Announcements

- Project feedback

As stated in the project description and multiple times in class:

- You must have data at the time of the proposal.
- The project must contain real data (not just synthetic).
- 1 page maximum

Use spell check.
Clearly define metrics that will drive your development.
Please submit a proposal per person (for grading). It won't be marked late, obviously, just for book keeping.
If you have a partner, compare notes on feedback (usually only gave it once)

Recap: Nearest Neighbor

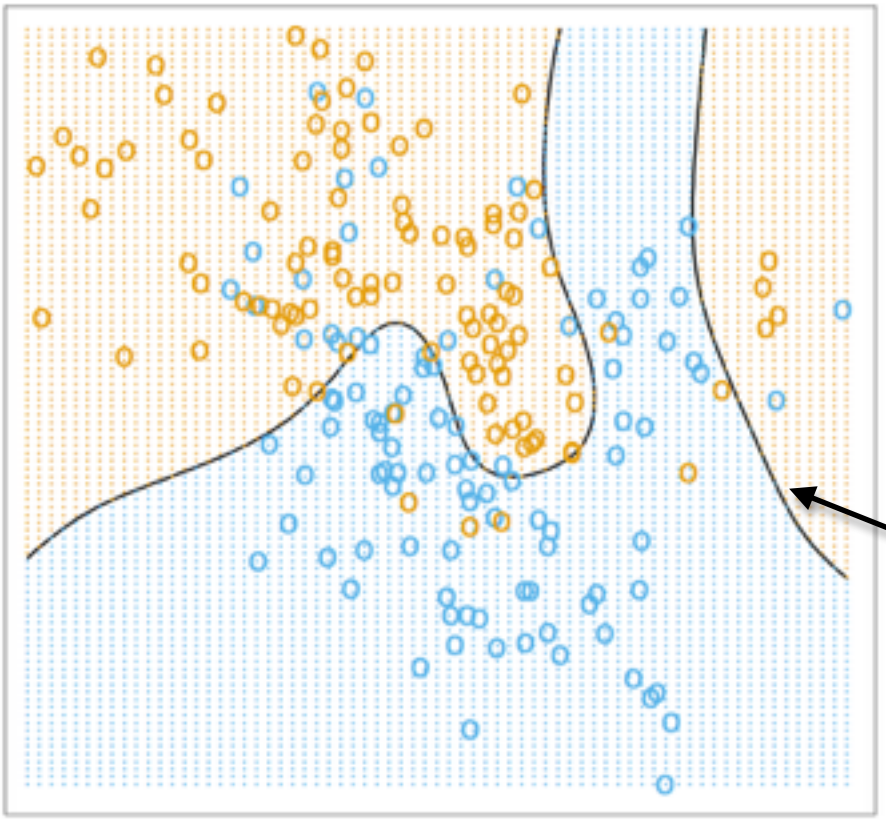
Machine Learning – CSE546

Kevin Jamieson

University of Washington

October 26, 2017

Some data, Bayes Classifier



Training data:

True label: +1

True label: -1

Optimal “Bayes” classifier:

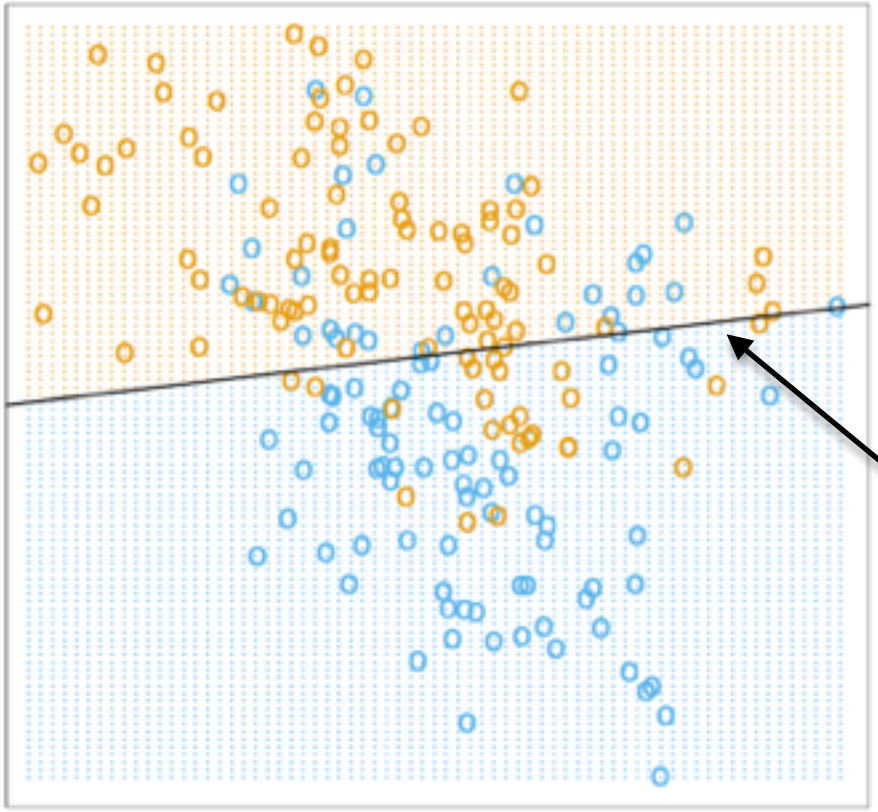
$$\mathbb{P}(Y = 1|X = x) = \frac{1}{2}$$

Predicted label: +1

Predicted label: -1

Figures stolen from Hastie et al

Linear Decision Boundary



Training data:

- True label: +1
- True label: -1

Learned:

