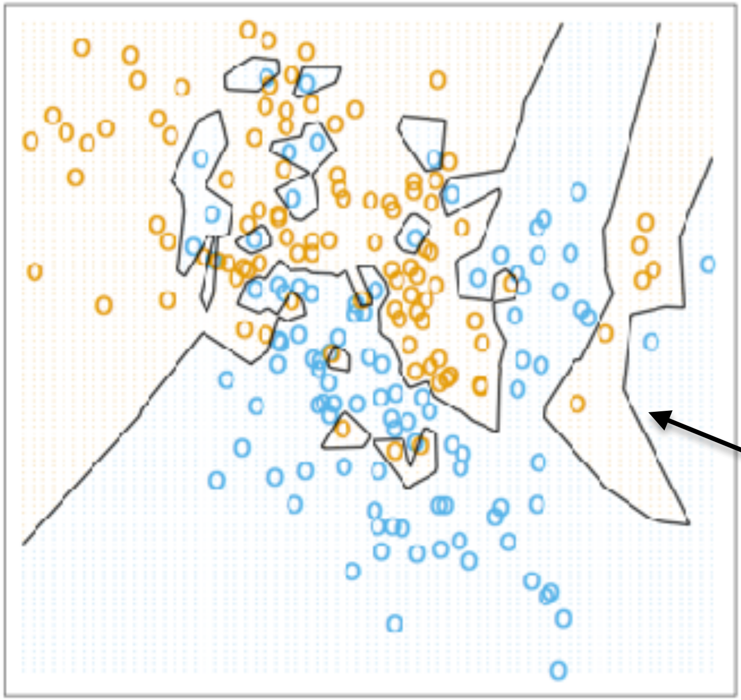
Learned:

15 nearest neighbor decision boundary (majority vote)


Predicted label: +1


Predicted label: -1

1 Nearest Neighbor Boundary



Training data:

True label: +1

True label: -1

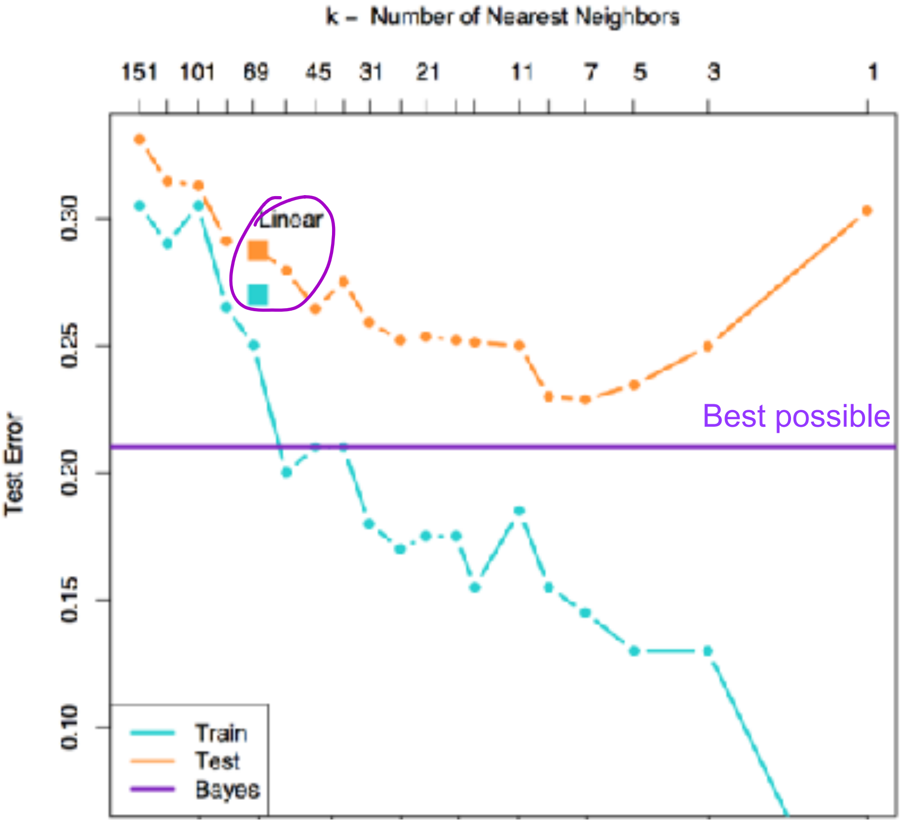
Learned:

1 nearest neighbor decision boundary (majority vote)

Predicted label: +1

Predicted label: -1

k-Nearest Neighbor Error



Bias-Variance tradeoff

As k->infinity?

Bias: increases

Variance: decreases

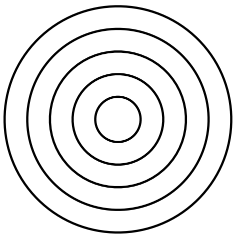
As k->1?

Bias:

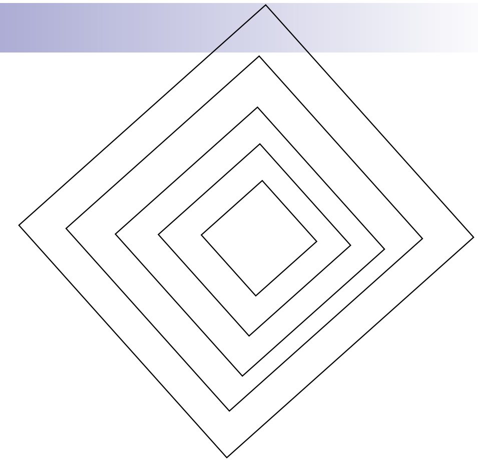
Variance:

Notable distance metrics (and their level sets)

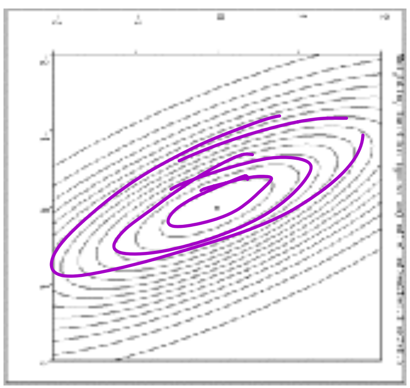
L_2 norm



$\|x\|_2^2 = x^T x$



L_1 norm (taxi-cab)

