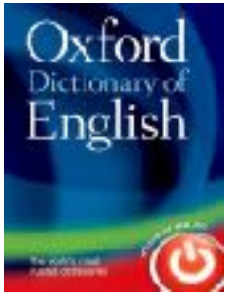
Multiclass Classification

Multi-class classification

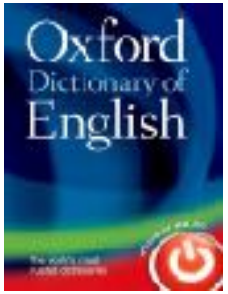
- So far: binary $y \in \{-1, +1\}$
- In general: $y \in \{c_1, c_2, \ldots, c_k\}$
- c_j's are called classes or labels



- A k-class classifier predicts y given x

How to handle categorical labels

- So far: binary $y \in \{-1, +1\}$
- In general: $y \in \{c_1, c_2, \ldots, c_k\}$
- c_j's are called classes or labels



- A k-class classifier predicts y given x

Ideas for how to use logistic regression for multi class prediction:

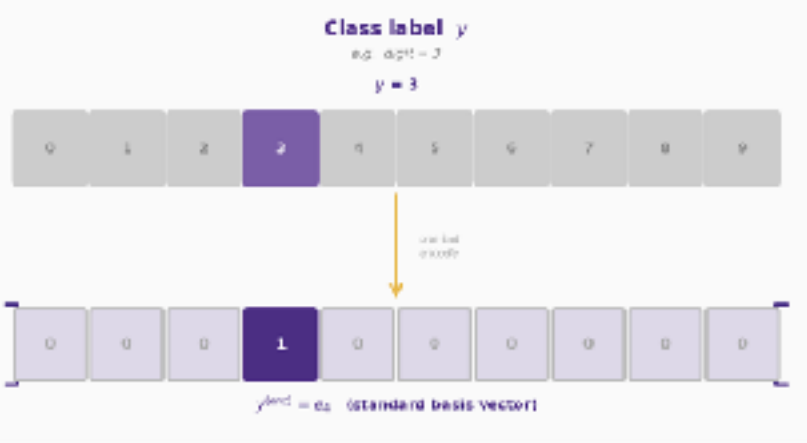
- Try to predict $\mathbb{P}[c_i \mid x]$ versus $\mathbb{P}[\text{not } c_i \mid x]$ for each i
- Try to predict $\mathbb{P}[c_i \mid x]$ as a linear function $w_i \cdot x$ for each i

One-HOT encoding labels + softmax

- So far: binary $y \in \{-1, +1\}$
- In general: $y \in \{c_1, c_2, \ldots, c_k\}$
- c_j's are called classes or labels

Ideas for how to use logistic regression for multi class prediction:

- Try to predict $\mathbb{P}[c_i \,|\, x]$ versus $\mathbb{P}[\text{not } c_i \,|\, x]$ for each i
- Try to predict $\mathbb{P}[c_i \,|\, x]$ as a linear function $w_i \cdot x$ for each i



$$\begin{array}{lll} w_1 \cdot x & \mathbb{P}[c_1 \,|\, x] & e^{w_1 \cdot x}/N \\ w_2 \cdot x & \mathbb{P}[c_2 \,|\, x] & e^{w_2 \cdot x}/N \\ \vdots & \vdots & \vdots \\ w_k \cdot x & \mathbb{P}[c_k \,|\, x] & e^{w_k \cdot x}/N \end{array}$$

Softmax classification

without loss of generality, set $k = 2,\ w_1 = 0$

2 classes

k classes

Sigmoid

Conditional probabilities

Softmax

$$P(y = 1 \mid x) = \frac{1}{1 + e^{-w^\top x}} = \frac{e^{w^\top x}}{1 + e^{w^\top x}}$$

$$P(y = -1 \mid x) = \frac{1}{1 + e^{w^\top x}}$$

$$P(y = c_j \mid x) = \frac{e^{w_j^\top x}}{\sum_{j'} e^{w_{j'}^\top x}}$$

MLE

$$\text{argmax}_w \sum_{i=1}^{n} \log\left(\frac{1}{1 + \mathrm{e}^{-y_i w^\top x_i}}\right)$$

Logistic loss

$$\text{argmax}_w \sum_{i=1}^{n} \sum_{j=1}^{k} 1\{y_i = c_j\} \log\left(\frac{\mathrm{e}^{w_j^\top x_i}}{\sum_{j'=1}^{k} e^{w_{j'}^\top x_i}}\right)$$

Binary cross entropy

Regression and classification

- ML paradigm: define prediction $f_w(x)$ and loss $\ell\left(f_w(x),\ y\right)$
- Then optimize:

$$\hat{w} = \text{argmin}_w \sum_{i=1}^{n} \ell\left(f_w(x_i),\ y_i\right)$$

- Squared error loss: $\ell\left(f_w(x),\ y\right) = \left(y - f_w(x)\right)^2$
- Logistic loss: $\ell\left(f_w(x),\ y\right) = \log(1 + \exp\left(-y f_w(x)\right))$

The ML pipeline

1. Task
2. Data
3. Model family
4. Training loss
4. Optimize
5. Pick hyperparameters
6. Evaluate

Bootstrap

Limitations of CV

- An 80/20 split throws out a relatively large amount of data if only have, say, 20 examples.
- Test error is informative, but how accurate is this number? (e.g., 3/5 heads vs. 30/50)
- How do I get confidence intervals on statistics like the median or variance of a distribution?
- Instead of the error for the entire dataset, what if I want to study the error for a *particular example* x?

The Bootstrap: Developed by Efron in 1979.

Bootstrap: basic idea

Given dataset drawn iid samples with CDF F_Z:

$$\mathcal{D} = \{z_1, \ldots, z_n\} \overset{i.i.d.}{\sim} F_Z$$

We compute a *statistic* of the data to get: $\widehat{\theta} = t(\mathcal{D})$

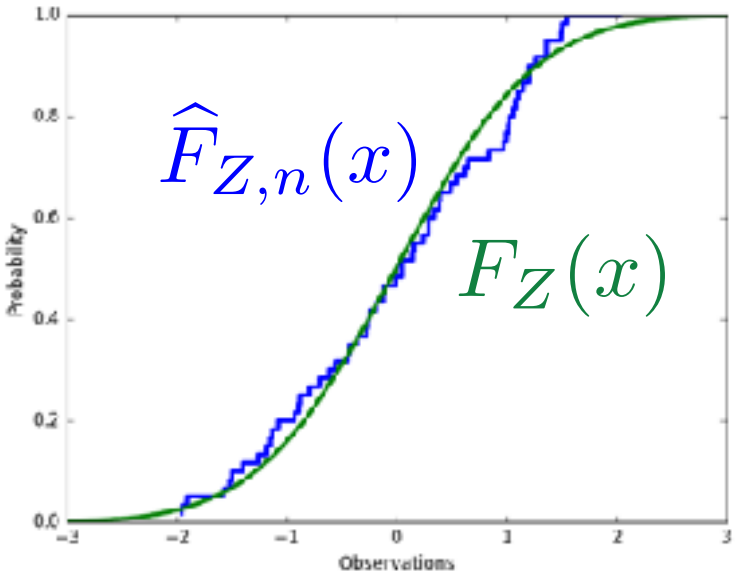
What is the distribution of $\widehat{\theta} = t(\mathcal{D})$

Bootstrap: basic idea

Given dataset drawn iid samples with CDF F_Z:

$$\mathcal{D} = \{z_1, \ldots, z_n\} \overset{i.i.d.}{\sim} F_Z$$

We compute a *statistic* of the data to get: $\widehat{\theta} = t(\mathcal{D})$



$$F_Z(x) = \mathbb{P}(Z \leq x)$$

$$\widehat{F}_{Z,n}(x) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\{z_i \leq x\}$$

$$|\widehat{F}_{Z,n}(x) - F_Z(x)| \overset{n \to \infty}{\to} 0 \quad \text{a.s.}$$

Bootstrap: basic idea

Given dataset drawn iid samples with CDF F_Z:

$$\mathcal{D} = \{z_1, \ldots, z_n\} \overset{i.i.d.}{\sim} F_Z$$

We compute a *statistic* of the data to get: $\widehat{\theta} = t(\mathcal{D})$

For b=1,…,B define the *b*th ***bootstrapped*** dataset as drawing *n* samples **with replacement** from *D*

$$\mathcal{D}^{*b} = \{z_1^{*b}, \ldots, z_n^{*b}\} \overset{i.i.d.}{\sim} \widehat{F}_{Z,n}$$

and the *b*th bootstrapped statistic as: $\theta^{*b} = t(\mathcal{D}^{*b})$

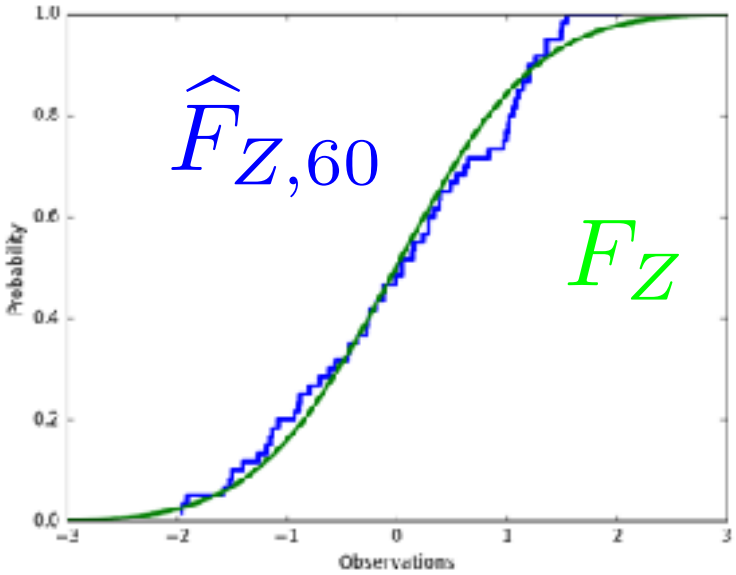
Bootstrap: basic idea

Given dataset drawn iid samples with CDF F_Z:

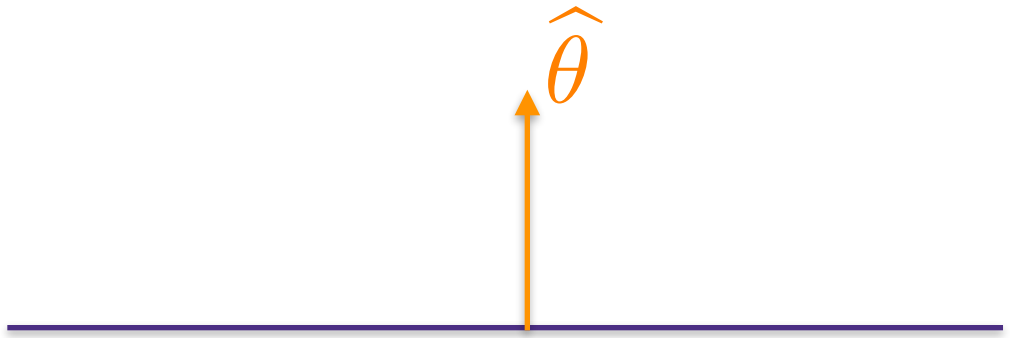
$$\mathcal{D} = \{z_1, \ldots, z_n\} \overset{i.i.d.}{\sim} F_Z$$

We compute a *statistic* of the data to get: $\widehat{\theta} = t(\mathcal{D})$

$$\mathcal{D}^{*b} = \{z_1^{*b}, \ldots, z_n^{*b}\} \overset{i.i.d.}{\sim} \widehat{F}_{Z,n} \quad \theta^{*b} = t(\mathcal{D}^{*b})$$



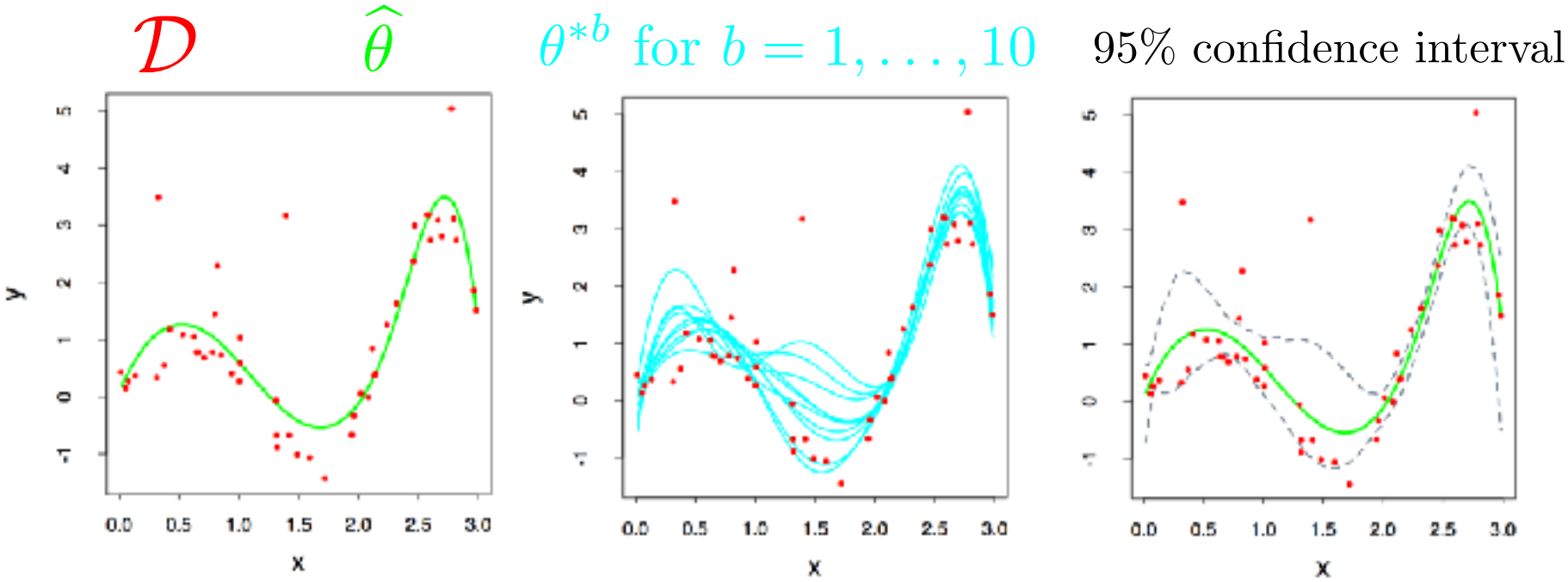
$$|\widehat{F}_{Z,n}(x) - F_Z(x)| \overset{n\to\infty}{\to} 0 \quad \text{a.s.}$$



Applications

Common applications of the bootstrap:

- **Estimate parameters that escape simple analysis like the variance or median of an estimate**
- **Confidence intervals**
- **Estimates of error for a particular example:**



Figures from Hastie et al

Takeaways

Advantages:

- **Bootstrap is very generally applicable. Build a confidence interval around *anything***
- **Very simple to use**
- **Appears to give meaningful results even when the amount of data is very small**
- **Very strong asymptotic theory (as num. examples goes to infinity)**

Takeaways

Advantages:

- **Bootstrap is very generally applicable. Build a confidence interval around *anything***
- **Very simple to use**
- **Appears to give meaningful results even when the amount of data is very small**
- **Very strong asymptotic theory (as num. examples goes to infinity)**

Disadvantages

- **Very few meaningful finite-sample guarantees**
- **Potentially computationally intensive**
- **Reliability relies on test statistic and rate of convergence of empirical CDF to true CDF, which is unknown**
- **Poor performance on "extreme statistics" (e.g., the max)**

Not perfect, but better than nothing.

SVMs and Kernels

UNIVERSITY of WASHINGTON

Two different approaches to regression/classification

- Assume something about P(x,y)
- Find f which maximizes likelihood of training data | assumption
  - Often reformulated as minimizing loss

Versus

- Pick a loss function
- Pick a set of hypotheses H
- Pick f from H which minimizes loss on training data

Our description of logistic regression was the former

- **Learn**: f:**X** —>Y
  - **X** – features
  - **Y – target classes**

$$Y \in \{-1, 1\}$$

- **Expected loss of f:**

$$\mathbb{E}_{XY}[\mathbf{1}\{f(X) \neq Y\}] = \mathbb{E}_X[\mathbb{E}_{Y|X}[\mathbf{1}\{f(x) \neq Y\}|X = x]]$$

$$\mathbb{E}_{Y|X}[\mathbf{1}\{f(x) \neq Y\}|X = x] = 1 - P(Y = f(x)|X = x)$$

- **Bayes optimal classifier:**

$$f(x) = \arg\max_y \mathbb{P}(Y = y|X = x)$$

- **Model of logistic regression:**

$$P(Y = y|x, w) = \frac{1}{1 + \exp(-y\, w^T x)}$$

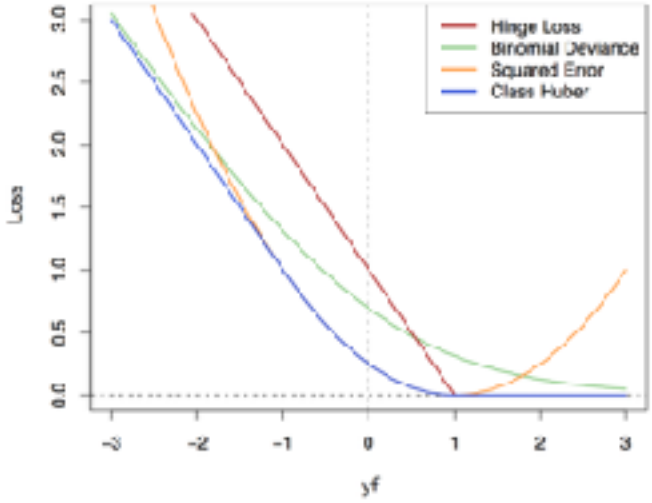
- **Loss function:**

$$\ell(f(x), y) = \mathbf{1}\{f(x) \neq y\}$$

What if the model is wrong? What other ways can we pick linear decision rules?