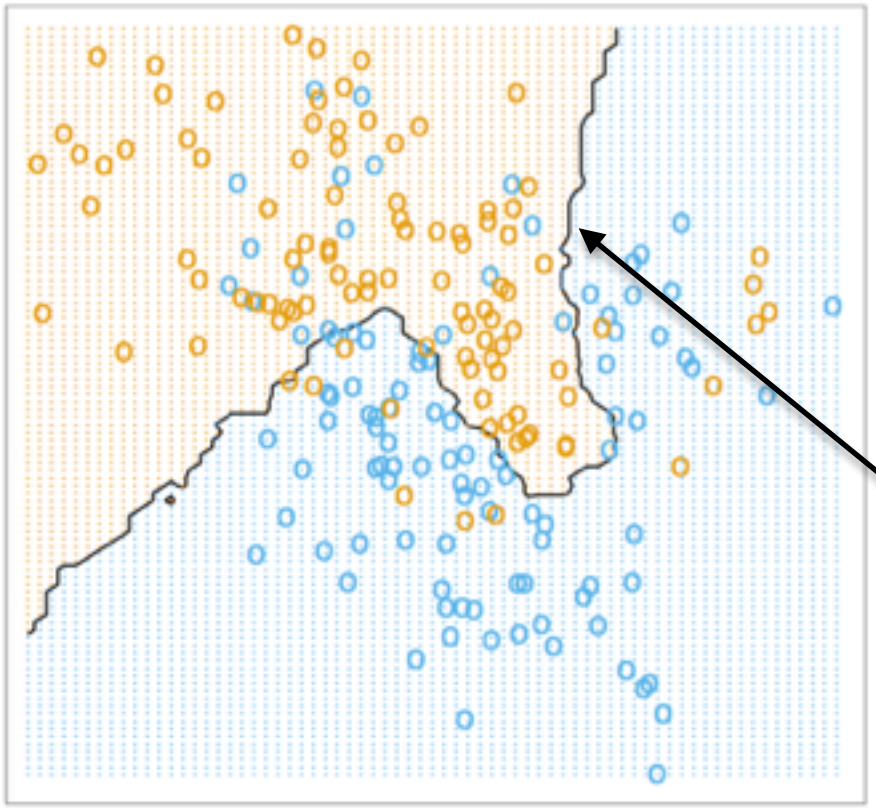
Linear Decision boundary

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

- Predicted label: +1
- Predicted label: -1

Figures stolen from Hastie et al

15 Nearest Neighbor Boundary



Training data:

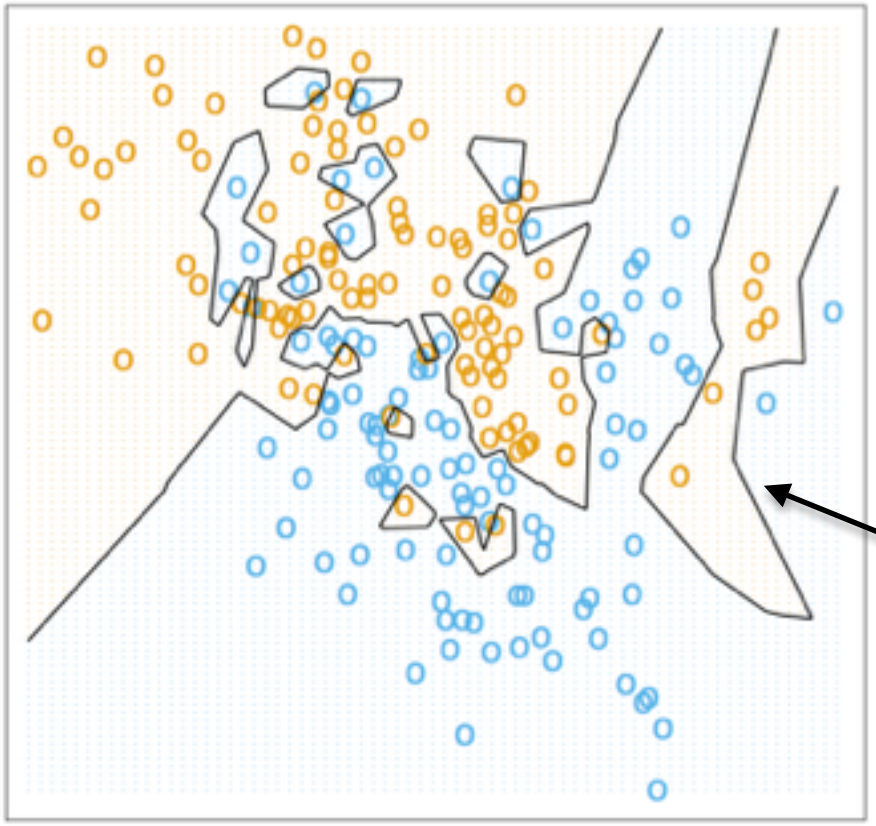
- True label: +1
- True label: -1

Learned:

15 nearest neighbor decision boundary (majority vote)

- Predicted label: +1
- Predicted label: -1

1 Nearest Neighbor Boundary



Training data:

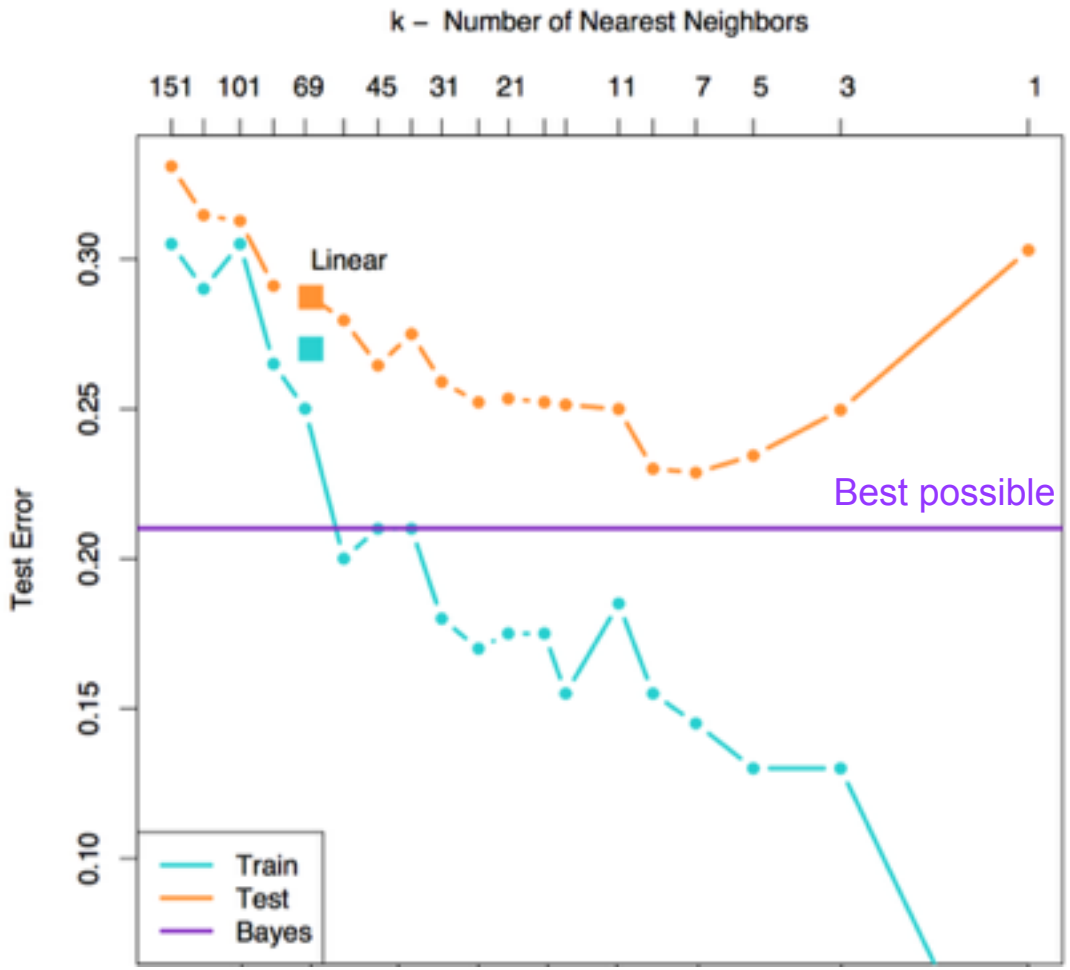
- True label: +1
- True label: -1

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:

Variance:

As k->1?