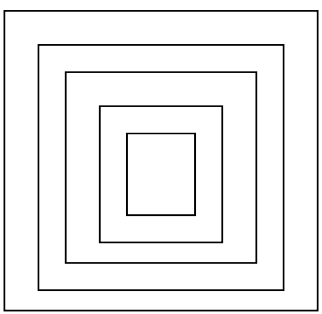


Mahalanobis

$x^T A x$

$\uparrow$

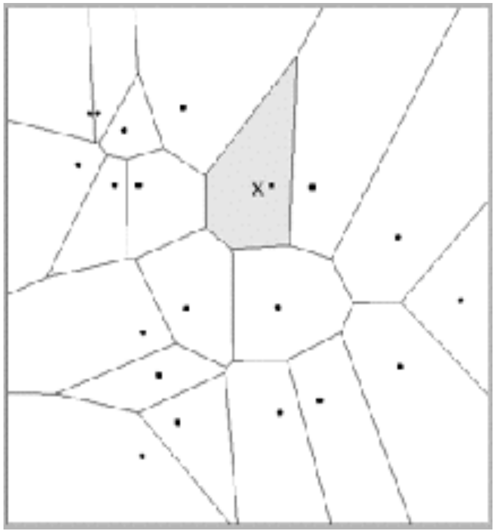
A symmetric positive definite



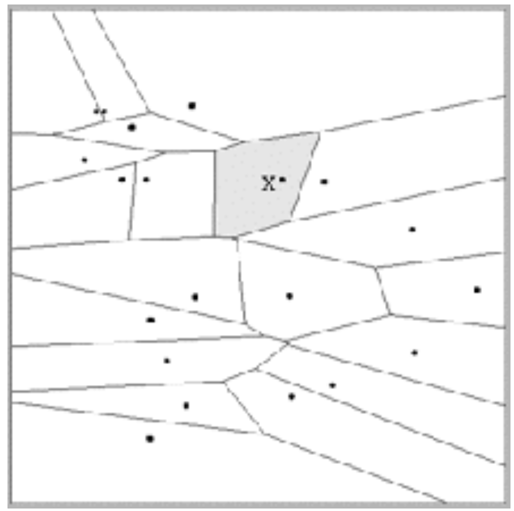
L-infinity *(max) norm*

1 nearest neighbor

One can draw the nearest-neighbor regions in input space.



$Dist(\mathbf{x}^i,\mathbf{x}^j) = (x^i_1 - x^j_1)^2 + (x^i_2 - x^j_2)^2$



$Dist(\mathbf{x}^i,\mathbf{x}^j) =(x^i_1 - x^j_1)^2+(3x^i_2 - 3x^j_2)^2$

The relative scalings in the distance metric affect region shapes

1 nearest neighbor guarantee

$$\{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d, y_i \in \{1, \dots, k\}$$

As $n \to \infty$ assume the x_i's become *dense* in $\mathbb{R}^d$ and $\mathbb{P}(Y = j | X = x)$ is *smooth*

As $x_a \to x_b$ we have $\mathbb{P}(Y_a = j) \to \mathbb{P}(Y_b = j)$ for all j

1 nearest neighbor guarantee

$$\{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d, y_i \in \{1, \ldots, k\}$$

As $n \to \infty$ assume the x_i's become *dense* in $\mathbb{R}^d$ and $\mathbb{P}(Y = j | X = x)$ is *smooth*

As $x_a \to x_b$ we have $\mathbb{P}(Y_a = j) \to \mathbb{P}(Y_b = j)$ for all j

If $p_\ell = \mathbb{P}(Y_a = \ell) = \mathbb{P}(Y_b = \ell)$ and $\ell^* = \arg \max_{\ell=1,\ldots,k} p_\ell$ then

$$\text{Bayes Error} = 1 - p_{\ell^*}$$

1 nearest neighbor guarantee

$$\{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d, y_i \in \{1, \ldots, k\}$$

As $n \to \infty$ assume the x_i's become *dense* in $\mathbb{R}^d$ and $\mathbb{P}(Y = j | X = x)$ is *smooth*

As $x_a \to x_b$ we have $\mathbb{P}(Y_a = j) \to \mathbb{P}(Y_b = j)$ for all j

If $p_\ell = \mathbb{P}(Y_a = \ell) = \mathbb{P}(Y_b = \ell)$ and $\ell^* = \arg\max_{\ell=1,\ldots,k} p_\ell$ then

$$\text{Bayes Error} = 1 - p_{\ell^*}$$

$$\text{1-nearest neighbor error} = \mathbb{P}(Y_a \neq Y_b) = \sum_{\ell=1}^{k} \mathbb{P}(Y_a = \ell, Y_b \neq \ell)$$

1 nearest neighbor guarantee

$$\{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d, y_i \in \{1, \dots, k\}$$

As $n \to \infty$ assume the x_i's become *dense* in $\mathbb{R}^d$ and $\mathbb{P}(Y = j | X = x)$ is *smooth*

As $x_a \to x_b$ we have $\mathbb{P}(Y_a = j) \to \mathbb{P}(Y_b = j)$ for all j

If $p_\ell = \mathbb{P}(Y_a = \ell) = \mathbb{P}(Y_b = \ell)$ and $\ell^* = \arg \max_{\ell = 1, \dots, k} p_\ell$ then

$$\text{Bayes Error} = 1 - p_{\ell^*}$$

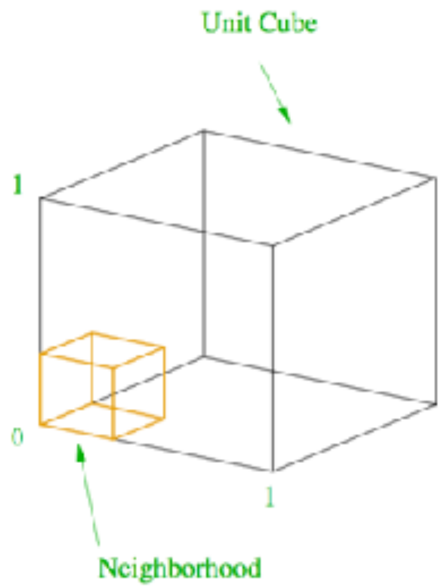
$$\text{1-nearest neighbor error} = \mathbb{P}(Y_a \neq Y_b) = \sum_{\ell=1}^k \mathbb{P}(Y_a = \ell, Y_b \neq \ell)$$

$$= \sum_{\ell=1}^k p_\ell (1 - p_\ell) \leq 2(1 - p_{\ell^*}) - \frac{k}{k-1}(1 - p_{\ell^*})^2$$