Loss Functions

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



- Loss functions: $\sum_{i=1}^{n} \ell_i(w)$

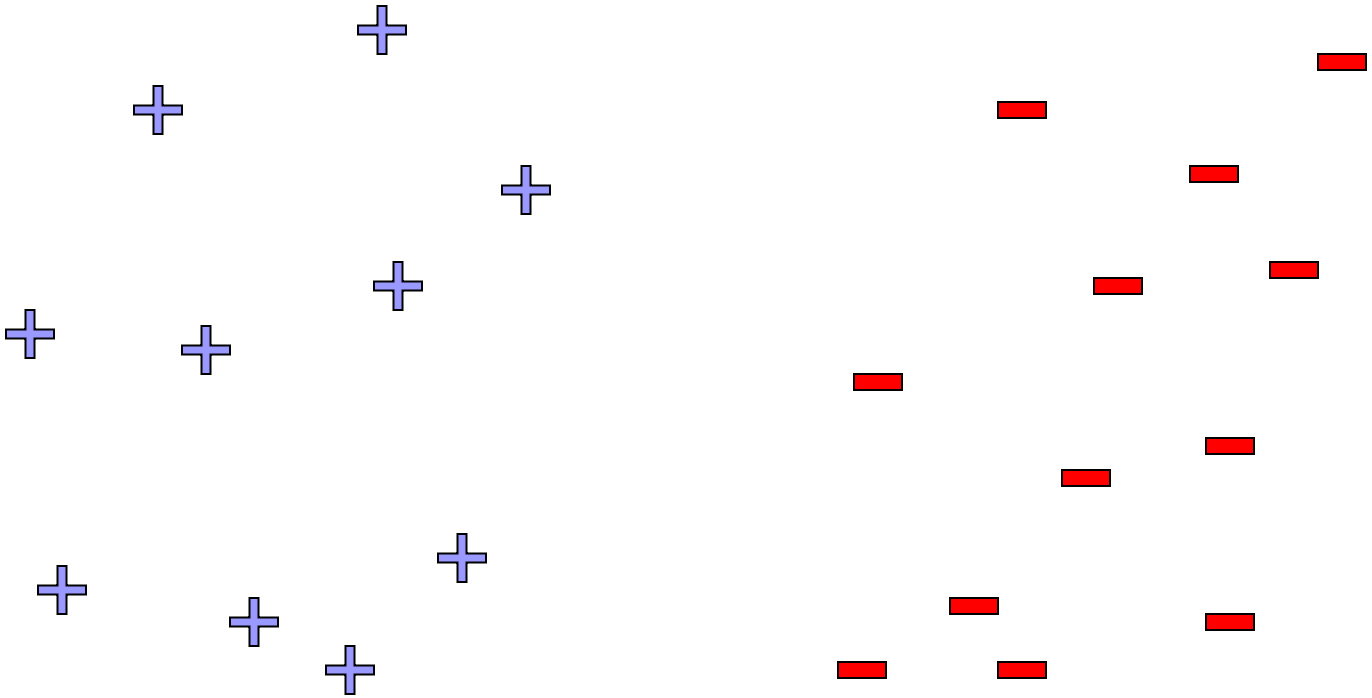
Squared error Loss: $\ell_i(w) = (y_i - x_i^T w)^2$

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

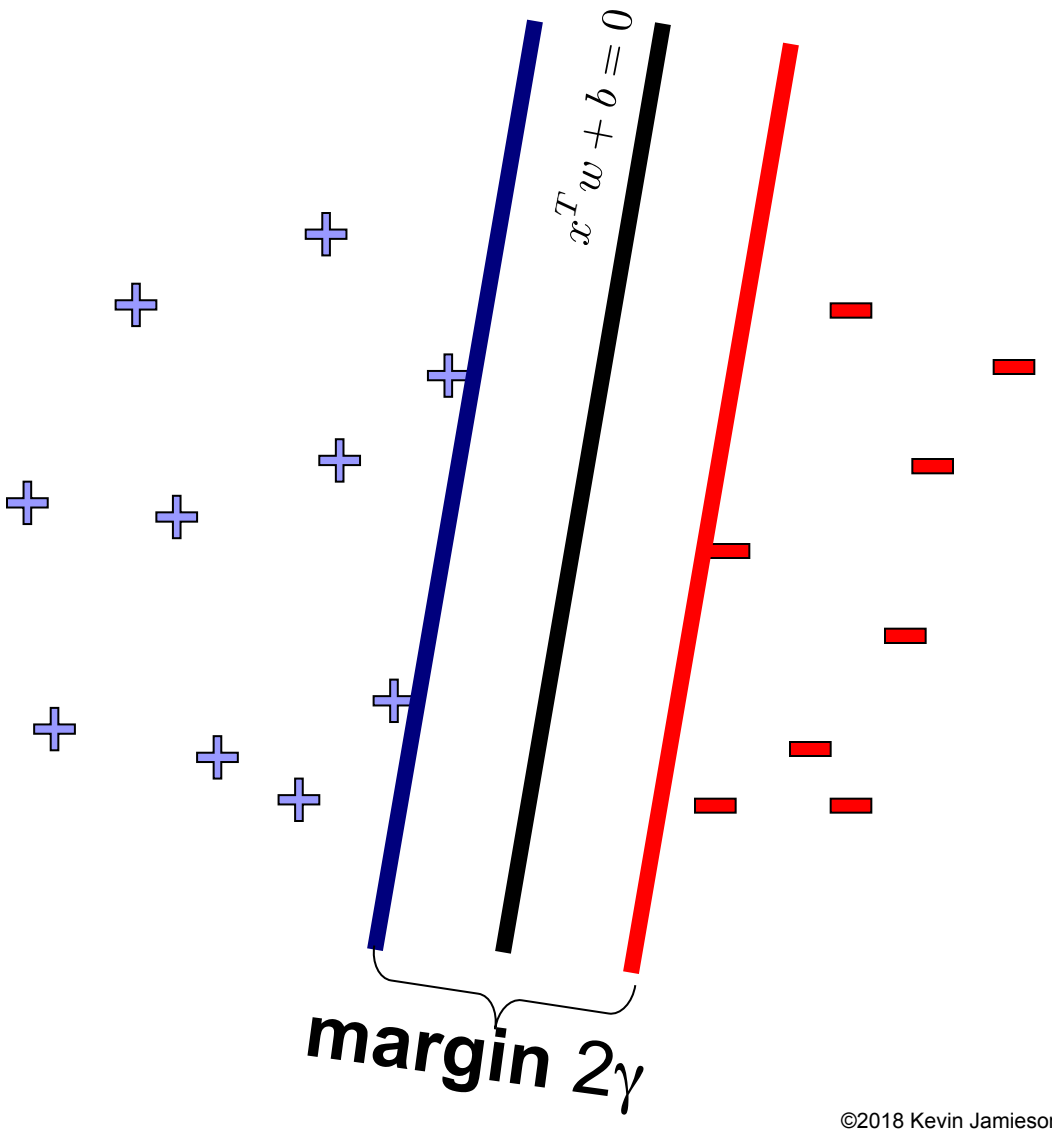
0/1 loss: $\ell_i(w) = \mathbb{I}[\text{sign}(y_i) \neq \text{sign}(x_i^T w)]$

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

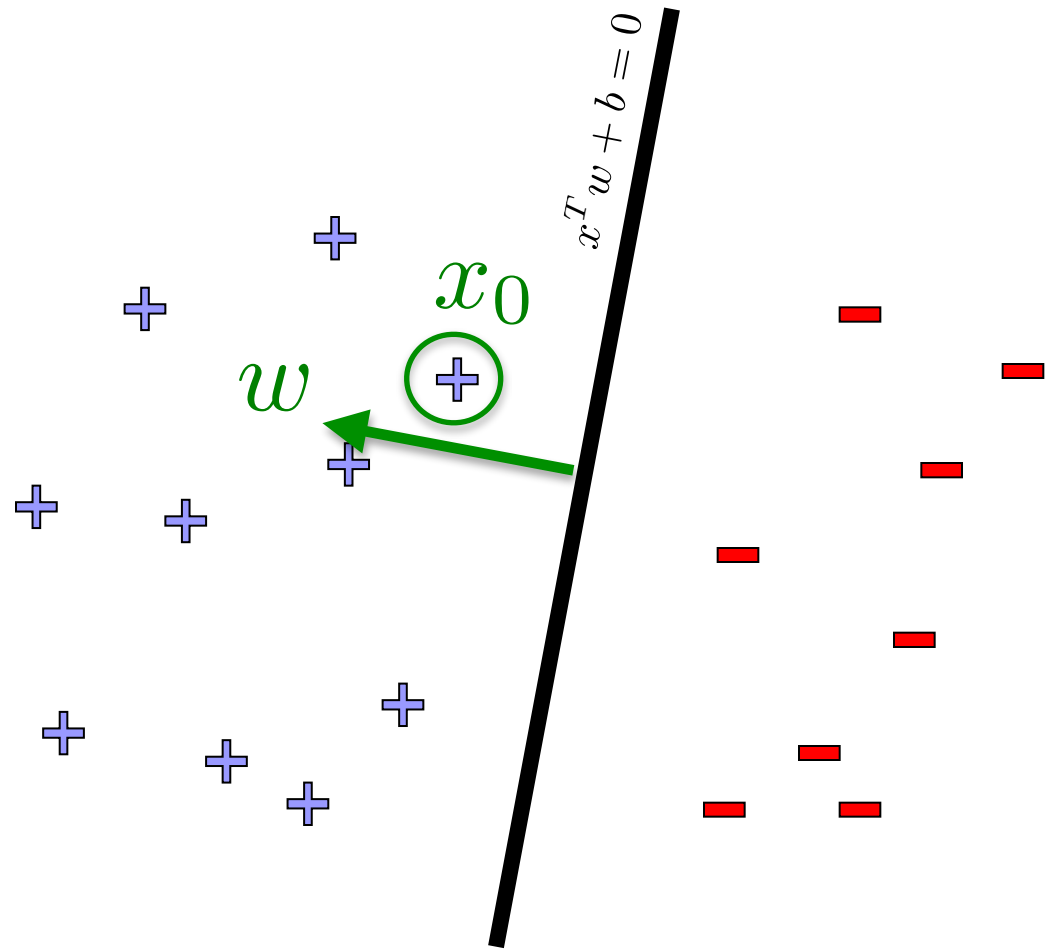
Linear classifiers – Which line is better?