Bias:

Variance:

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 $\mathbb{R}^d$

Note: any $x_a \in \mathbb{R}^d$ has the same label distribution as x_b with $b = 1NN(a)$

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

$$\text{Bates error} = 1 - p_{\ell^*}$$

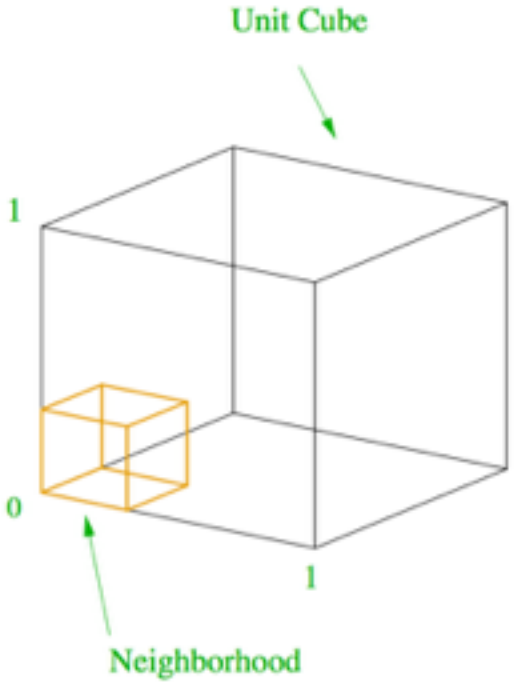
$$\text{1-nearest neighbor error} = \mathbb{P}(Y_a \neq Y_b) = \sum_{\ell=1}^{k} \mathbb{P}(Y_a = \ell, Y_b \neq \ell)$$

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

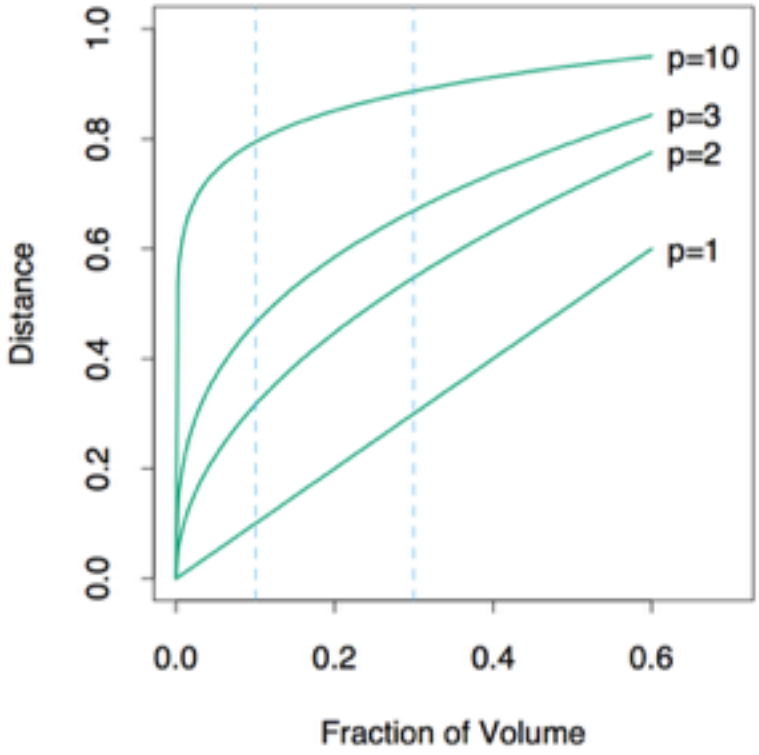
As x->infinity, then 1-NN rule error is at most twice the Bayes error!

[Cover, Hart, 1967]

Curse of dimensionality Ex. 1



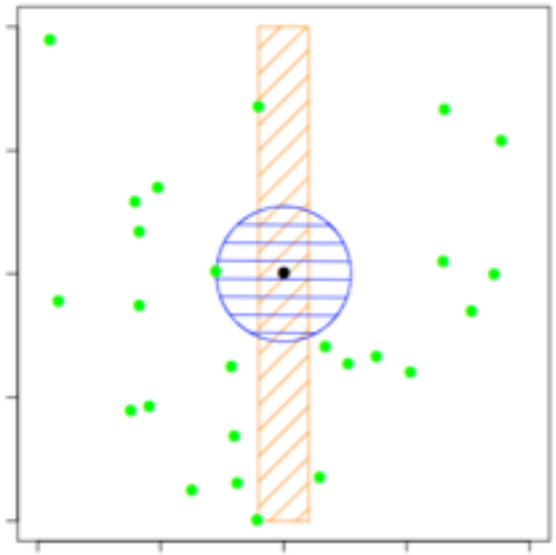
side length r

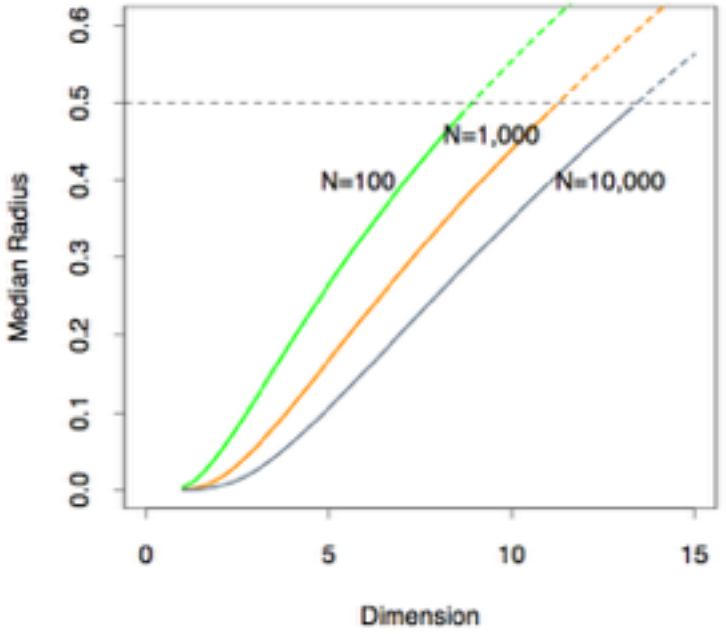


X is uniformly distributed over $[0,1]^p$. What is $\mathbb{P}(X \in [0,r]^p)$?