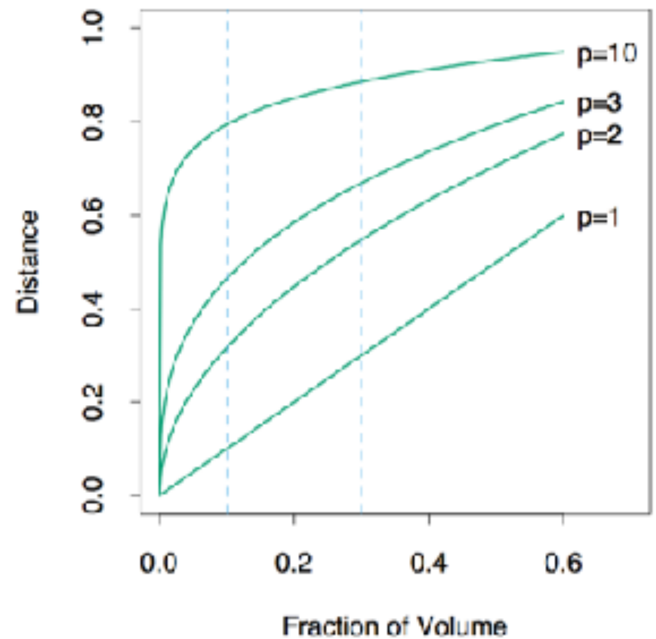
As n->infinity, then 1-NN rule error is at most twice the Bayes error!

[Cover, Hart, 1967]

Curse of dimensionality Ex. 1



side length r

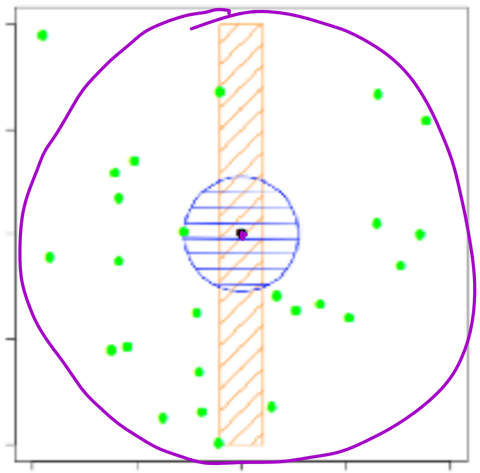


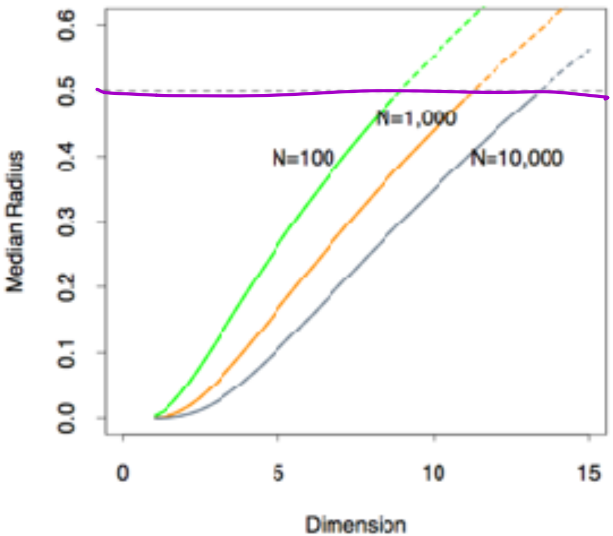
X is uniformly distributed over $[0,1]^p$. What is $\mathbb{P}(X \in [0,r]^p)$?

$= r^p = \frac{1}{2}$

Curse of dimensionality Ex. 2

$\{X_i\}_{i=1}^n$ are uniformly distributed over $[-.5, .5]^p$.

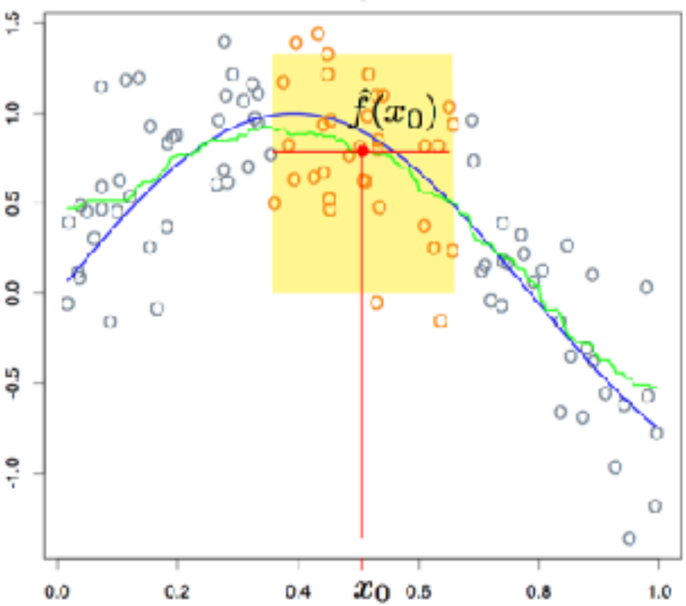




What is the median distance from a point at origin to its 1NN?

Nearest neighbor regression

$$\{(x_i, y_i)\}_{i=1}^n$$

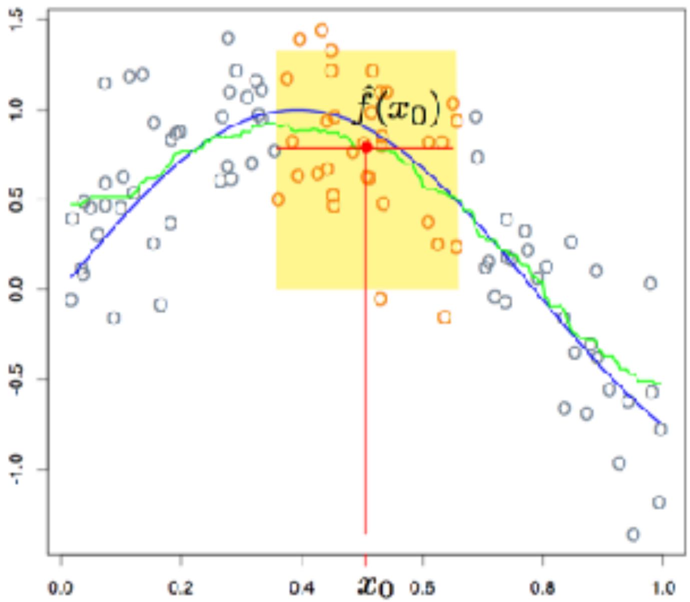


$$\mathcal{N}_k(x_0) = k\text{-nearest neighbors of } x_0$$

$$\widehat{f}(x_0) = \sum_{x_i \in \mathcal{N}_k(x_0)} \frac{1}{k}\, y_i$$

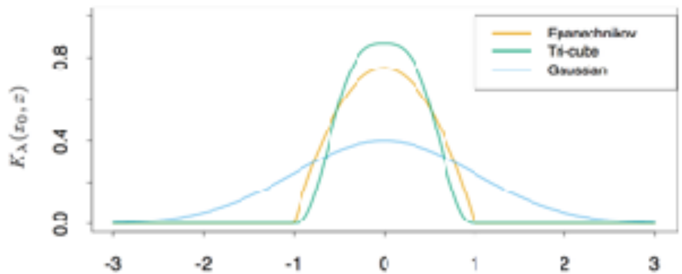
Nearest neighbor regression

$$\{(x_i, y_i)\}_{i=1}^n$$



Why are far-away neighbors weighted same as close neighbors!