Pick the one with the largest margin!

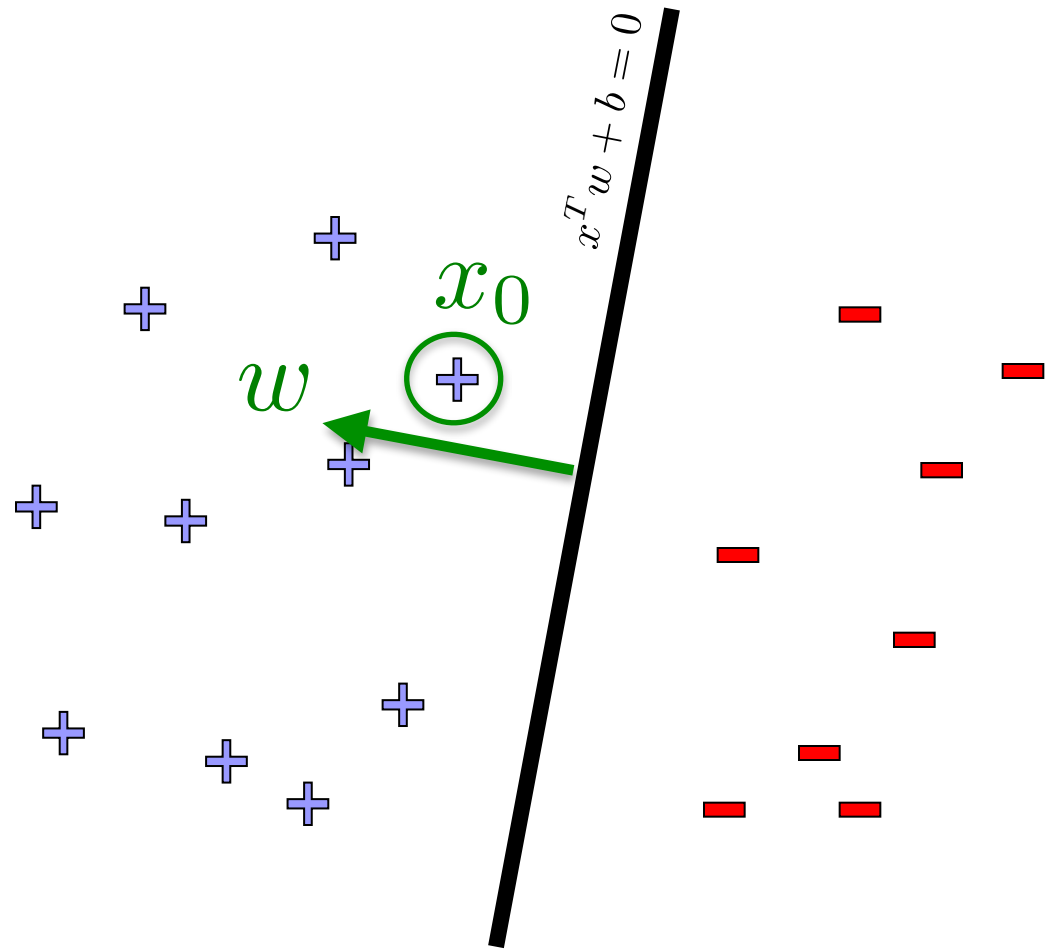


Pick the one with the largest margin!



Distance from x_0 to hyperplane defined by $x^T w + b = 0$?

Pick the one with the largest margin!



Distance from x_0 to hyperplane defined by $x^T w + b = 0$?

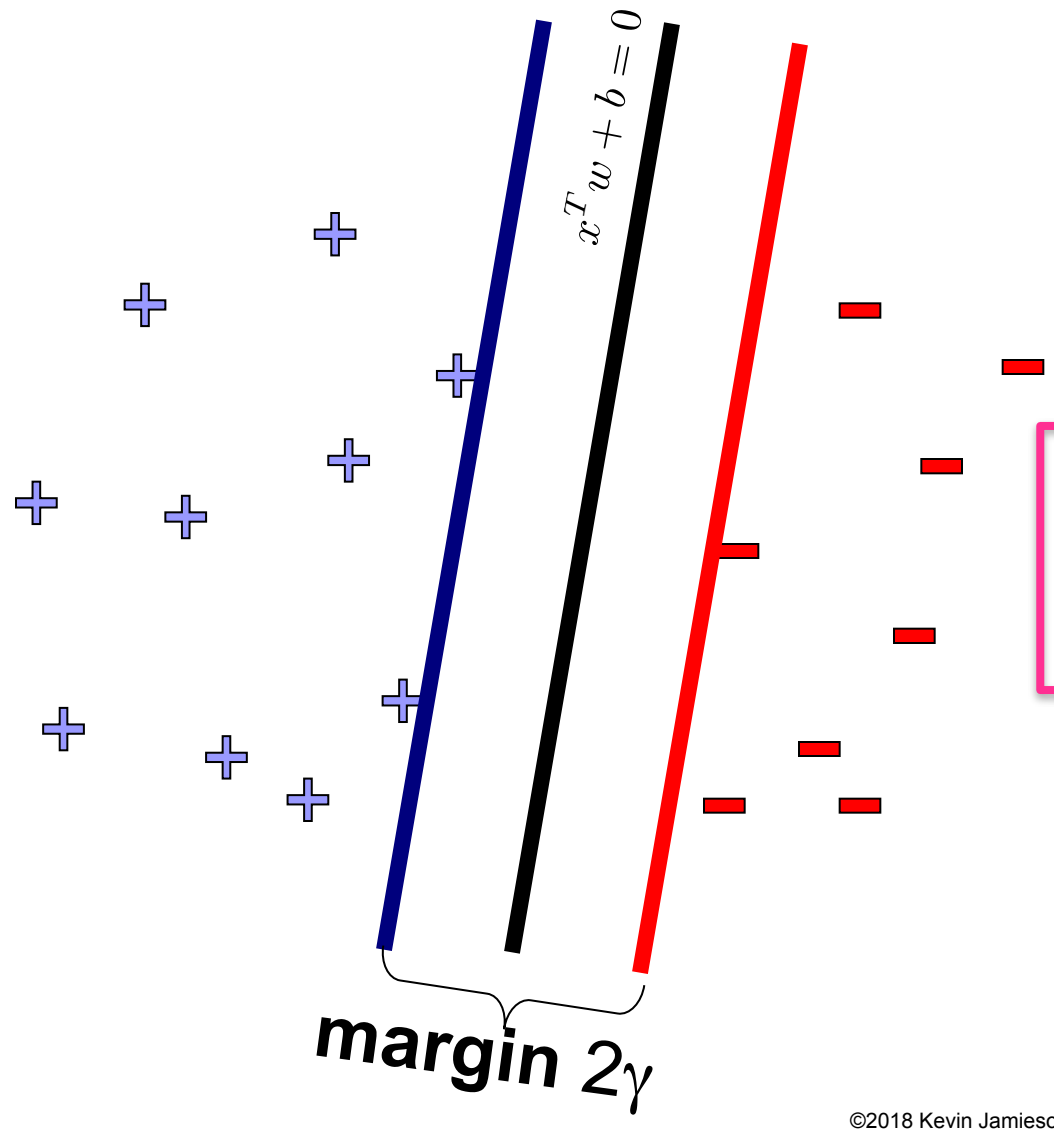
If $\widetilde{x}_0$ is the projection of x_0 onto the hyperplane then

$$||x_0 - \widetilde{x}_0||_2 = |(x_0^T - \widetilde{x}_0)^T \frac{w}{||w||_2}|$$

$$= \frac{1}{||w||_2} |x_0^T w - \widetilde{x}_0^T w|$$

$$= \frac{1}{||w||_2} |x_0^T w + b|$$

Pick the one with the largest margin!