Curse of dimensionality Ex. 2

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

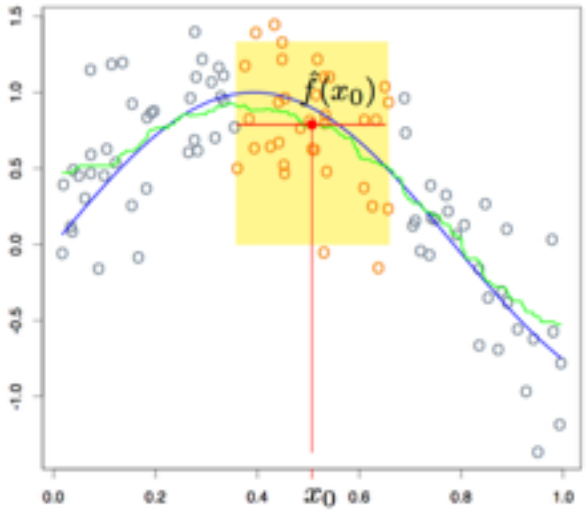


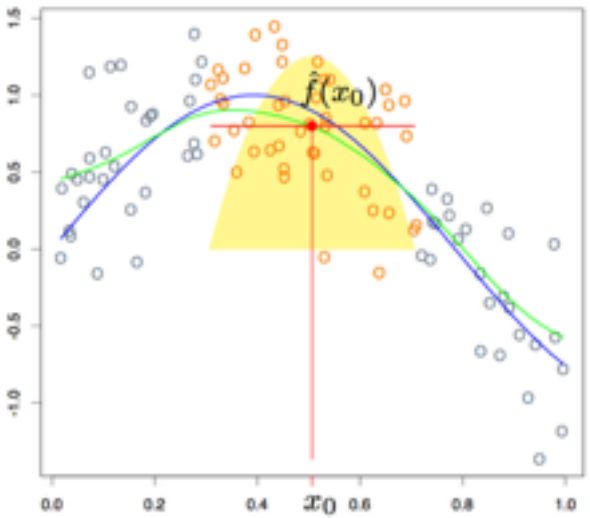


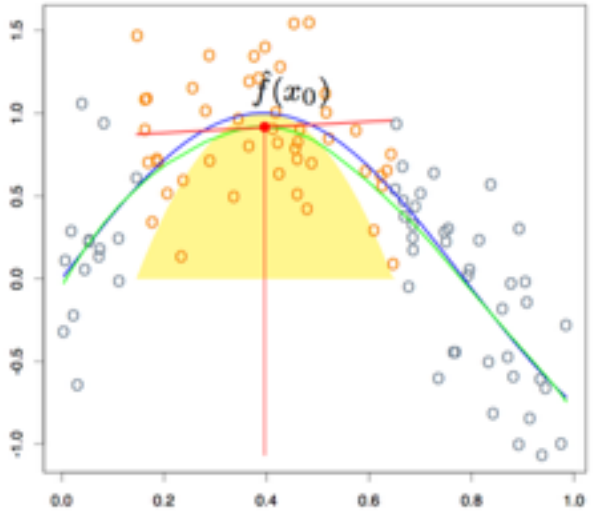
What is the median distance from a point at origin to its 1NN?

Nearest neighbor regression

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







$\mathcal{N}_k(x_0) = k$-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)
- 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. With not a lot of data, neighborhoods aren’t “local” and methods suffer.

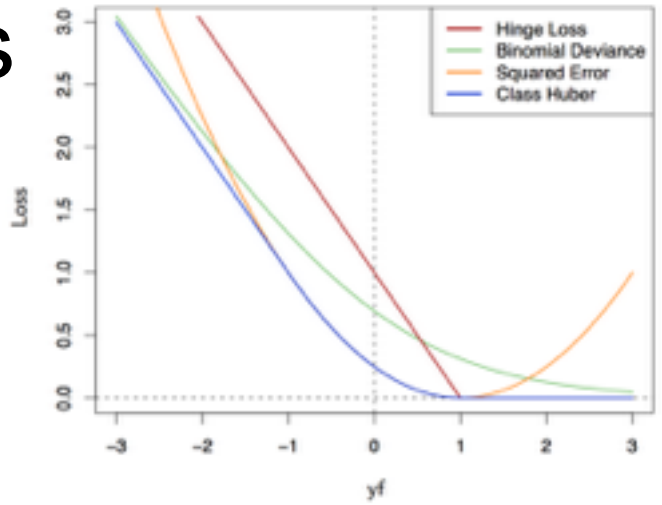
Kernels

Machine Learning – CSE546

Kevin Jamieson
University of Washington

October 26, 2017

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.

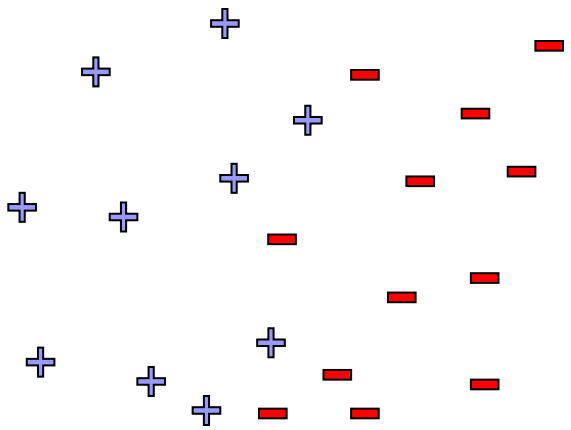
Hinge Loss: $\ell_i(w) = \max\{0, 1 - y_i x_i^T w\}$

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

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

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….

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

Feature space can get really large really quickly!

Dot-product of polynomials

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

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

Dot-product of polynomials

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

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

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

Dot-product of polynomials

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

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

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

General 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?