Kernel smoothing: $K(x, y)$



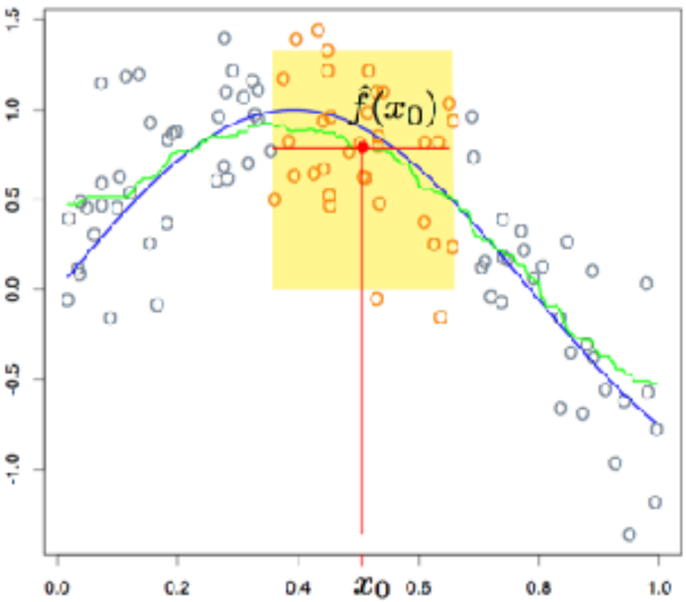
$\mathcal{N}_k(x_0) = k$-nearest neighbors of x_0

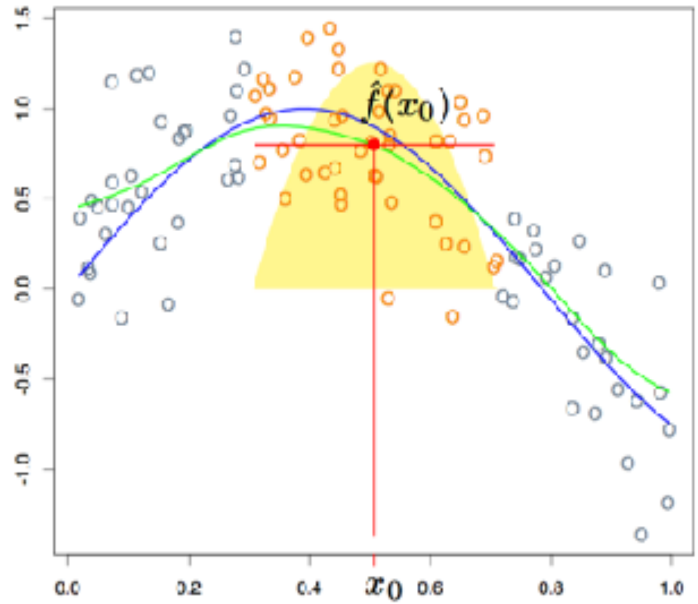
$$\widehat{f}(x_0) = \sum_{x_i \in \mathcal{N}_k(x_0)} \frac{1}{k} y_i$$

$$\widehat{f}(x_0) = \frac{\sum_{i=1}^n K(x_0, x_i) y_i}{\sum_{i=1}^n K(x_0, x_i)}$$

Nearest neighbor regression

$\{(x_i, y_i)\}_{i=1}^n$



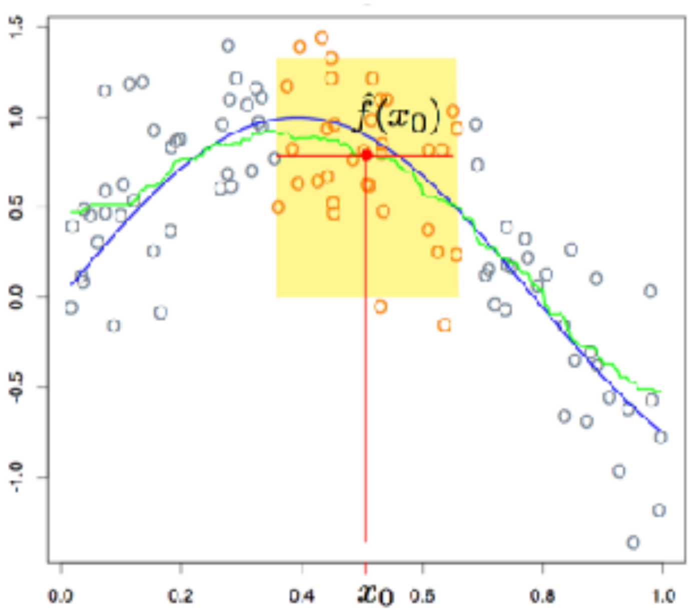


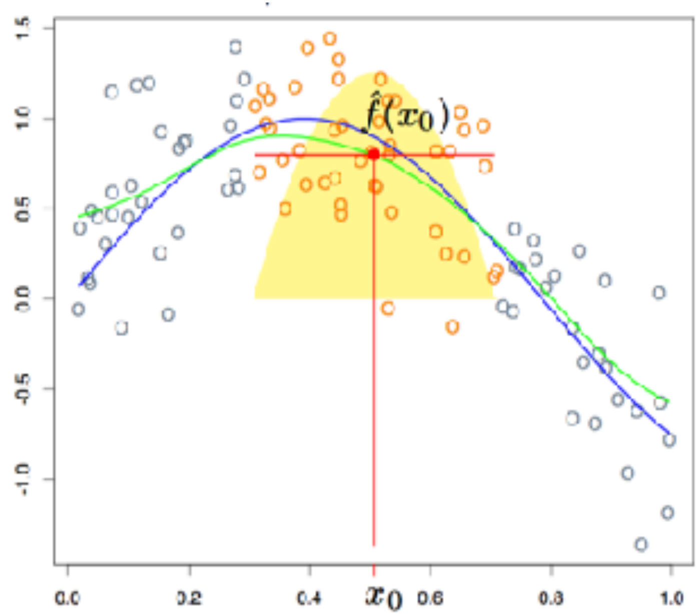
$\mathcal{N}_k(x_0) = k\text{-nearest neighbors of } x_0$

$$\widehat{f}(x_0) = \sum_{x_i \in \mathcal{N}_k(x_0)} \frac{1}{k} y_i \qquad \widehat{f}(x_0) = \frac{\sum_{i=1}^n K(x_0, x_i) y_i}{\sum_{i=1}^n K(x_0, x_i)}$$