Distance of x_0 from hyperplane $x^T w + b$:

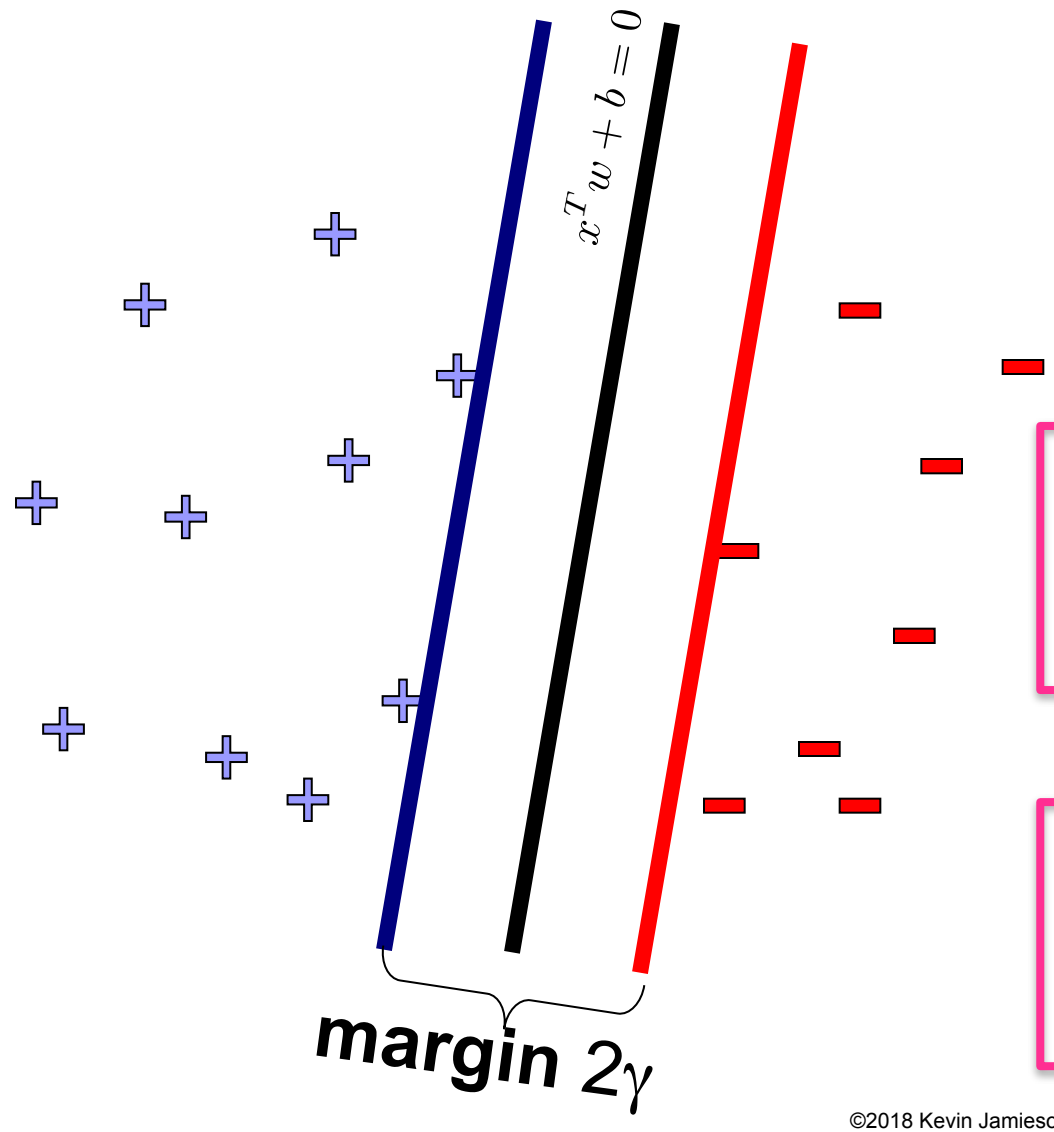
$$\frac{1}{||w||_2}(x_0^T w + b)$$

Optimal Hyperplane

$$\max_{w,b} \gamma$$

$$\text{subject to } \frac{1}{||w||_2} y_i(x_i^T w + b) \geq \gamma \quad \forall i$$

Pick the one with the largest margin!



Distance of x_0 from hyperplane $x^T w + b$:

$$\frac{1}{||w||_2}(x_0^T w + b)$$

Optimal Hyperplane

$$\max_{w,b} \gamma$$

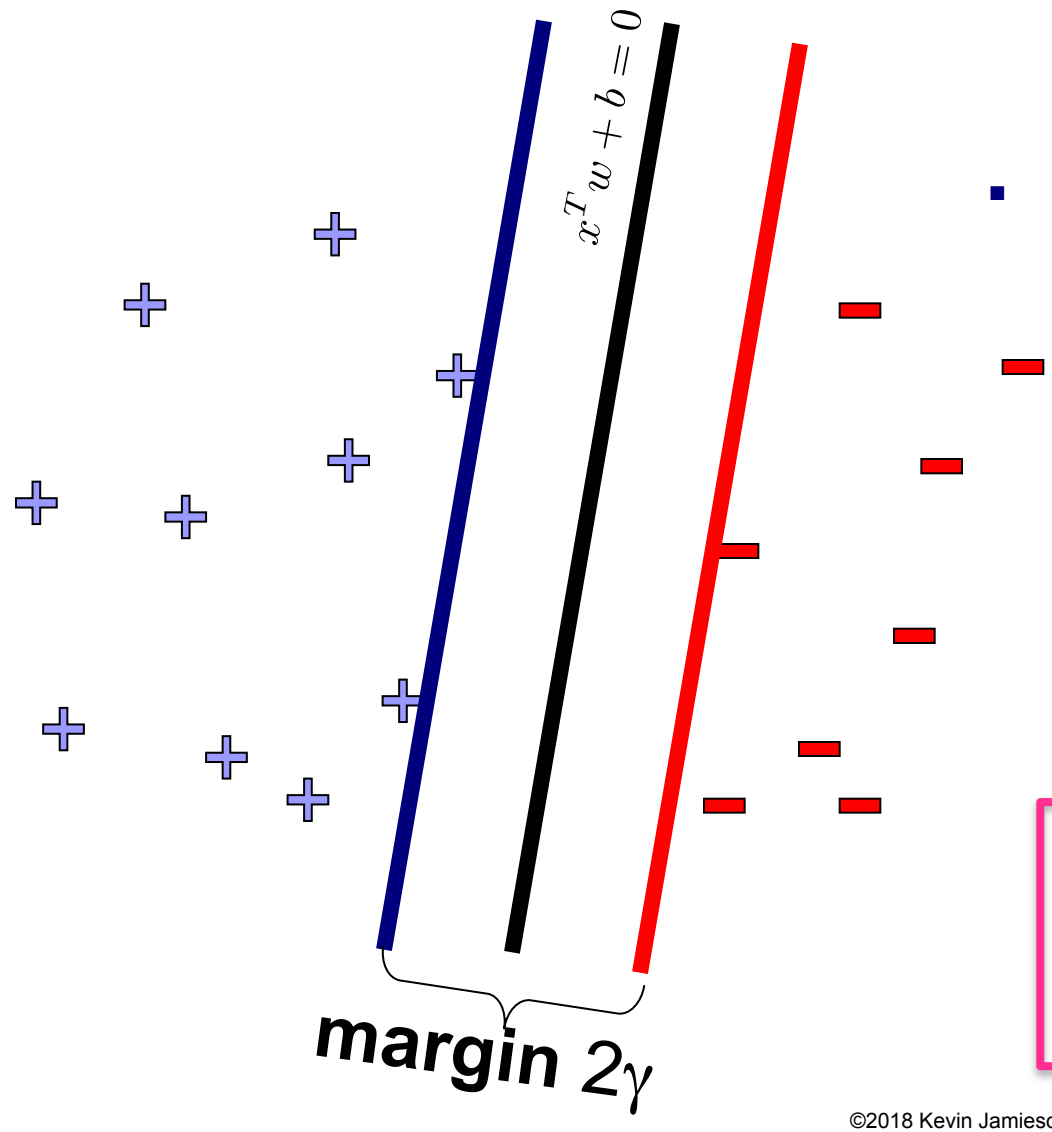
$$\text{subject to } \frac{1}{||w||_2} y_i(x_i^T w + b) \geq \gamma \quad \forall i$$

Optimal Hyperplane (reparameterized)

$$\min_{w,b} ||w||_2^2$$

$$\text{subject to } y_i(x_i^T w + b) \geq 1 \quad \forall i$$

Pick the one with the largest margin!



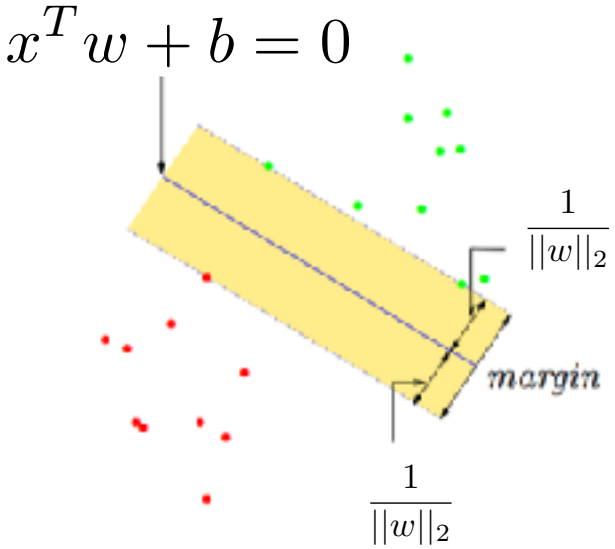
- Solve efficiently by many methods, e.g.,
  - quadratic programming (QP)
    - Well-studied solution algorithms
  - Stochastic gradient descent
  - Coordinate descent (in the dual)

Optimal Hyperplane (reparameterized)

$$\min_{w,b} ||w||_2^2$$

$$\text{subject to } y_i(x_i^T w + b) \geq 1 \quad \forall i$$

What if the data is not linearly separable?