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

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

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

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

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

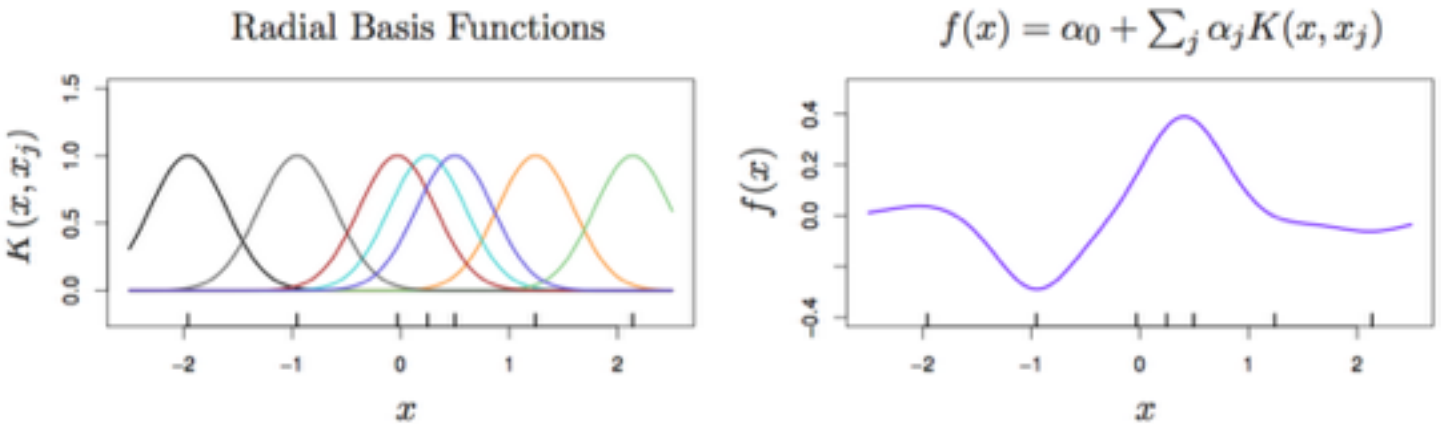
- Mercer's Theorem:
  - K(x,x') is a valid kernel if and only if K is a positive semi-definite.
  - PSD in the following sense:

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

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

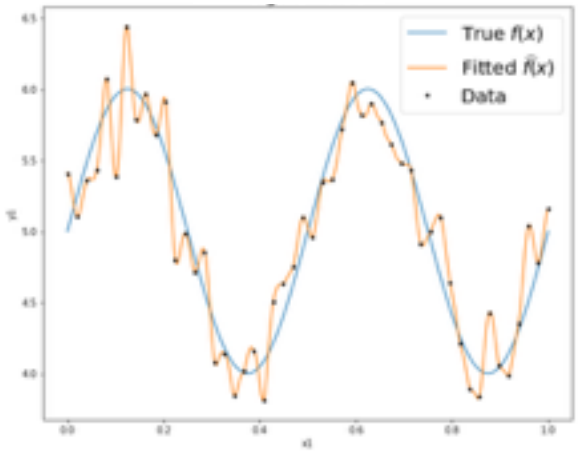


RBF Kernel

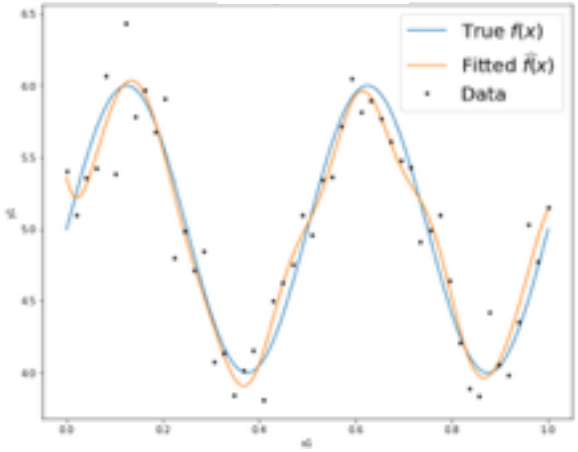
$$K(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{||\mathbf{u} - \mathbf{v}||_2^2}{2\sigma^2}\right)$$

The bandwidth sigma has an enormous effect on fit:

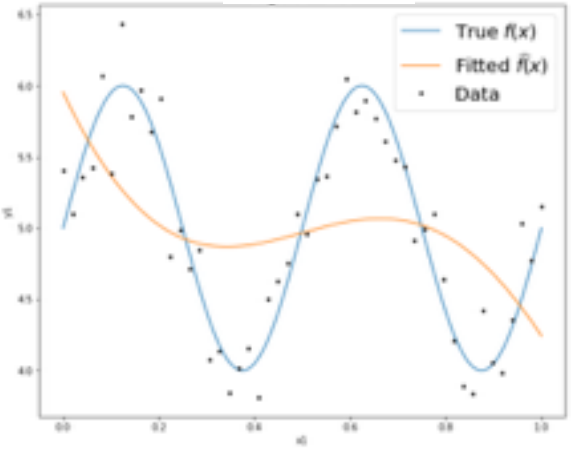
$\sigma = 10^{-2}$ $\lambda = 10^{-4}$



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



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



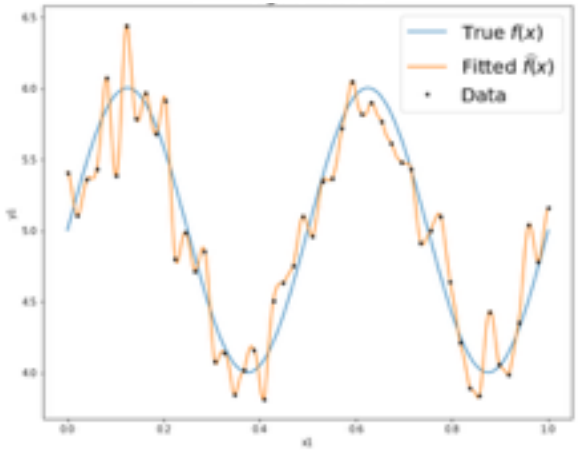
$$\widehat{f}(x) = \sum_{i=1}^{n} \widehat{\alpha}_i K(x_i, x)$$

RBF Kernel

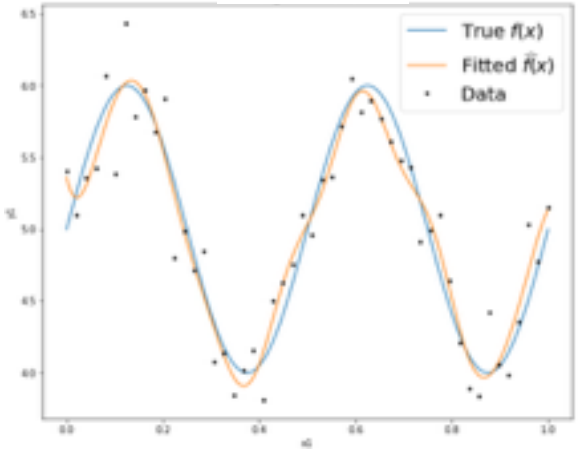
$$K(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{||\mathbf{u} - \mathbf{v}||_2^2}{2\sigma^2}\right)$$

The bandwidth sigma has an enormous effect on fit:

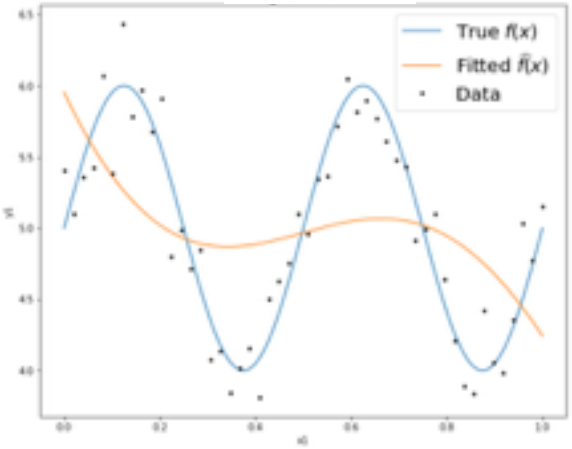
$\sigma = 10^{-2}$ $\lambda = 10^{-4}$



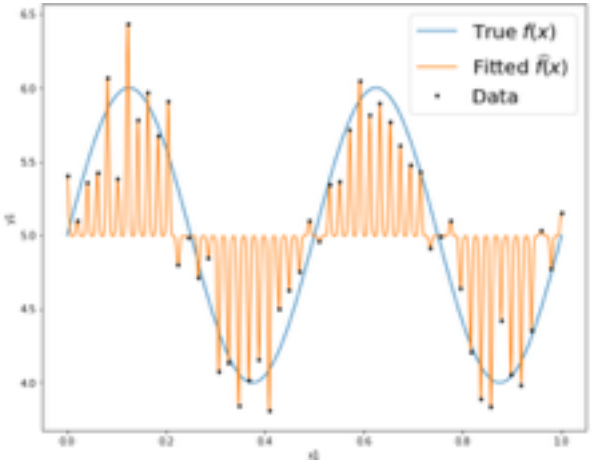
$\sigma = 10^{-1}$ $\lambda = 10^{-4}$



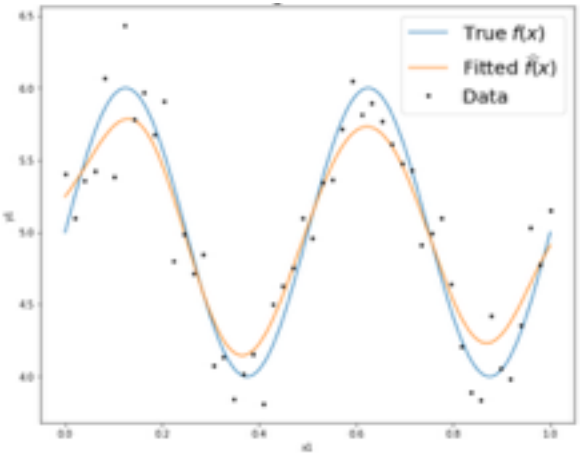
$\sigma = 10^{-0}$ $\lambda = 10^{-4}$



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



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

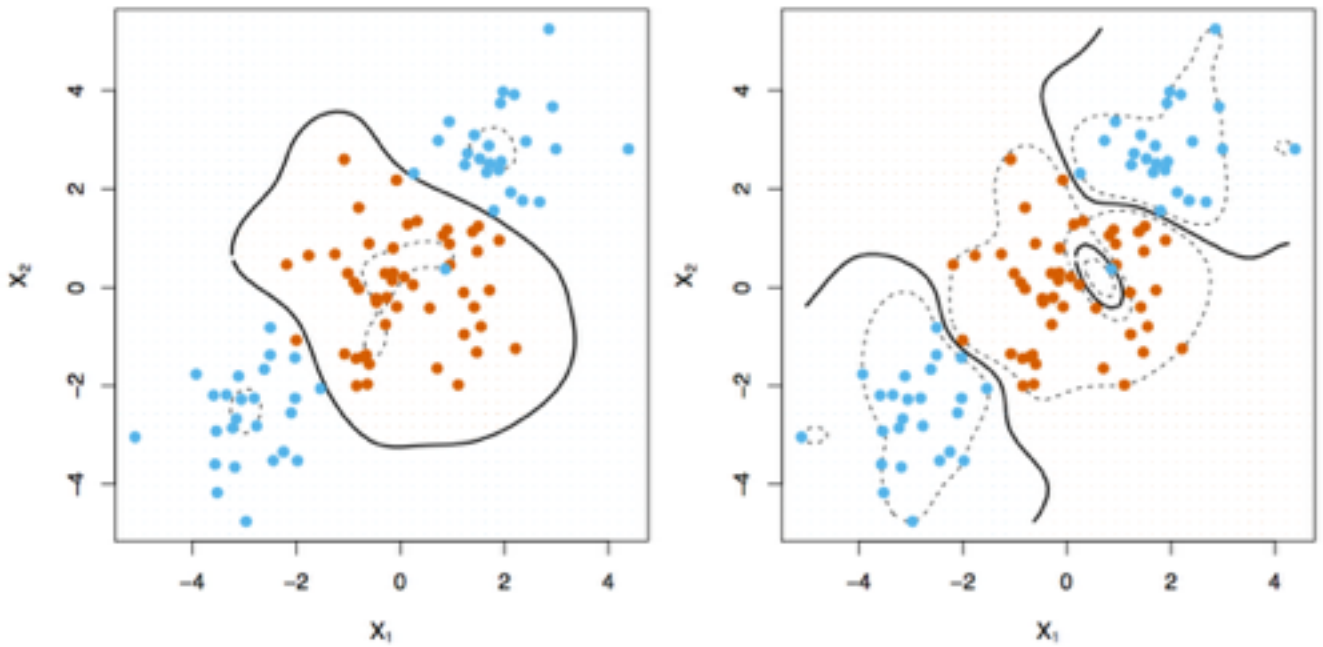


$$\widehat{f}(x) = \sum_{i=1}^{n} \widehat{\alpha}_i K(x_i, x)$$

RBF Classification

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

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



RBF kernel Secretly random features

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

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

$$b \sim \text{uniform}(0, \pi) \qquad w \sim \mathcal{N}(0, 2\gamma)$$

$$\phi(x) = \sqrt{2}\cos(w^T x + b)$$

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

RBF kernel Secretly random features

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

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

$$b \sim \text{uniform}(0, \pi) \qquad w \sim \mathcal{N}(0, 2\gamma)$$

$$\phi(x) = \sqrt{2}\cos(w^T x + b)$$

$$\mathbb{E}_{w,b}[\phi(x)^T \phi(y)] = e^{-\gamma||x-y||_2^2}$$

[Rahimi, Recht 2007]