Nearest neighbor regression

$\{(x_i, y_i)\}_{i=1}^n$





$\mathcal{N}_k(x_0) = k\text{-nearest neighbors of } x_0$

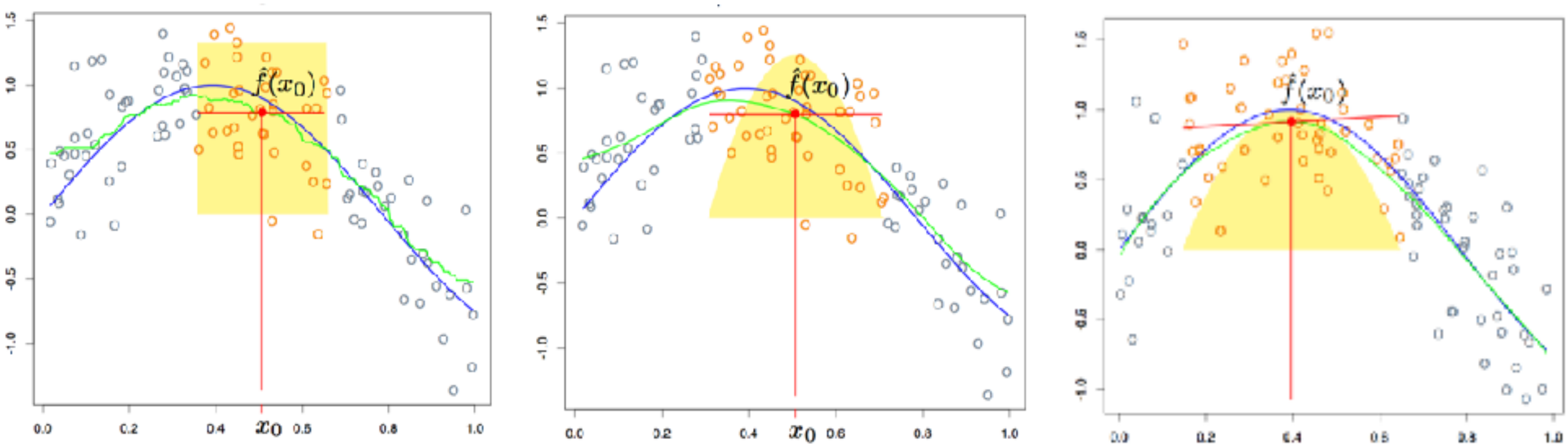
Why just average them?

$$\widehat{f}(x_0) = \sum_{x_i \in \mathcal{N}_k(x_0)} \frac{1}{k} y_i$$

$$\widehat{f}(x_0) = \frac{\sum_{i=1}^n K(x_0, x_i) y_i}{\sum_{i=1}^n K(x_0, x_i)}$$

Nearest neighbor regression

$$\{(x_i, y_i)\}_{i=1}^n$$



$\mathcal{N}_k(x_0) = k\text{-nearest neighbors of } x_0$

$$\widehat{f}(x_0) = \sum_{x_i \in \mathcal{N}_k(x_0)} \frac{1}{k} y_i$$

$$\widehat{f}(x_0) = \frac{\sum_{i=1}^n K(x_0, x_i) y_i}{\sum_{i=1}^n K(x_0, x_i)}$$

$$\widehat{f}(x_0) = b(x_0) + w(x_0)^T x_0$$

$$w(x_0), b(x_0) = \arg\min_{w,b} \sum_{i=1}^n K(x_0, x_i)(y_i - (b + w^T x_i))^2$$

Local Linear Regression

Nearest Neighbor Overview

- Very simple to explain and implement
- No training! But finding nearest neighbors in large dataset at test can be computationally demanding (kD-trees help)

Nearest Neighbor Overview

- Very simple to explain and implement
- No training! But finding nearest neighbors in large dataset at test can be computationally demanding (kD-trees help)
- You can use other forms of distance (not just Euclidean)
- Smoothing with Kernels and local linear regression can improve performance (at the cost of higher variance)

Nearest Neighbor Overview

- Very simple to explain and implement
- No training! But finding nearest neighbors in large dataset at test can be computationally demanding (kD-trees help)
- You can use other forms of distance (not just Euclidean)
- Smoothing with Kernels and local linear regression can improve performance (at the cost of higher variance)
- With a lot of data, “local methods” have strong, simple theoretical guarantees.
- Without a lot of data, neighborhoods aren’t “local” and methods suffer.

Kernels

Machine Learning – CSE546

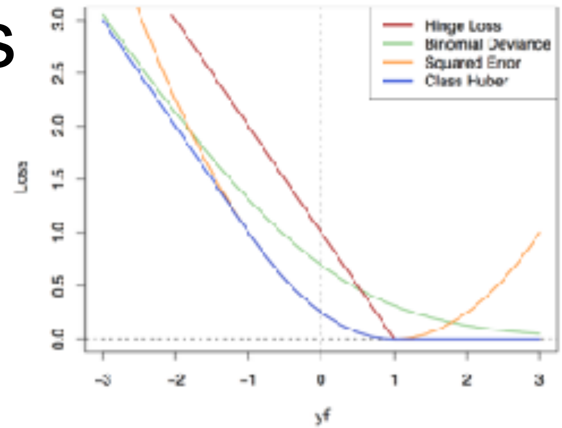
Kevin Jamieson

University of Washington

November 1

~~October 26~~, 2018

Machine Learning Problems



- Have a bunch of iid data of the form:

$$\{(x_i, y_i)\}_{i=1}^n \qquad x_i \in \mathbb{R}^d \qquad y_i \in \mathbb{R}$$

- Learning a model's parameters: $\sum_{i=1}^n \ell_i(w)$

  Each $\ell_i(w)$ is convex.

Hinge Loss: $\ell_i(w) = \max\{0, 1 - y_i \underbrace{x_i^T w}\}$

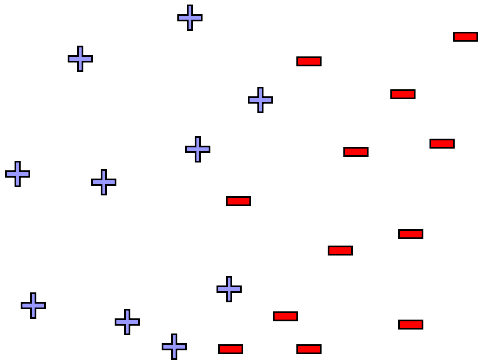
Logistic Loss: $\ell_i(w) = \log(1 + \exp(-y_i \underbrace{x_i^T w}))$

Squared error Loss: $\ell_i(w) = (y_i - \underbrace{x_i^T w})^2$

$\|x_i - x_j\|_2^2 = x_i^T x_i - 2 x_i^T x_j + x_j^T x_j$

All in terms of inner products! Even nearest neighbor can use inner products!