If data is linearly separable

$$\min_{w,b} ||w||_2^2$$

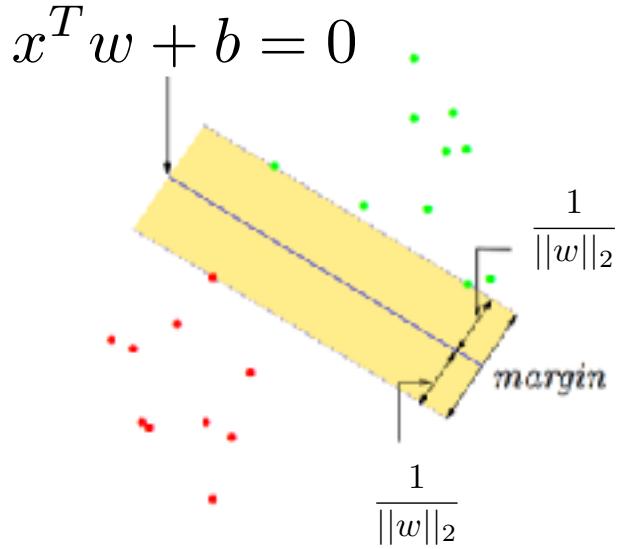
$$y_i(x_i^T w + b) \geq 1 \quad \forall i$$

If data is not linearly separable, some points don't satisfy margin constraint:

Two options:

1. Introduce slack to this optimization problem
2. Lift to higher dimensional space

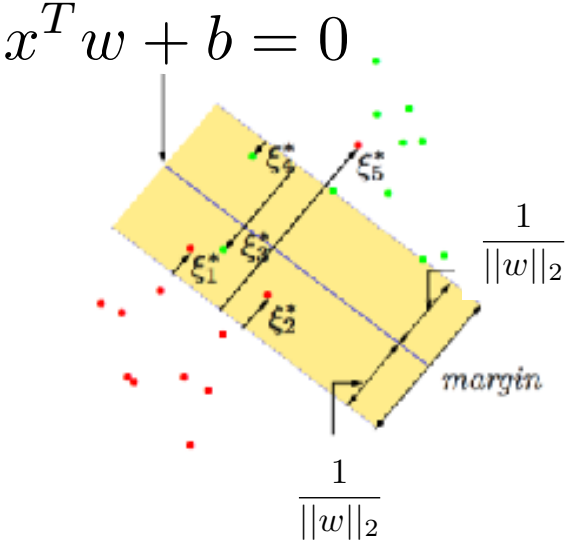
What if the data is not linearly separable?



If data is linearly separable:

$$\min_{w,b} ||w||_2^2$$

$$y_i(x_i^T w + b) \geq 1 \quad \forall i$$



If data is not linearly separable,

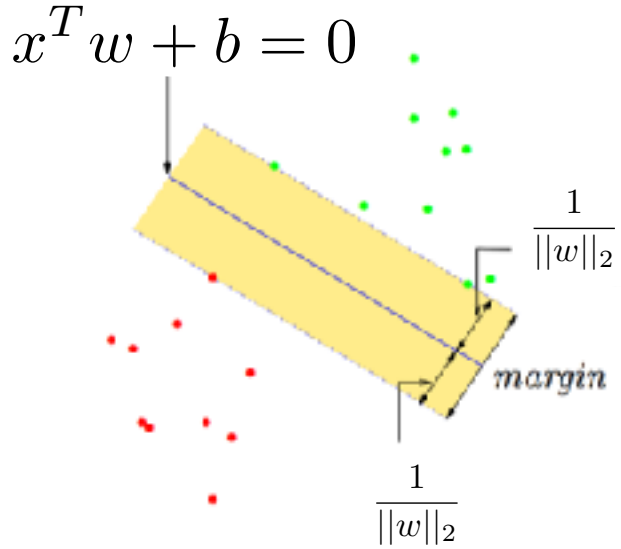
some points don't satisfy margin constraint:

$$\min_{w,b} ||w||_2^2$$

$$y_i(x_i^T w + b) \geq 1 - \xi_i \quad \forall i$$

$$\xi_i \geq 0, \sum_{j=1}^{n} \xi_j \leq \nu$$

What if the data is not linearly separable?



If data is linearly separable:

$$\min_{w,b} ||w||_2^2$$

$$y_i(x_i^T w + b) \geq 1 \quad \forall i$$

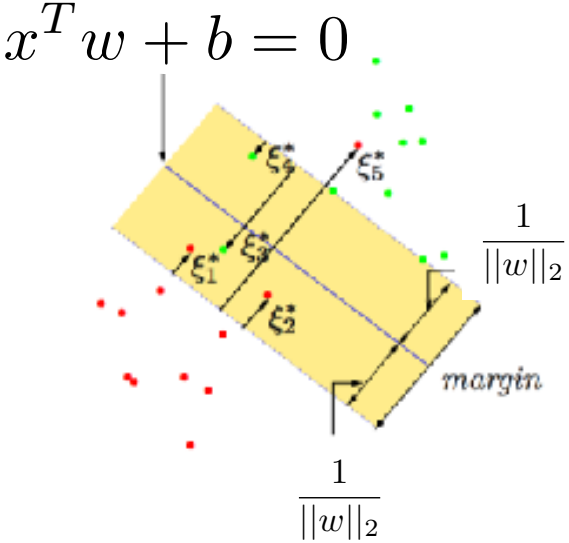
If data is not linearly separable,

some points don't satisfy margin constraint:

$$\min_{w,b} ||w||_2^2$$

$$y_i(x_i^T w + b) \geq 1 - \xi_i \quad \forall i$$

$$\xi_i \geq 0, \sum_{j=1}^{n} \xi_j \leq \nu$$



- What are "support vectors?"

SVM as penalization method

- Original quadratic program with linear constraints:

$$\min_{w,b} ||w||_2^2$$

$$y_i(x_i^T w + b) \geq 1 - \xi_i \quad \forall i$$

$$\xi_i \geq 0, \sum_{j=1}^{n} \xi_j \leq \nu$$

SVM as penalization method

- Original quadratic program with linear constraints:

$$\min_{w,b} ||w||_2^2$$

$$y_i(x_i^T w + b) \geq 1 - \xi_i \quad \forall i$$

$$\xi_i \geq 0, \sum_{j=1}^{n} \xi_j \leq \nu$$

- Using same constrained convex optimization trick as for lasso:

For any $\nu \geq 0$ there exists a $\lambda \geq 0$ such that the solution the following solution is equivalent:

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

SVMs: optimizing what?

SVM objective:

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

$$\nabla_w \ell_i(w, b) = \begin{cases} -x_i y_i + \frac{2\lambda}{n} w & \text{if } y_i(b + x_i^T w) < 1 \\ \frac{2\lambda}{n} & \text{otherwise} \end{cases}$$

$$\nabla_b \ell_i(w, b) = \begin{cases} -y_i & \text{if } y_i(b + x_i^T w) < 1 \\ 0 & \text{otherwise} \end{cases}$$

SVMs: optimizing what?

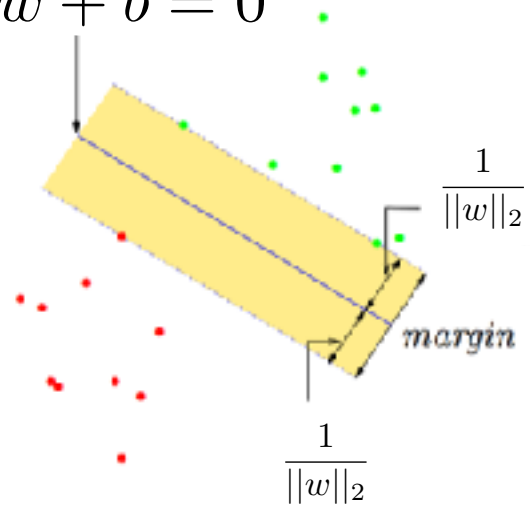
SVM objective:

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

Note: the minimizer of this can be written in terms of very few of the training points. These points are known as support vectors.

What if the data is not linearly separable?

- $x^T w + b = 0$



le, some points don't satisfy margin

$$\min_{w,b} ||w||_2^2$$

$$y_i(x_i^T w + b) \geq 1 \quad \forall i$$

Two options:

1. Introduce slack to this optimization problem
2. **Lift to higher dimensional space**

SVM as penalization method

- Original quadratic program with linear constraints:

$$\min_{w,b} ||w||_2^2$$

$$y_i(x_i^T w + b) \geq 1 - \xi_i \quad \forall i$$

$$\xi_i \geq 0, \sum_{j=1}^{n} \xi_j \leq \nu$$

- Using same constrained convex optimization trick as for lasso:

For any $\nu \geq 0$ there exists a $\lambda \geq 0$ such that the solution the following solution is equivalent:

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

SVMs: optimizing what?

SVM objective:

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

$$\nabla_w \ell_i(w, b) = \begin{cases} -x_i y_i + \frac{2\lambda}{n} w & \text{if } y_i(b + x_i^T w) < 1 \\ \frac{2\lambda}{n} & \text{otherwise} \end{cases}$$

$$\nabla_b \ell_i(w, b) = \begin{cases} -y_i & \text{if } y_i(b + x_i^T w) < 1 \\ 0 & \text{otherwise} \end{cases}$$

SVMs: optimizing what?