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) $20^3 = 8{,}000$

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

Trees

Machine Learning – CSE546

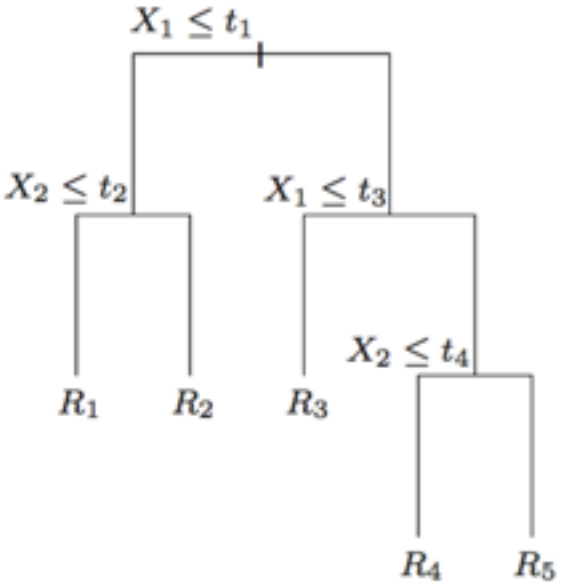
Kevin Jamieson
University of Washington

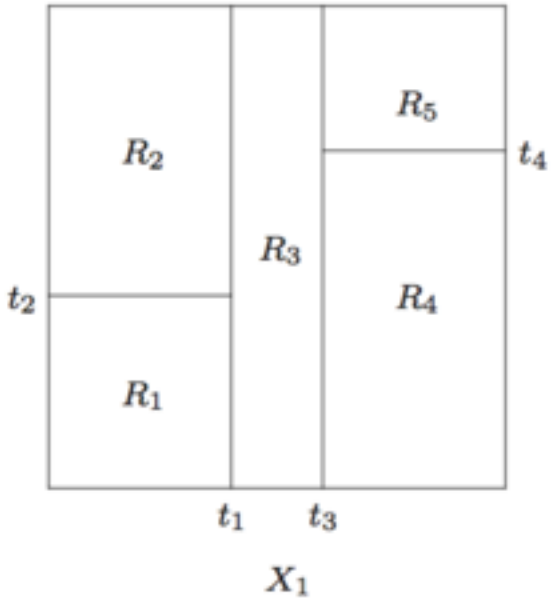
October 26, 2017

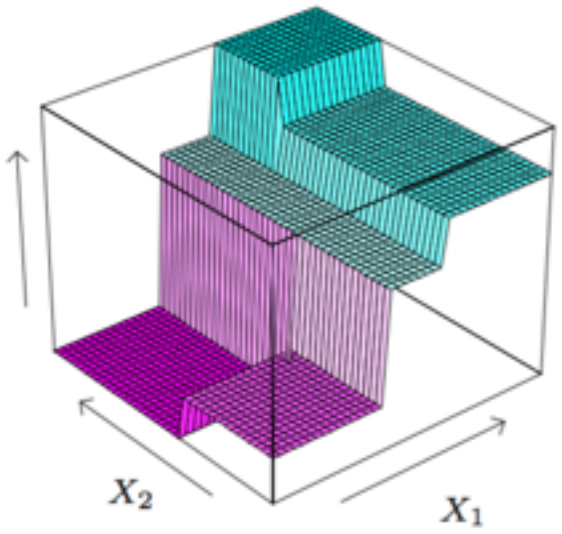
Trees

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m).$$

Build a binary tree, splitting along axes



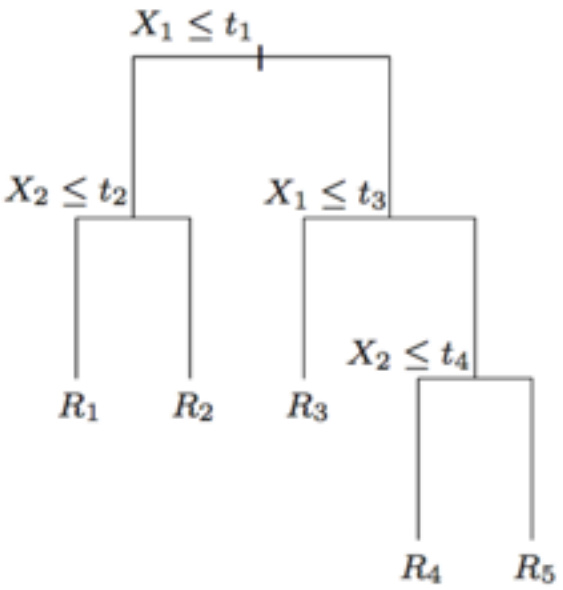




Trees

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m).$$

Build a binary tree, splitting along axes

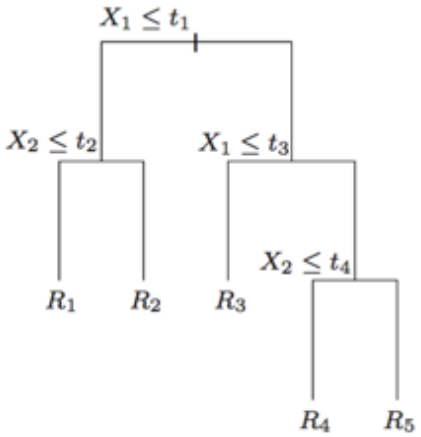


How do you split?

When do you stop?

Learning decision trees

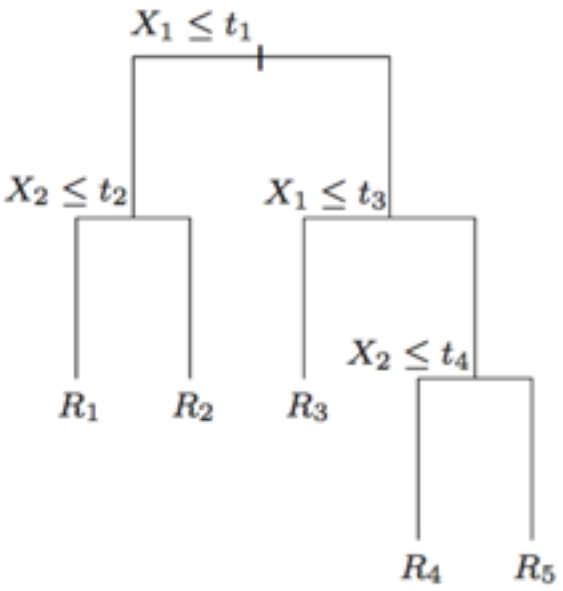
- Start from empty decision tree
- Split on **next best attribute (feature)**
  - Use, for example, information gain to select attribute
  - Split on $\arg\max_i IG(X_i) = \arg\max_i H(Y) - H(Y \mid X_i)$
- Recurse
- Prune