What if the data is not linearly separable?



Use features of features of features of features….

$$\phi(x) : \mathbb{R}^d \rightarrow \mathbb{R}^p$$

$$\phi(x) = \begin{bmatrix} x \\ x^2 \\ x^3 \\ \vdots \end{bmatrix}$$

Feature space can get really large really quickly!

Dot-product of polynomials

$\Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) =$ polynomials of degree exactly d

$$d = 1 : \phi(u) = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \qquad \langle \phi(u), \phi(v) \rangle = u_1 v_1 + u_2 v_2$$

Dot-product of polynomials

$\Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) =$ polynomials of degree exactly d

$$d = 1 : \phi(u) = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \qquad \langle \phi(u), \phi(v) \rangle = u_1 v_1 + u_2 v_2$$

$$d = 2 : \phi(u) = \begin{bmatrix} u_1^2 \\ u_2^2 \\ u_1 u_2 \\ u_2 u_1 \end{bmatrix} \qquad \langle \phi(u), \phi(v) \rangle = u_1^2 v_1^2 + u_2^2 v_2^2 + 2 u_1 u_2 v_1 v_2$$

$$= (u^T v)^2$$

Dot-product of polynomials

$\Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) =$ polynomials of degree exactly d

$$d = 1 : \phi(u) = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \qquad \langle \phi(u), \phi(v) \rangle = u_1 v_1 + u_2 v_2$$

$$d = 2 : \phi(u) = \begin{bmatrix} u_1^2 \\ u_2^2 \\ u_1 u_2 \\ u_2 u_1 \end{bmatrix} \qquad \langle \phi(u), \phi(v) \rangle = u_1^2 v_1^2 + u_2^2 v_2^2 + 2 u_1 u_2 v_1 v_2$$

$= (u^T v)^2$

General d : $(u^T v)^d$

Dimension of $\phi(u)$ is roughly p^d if $u \in \mathbb{R}^p$

Kernel Trick

$$\widehat{w} = \arg\min_w \sum_{i=1}^n (y_i - x_i^T w)^2 + \lambda ||w||_w^2$$

There exists an $\alpha \in \mathbb{R}^n$: $\widehat{w} = \sum_{i=1}^n \alpha_i x_i$ Why?

$$\widehat{\alpha} = \arg\min_\alpha \sum_{i=1}^n (y_i - \sum_{j=1}^n \alpha_j \langle x_j, x_i \rangle)^2 + \lambda \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j \langle x_i, x_j \rangle$$

Kernel Trick

$$\widehat{w} = \arg\min_w \sum_{i=1}^n (y_i - x_i^T w)^2 + \lambda ||w||_w^2$$

There exists an $\alpha \in \mathbb{R}^n$: $\widehat{w} = \sum_{i=1}^n \alpha_i x_i$ Why?

$$\begin{aligned}
\widehat{\alpha} &= \arg\min_\alpha \sum_{i=1}^n (y_i - \sum_{j=1}^n \alpha_j \langle x_j, x_i \rangle)^2 + \lambda \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j \langle x_i, x_j \rangle \\
&= \arg\min_\alpha \sum_{i=1}^n (y_i - \sum_{j=1}^n \alpha_j K(x_i, x_j))^2 + \lambda \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j K(x_i, x_j) \\
&= \arg\min_\alpha ||\mathbf{y} - \mathbf{K}\alpha||_2^2 + \lambda \alpha^T \mathbf{K} \alpha
\end{aligned}$$

$$K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$$

Why regularization?

Typically, $\mathbf{K} \succ 0$. What if $\lambda = 0$?

$$\widehat{\alpha} = \arg\min_{\alpha} ||\mathbf{y} - \mathbf{K}\alpha||_2^2 + \lambda \alpha^T \mathbf{K} \alpha$$

Why regularization?

Typically, $\mathbf{K} \succ 0$. What if $\lambda = 0$?

$$\widehat{\alpha} = \arg\min_{\alpha} ||\mathbf{y} - \mathbf{K}\alpha||_2^2 + \lambda \alpha^T \mathbf{K} \alpha$$

Unregularized kernel least squares can (over) fit **any data**!

$$\widehat{\alpha} = \mathbf{K}^{-1}\mathbf{y}$$

Common kernels

- Polynomials of degree exactly d

$$K(\mathbf{u},\mathbf{v}) = (\mathbf{u}\cdot\mathbf{v})^d$$

- Polynomials of degree up to d

$$K(\mathbf{u},\mathbf{v}) = (\mathbf{u}\cdot\mathbf{v}+1)^d$$

- Gaussian (squared exponential) kernel

$$K(\mathbf{u},\mathbf{v}) = \exp\left(-\frac{||\mathbf{u}-\mathbf{v}||_2^2}{2\sigma^2}\right)$$

- Sigmoid

$$K(\mathbf{u},\mathbf{v}) = \tanh(\eta\mathbf{u}\cdot\mathbf{v}+\nu)$$

Mercer's Theorem

- When do we have a valid Kernel K(x,x')?
- Definition 1: when it is an inner product

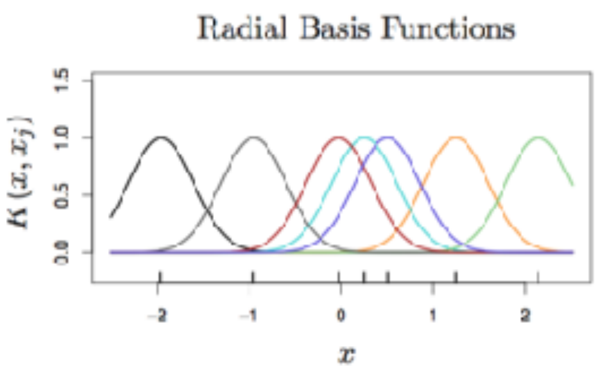
- Mercer's Theorem:
  - K(x,x') is a valid kernel if and only if K is a positive semi-definite.
  - PSD in the following sense:

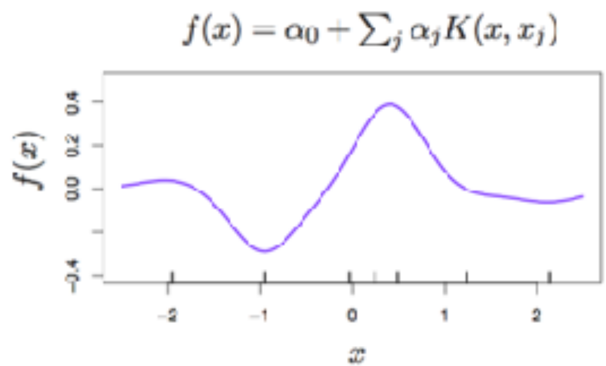
$$\int_{x,x'} h(x)K(x,x')h(x')dxdx' \geq 0 \quad \forall h : \mathbb{R}^d \to \mathbb{R}, \int_x |h(x)|^2 dx \leq \infty$$

RBF Kernel

$$K(\mathbf{u},\mathbf{v}) = \exp\left(-\frac{||\mathbf{u}-\mathbf{v}||_2^2}{2\sigma^2}\right)$$

- Note that this is like weighting “bumps” on each point like kernel smoothing but now we **learn** the weights



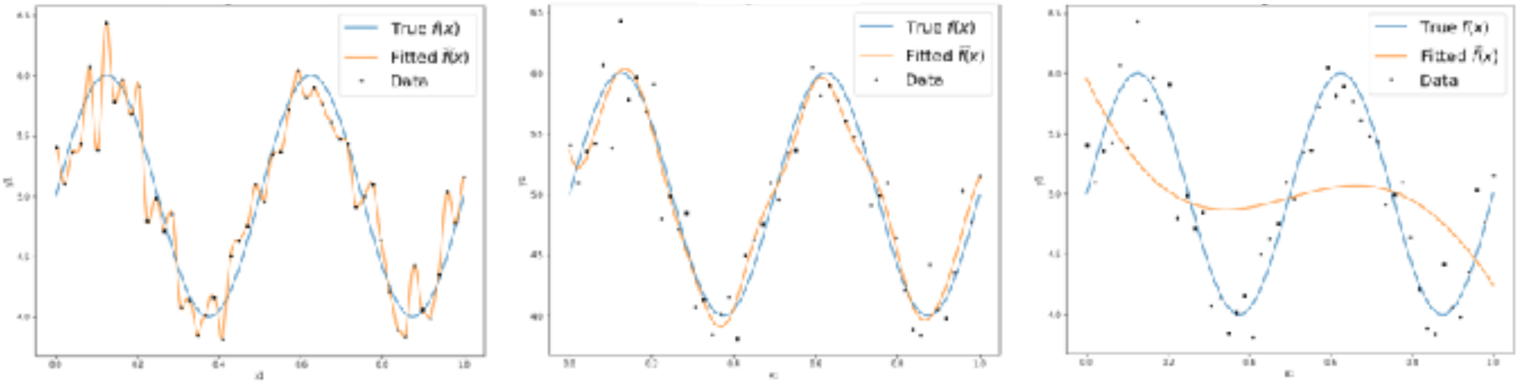


RBF Kernel

$$K(\mathbf{u},\mathbf{v}) = \exp\left(-\frac{||\mathbf{u}-\mathbf{v}||_2^2}{2\sigma^2}\right)$$

The bandwidth sigma has an enormous effect on fit:

$\sigma = 10^{-2}\quad \lambda = 10^{-4}$ $\sigma = 10^{-1}\quad \lambda = 10^{-4}$ $\sigma = 10^{-0}\quad \lambda = 10^{-4}$



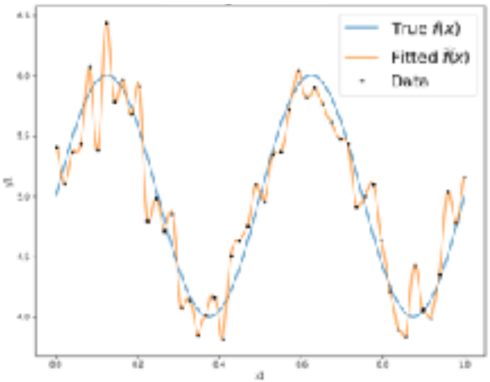
$$\widehat{f}(x) = \sum_{i=1}^{n} \widehat{\alpha}_i K(x_i, x)$$

RBF Kernel

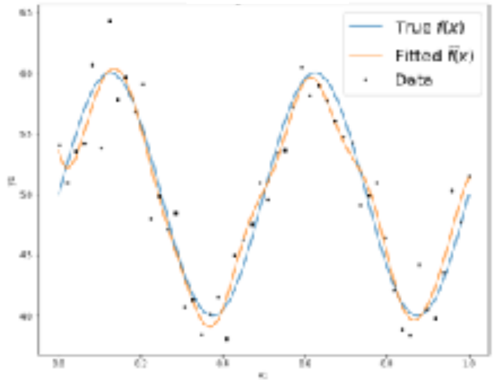
$$K(\mathbf{u},\mathbf{v}) = \exp\left(-\frac{||\mathbf{u}-\mathbf{v}||_2^2}{2\sigma^2}\right)$$

The bandwidth sigma has an enormous effect on fit:

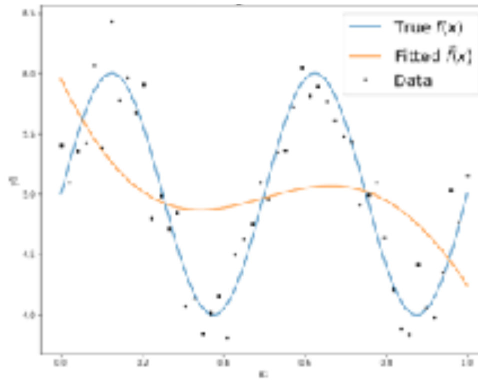
$\sigma = 10^{-2}\quad \lambda = 10^{-4}$



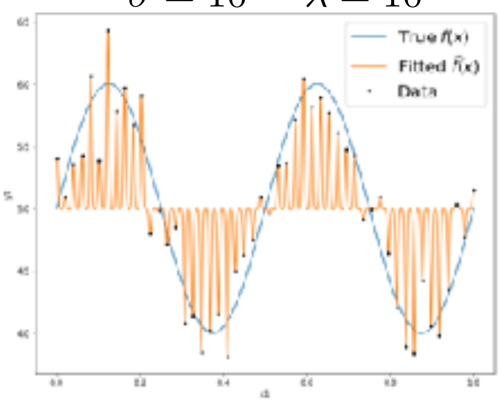
$\sigma = 10^{-1}\quad \lambda = 10^{-4}$