$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m).$$

Trees

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m).$$



- Trees
  - **have low bias, high variance**
  - deal with categorial variables well
  - intuitive, interpretable
  - good software exists
  - Some theoretical guarantees

Random Forests

Machine Learning – CSE546

Kevin Jamieson

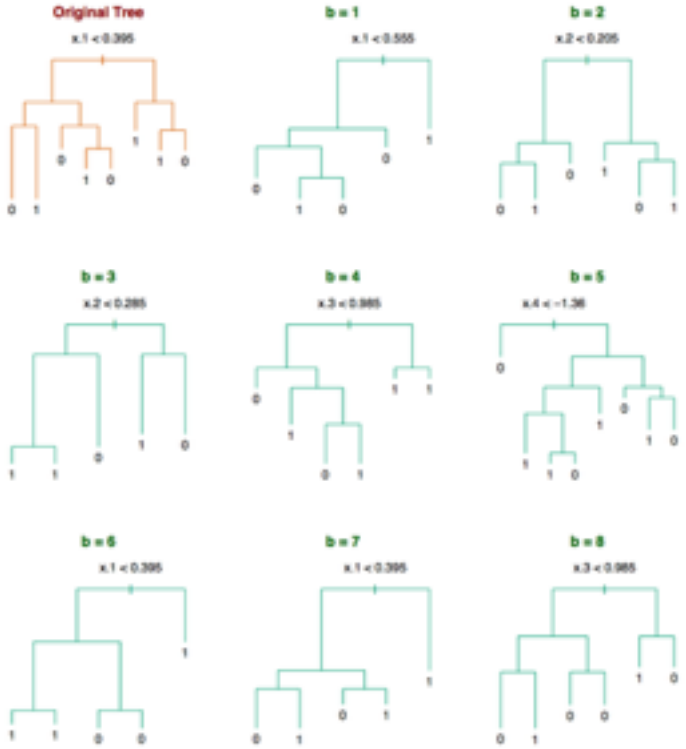
University of Washington

October 26, 2017

Random Forests

Tree methods have **low bias** but **high variance**.

One way to reduce variance is to construct a lot of “lightly correlated” trees and average them:



“Bagging:” Bootstrap aggregating

Random Forrests

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B:

   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.

   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.

      i. Select m variables at random from the p variables.

      ii. Pick the best variable/split-point among the m.

      iii. Split the node into two daughter nodes.

2. Output the ensemble of trees $\{T_b\}_1^B$.

m~sqrt(p),p/3

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

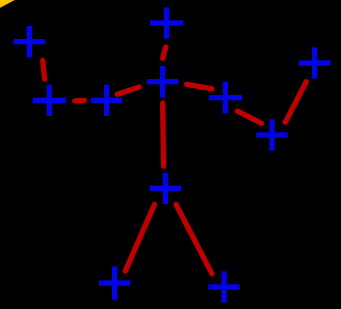
Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

The Kinect pose estimation pipeline

capture
depth image &
remove bg

infer
body parts
per pixel

cluster pixels to
hypothesize
body joint
positions

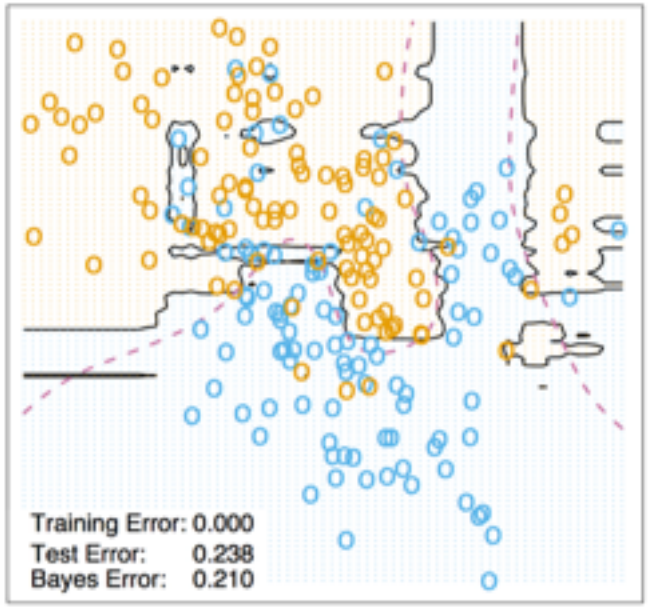


fit model &
track skeleton

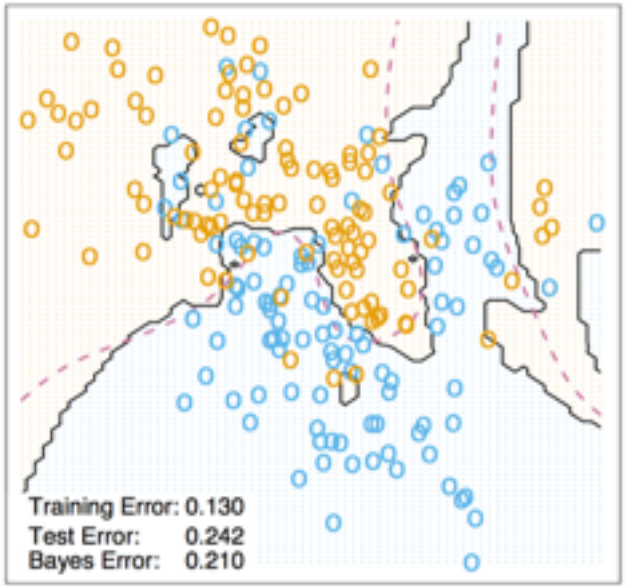
https://www.microsoft.com/en-us/
research/wp-content/uploads/2016/02/
CVPR20201120-20Final20Video.mp4

Random Forrest

Random forrest



3 nearest neighbor



Random Forrest

Given random variables $Y_1, Y_2, \ldots, Y_B$ with
$\mathbb{E}[Y_i] = y$, $\mathbb{E}[(Y_i - y)^2] = \sigma^2$, $\mathbb{E}[(Y_i - y)(Y_j - y)] = \rho\sigma^2$

The Yi's are identically distributed but **not** independent

$$\mathbb{E}[(\frac{1}{B}\sum_{i=1}^{B} Y_i - y)^2] =$$

Random Forests

- Random Forests
  - **have low bias, low variance**
  - deal with categorial variables well
  - not that intuitive or interpretable
  - good software exists
  - Some theoretical guarantees
  - Can still overfit

Random Forests

- Random Forests
  - **have low bias, low variance**
  - deal with categorial variables well
  - not that intuitive or interpretable
  - good software exists
  - Some theoretical guarantees
  - Can still overfit