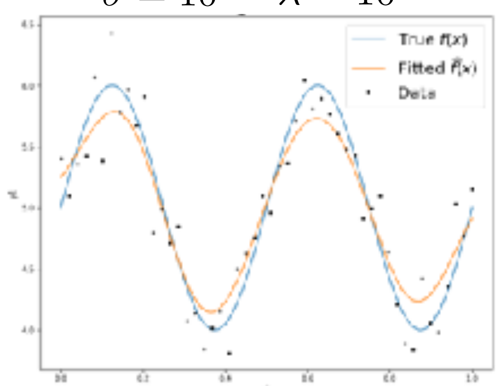
$\sigma = 10^{-0}\quad \lambda = 10^{-4}$



$\sigma = 10^{-3}\quad \lambda = 10^{-4}$



$\sigma = 10^{-1}\quad \lambda = 10^{-0}$



$$\widehat{f}(x) = \sum_{i=1}^{n} \widehat{\alpha}_i K(x_i, x)$$

RBF kernel and random features

$$2\cos(\alpha)\cos(\beta) = \cos(\alpha+\beta) + \cos(\alpha-\beta)$$

$$e^{jz} = \cos(z) + \sin(z)$$

Recall HW1 where we used the feature map:

$$\phi(x) = \begin{bmatrix} \sqrt{2}\cos(w_1^T x + b_1) \\ \vdots \\ \sqrt{2}\cos(w_p^T x + b_p) \end{bmatrix}$$

$$w_k \sim \mathcal{N}(0, 2\gamma\, I)$$

$$b_k \sim \text{uniform}(0, \pi)$$

$$\mathbb{E}[\frac{1}{p}\phi(x)^T\phi(y)] = \frac{1}{p}\sum_{k=1}^{p} \mathbb{E}[2\cos(w_k^T x + b_k)\cos(w_k^T y + b_k)]$$

$$= \mathbb{E}_{w,b}[2\cos(w^T x + b)\cos(w^T y + b)]$$

RBF kernel and random features

$$2\cos(\alpha)\cos(\beta) = \cos(\alpha+\beta) + \cos(\alpha-\beta)$$

$$e^{jz} = \cos(z) + \sin(z)$$

Recall HW1 where we used the feature map:

$$\phi(x) = \begin{bmatrix} \sqrt{2}\cos(w_1^T x + b_1) \\ \vdots \\ \sqrt{2}\cos(w_p^T x + b_p) \end{bmatrix} \qquad \begin{aligned} w_k &\sim \mathcal{N}(0, 2\gamma\, I) \\ b_k &\sim \text{uniform}(0, \pi) \end{aligned}$$

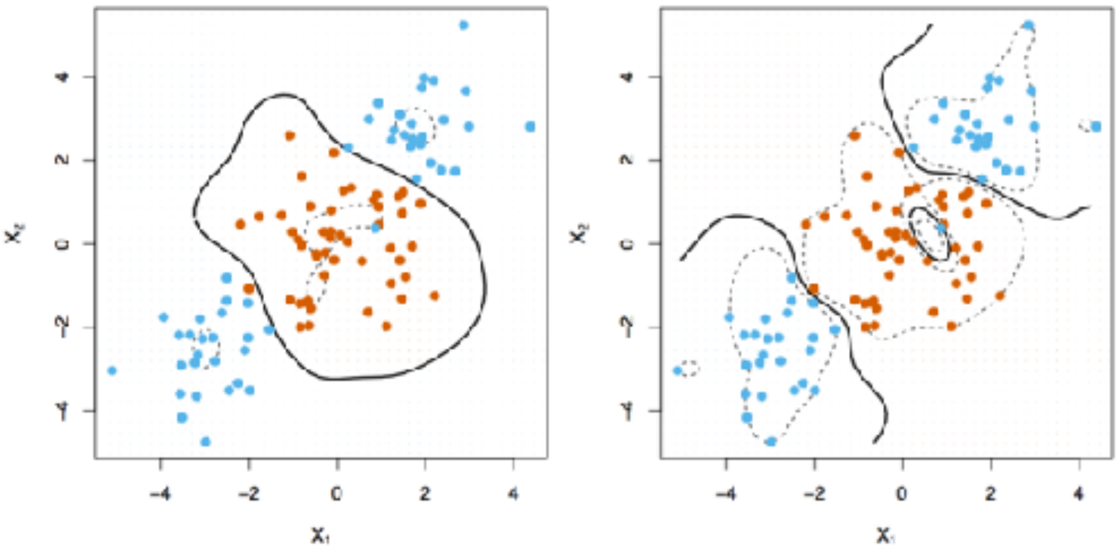
$$\begin{aligned} \mathbb{E}[\frac{1}{p}\phi(x)^T\phi(y)] &= \frac{1}{p}\sum_{k=1}^{p}\mathbb{E}[2\cos(w_k^T x + b_k)\cos(w_k^T y + b_k)] \\ &= \mathbb{E}_{w,b}[2\cos(w^T x + b)\cos(w^T y + b)] \\ &= e^{-\gamma||x-y||_2^2} \end{aligned}$$

[Rahimi, Recht NIPS 2007]
“NIPS Test of Time Award, 2018”

RBF Classification

$$\widehat{w} = \sum_{i=1}^{n} \max\{0, 1 - y_i(b + x_i^T w)\} + \lambda ||w||_2^2$$

$$\min_{\alpha,b} \sum_{i=1}^{n} \max\{0, 1 - y_i(b + \sum_{j=1}^{n} \alpha_j \langle x_i, x_j \rangle)\} + \lambda \sum_{i,j=1}^{n} \alpha_i \alpha_j \langle x_i, x_j \rangle$$



Wait, infinite dimensions?

- Isn't everything separable there? How are we not overfitting?

- Regularization! Fat shattering (R/margin)^2

String Kernels

Example from Efron and Hastie, 2016

Amino acid sequences of different lengths:

x1

IPTSALVKETLALLSTHRTLLIANETLRIPVPVHKNHQLCTEEIFQGIGTLESQTVQGGTV
ERLFKNLSLIKKYIDGQKKKCGEERRRVNQFLDY**LQE**FLGVMNTEWI

x2

PHRRDLCSRSIWLARKIRSDLTALTESYVKHQGLWSELTEAER**LQE**NLQAYRTFHVLLA
RLLEDQQVHFTPTEGDFHQAIHTLLLQVAAFAYQIEELMILLEYKIPRNEADGMLFEKK
LWGLKV**LQE**LSQWTVRSIHDLRFISSHQTGIP

All subsequences of length 3 (of possible 20 amino acids) $20^3 = 8{,}000$

$$h^3_{\text{LQE}}(x_1) = 1 \text{ and } h^3_{\text{LQE}}(x_2) = 2.$$