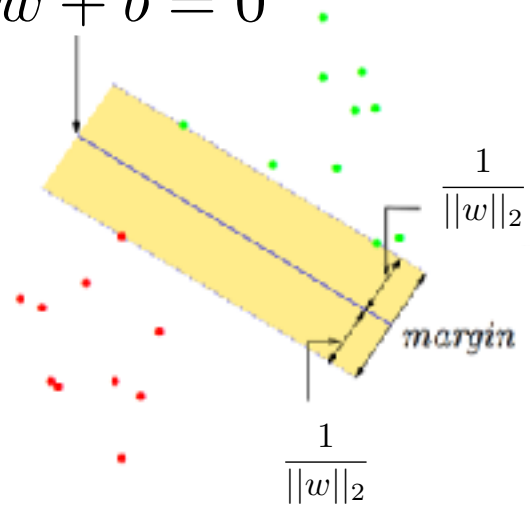
SVM objective:

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

Note: the minimizer of this can be written in terms of very few of the training points. These points are known as support vectors.

What if the data is not linearly separable?

- $x^T w + b = 0$



le, some points don't satisfy margin

$$\min_{w,b} ||w||_2^2$$

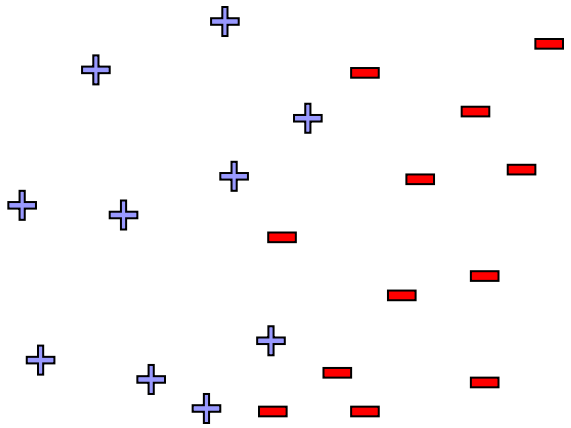
$$y_i(x_i^T w + b) \geq 1 \quad \forall i$$

Two options:

1. Introduce slack to this optimization problem
2. **Lift to higher dimensional space**

What if the data is not linearly separable?

Use 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

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

Dot-product of polynomials

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

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

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

Dot-product of polynomials

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

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

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

Feature space can get really large really quickly!

General d :

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

How do we deal with high-dimensional lifts/data?

A fundamental trick in ML: use kernels

A function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a *kernel* for a map ϕ if $K(x, x') = \phi(x) \cdot \phi(x')$ for all x, x'.

So, if we can represent our algorithms/decision rules as dot products
and we can find a kernel for our feature map
then we can avoid explicitly dealing with ϕ(x).

Examples of Kernels

- **Polynomials of degree exactly d**

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

- **Polynomials of degree up to d**

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

- **Gaussian (squared exponential) kernel**

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

- **Sigmoid**

$$K(u, v) = \tanh(\gamma \cdot u^T v + r)$$

The Kernel Trick

Pick a kernel K

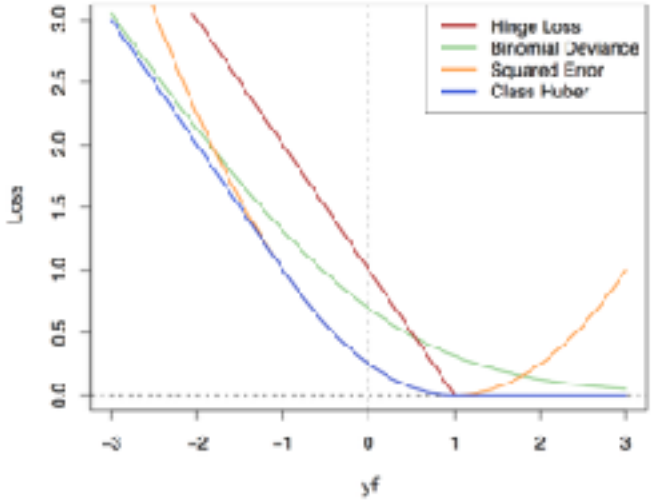
Prove $w = \sum_i \alpha_i x_i$

Change loss function/decision rule to only access data through dot products

- **Decision rule is easy: why?**

Substitute $K(x_i, x_j)$ for $x_i^T x_j$

Loss Functions



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

- Loss functions: $\sum_{i=1}^{n} \ell_i(w)$

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

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

0/1 loss: $\ell_i(w) = \mathbb{I}[\text{sign}(y_i) \neq \text{sign}(x_i^T w)]$

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

The Kernel Trick for regularized least squares

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

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

The Kernel Trick for regularized least squares

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

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

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

The Kernel Trick for regularized least squares

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

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

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

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

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?

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

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

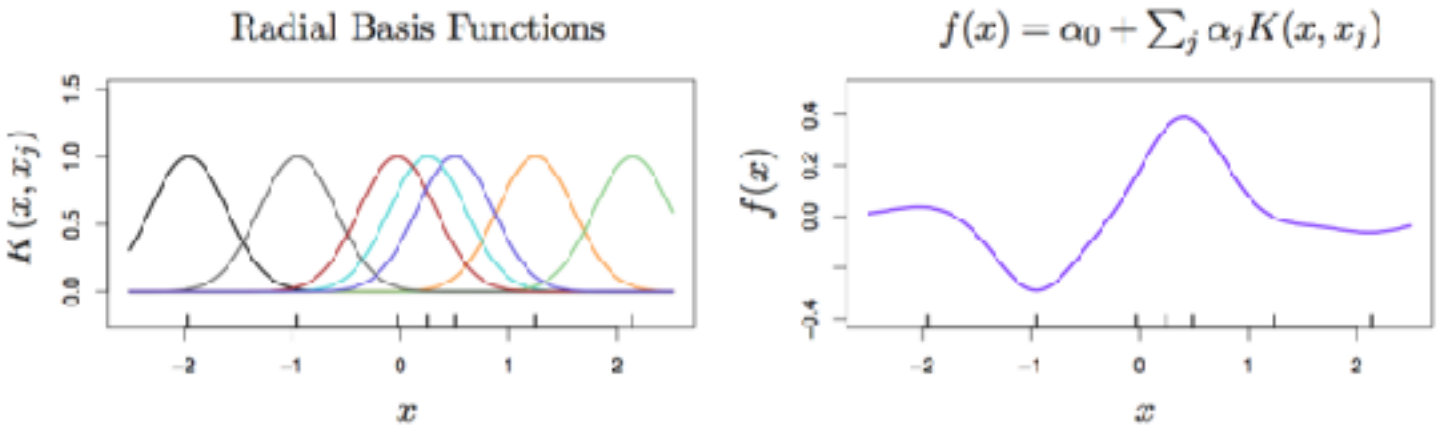
Unregularized kernel least squares can (over) fit **any data**!

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

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

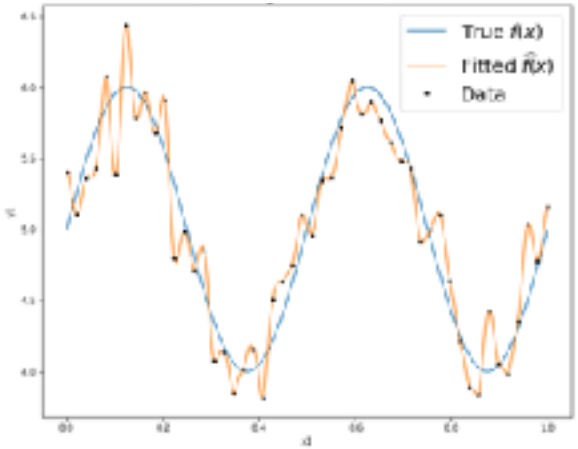


RBF Kernel

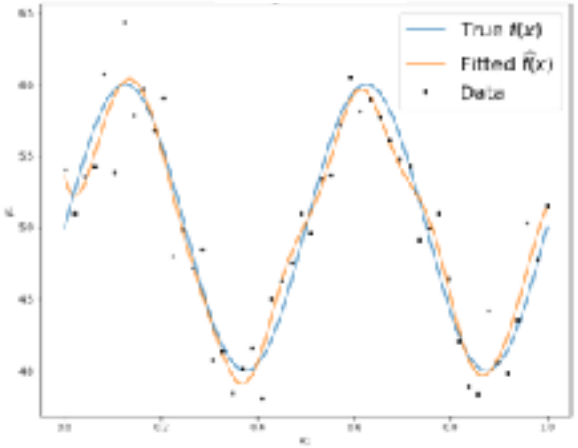
$$K(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{||\mathbf{u} - \mathbf{v}||_2^2}{2\sigma^2}\right)$$

The bandwidth sigma has an enormous effect on fit:

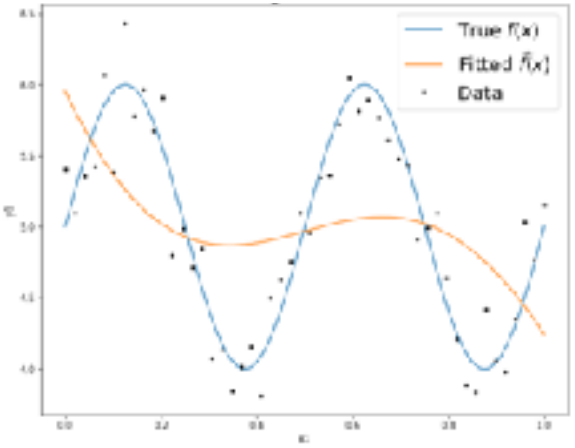
$\sigma = 10^{-2}$ $\lambda = 10^{-4}$



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



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



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

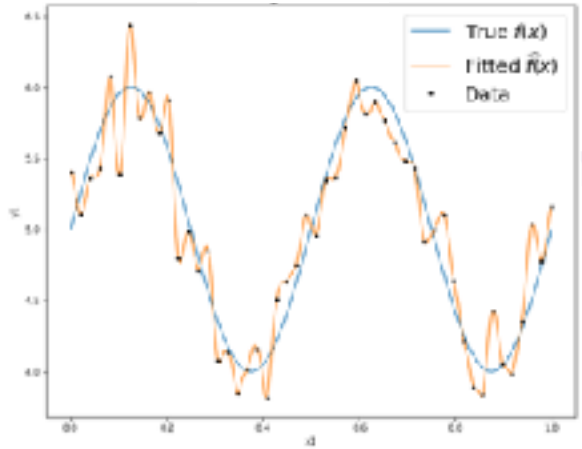
$$\widehat{f}(x) = \sum_{i=1}^{n} \widehat{\alpha}_i K(x_i, x)$$

RBF Kernel

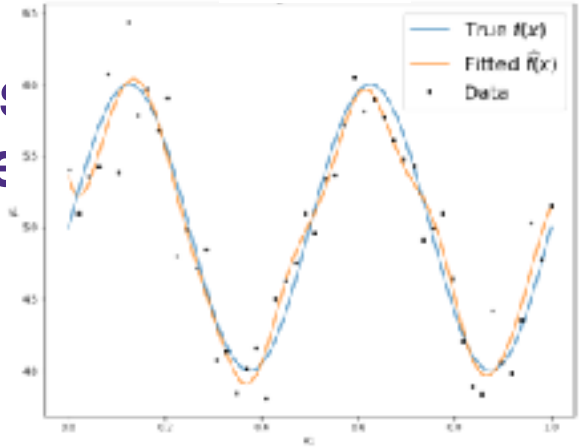
$$K(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{||\mathbf{u} - \mathbf{v}||_2^2}{2\sigma^2}\right)$$

The bandwidth sigma has an enormous effect on fit:

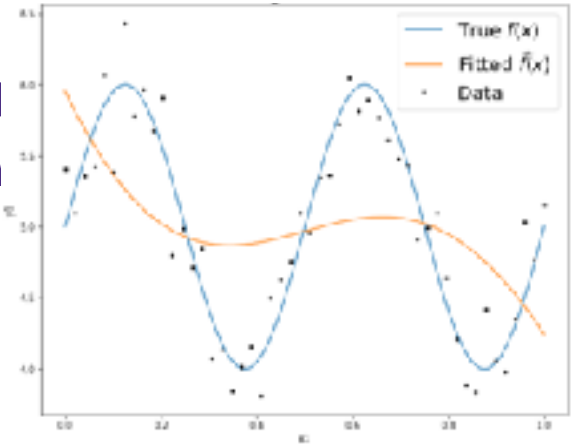
$\sigma = 10^{-2}$ $\lambda = 10^{-4}$



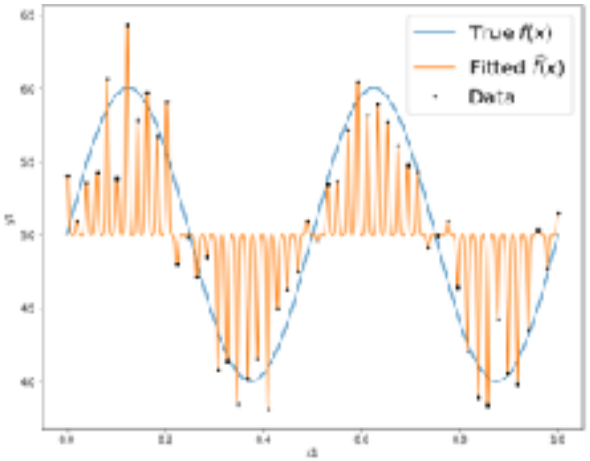
$\sigma = 10^{-1}$ $\lambda = 10^{-4}$



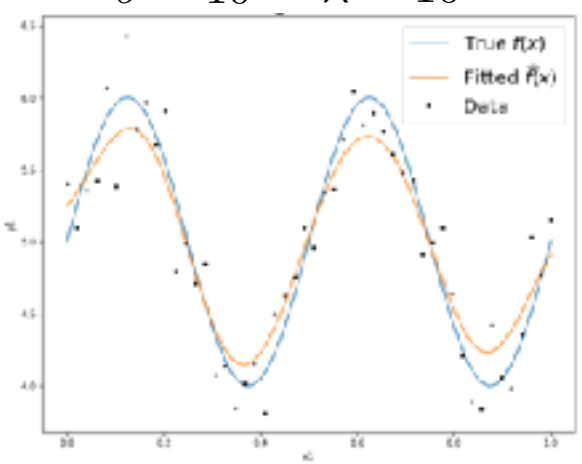
$\sigma = 10^{-0}$ $\lambda = 10^{-4}$



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



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



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

RBF Kernel

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

Basis representation in 1d?

$$[\phi(x)]_i = \frac{1}{\sqrt{i!}} e^{-\frac{x^2}{2}} x^i \quad \text{for } i = 0, 1, \ldots$$

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

$$\begin{aligned}
\phi(x)^T \phi(x') &= \sum_{i=0}^{\infty} \left(\frac{1}{\sqrt{i!}} e^{-\frac{x^2}{2}} x^i\right)\left(\frac{1}{\sqrt{i!}} e^{-\frac{(x')^2}{2}} (x')^i\right) \\
&= e^{-\frac{x^2+(x')^2}{2}} \sum_{i=0}^{\infty} \frac{1}{i!}(xx')^i \\
&= e^{-|x-x'|^2/2}
\end{aligned}$$

If n is very large, allocating an n-by-n matrix is tough. Can we truncate the above sum to approximate the kernel?

RBF kernel and random features

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

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

Recall HW1 where we used the feature map:

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

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

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

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

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

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

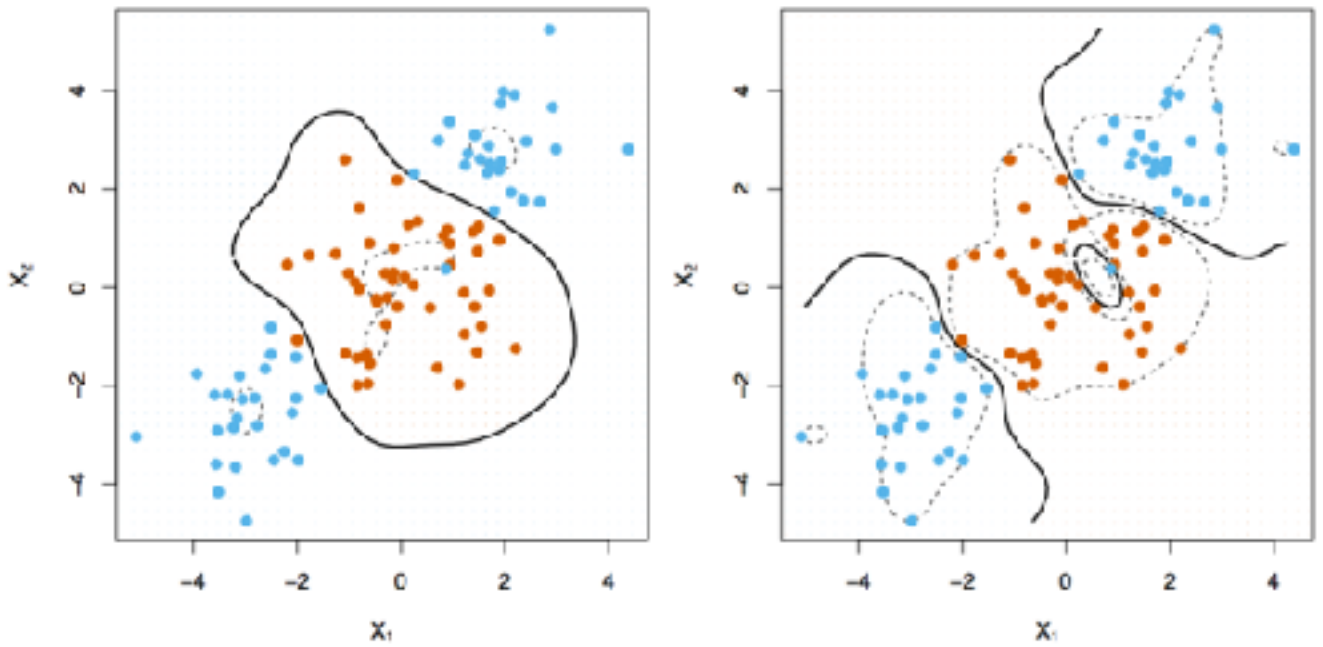
- **Regularization! Fat shattering (R/margin)^2**

- **What about sparsity?**

RBF Classification

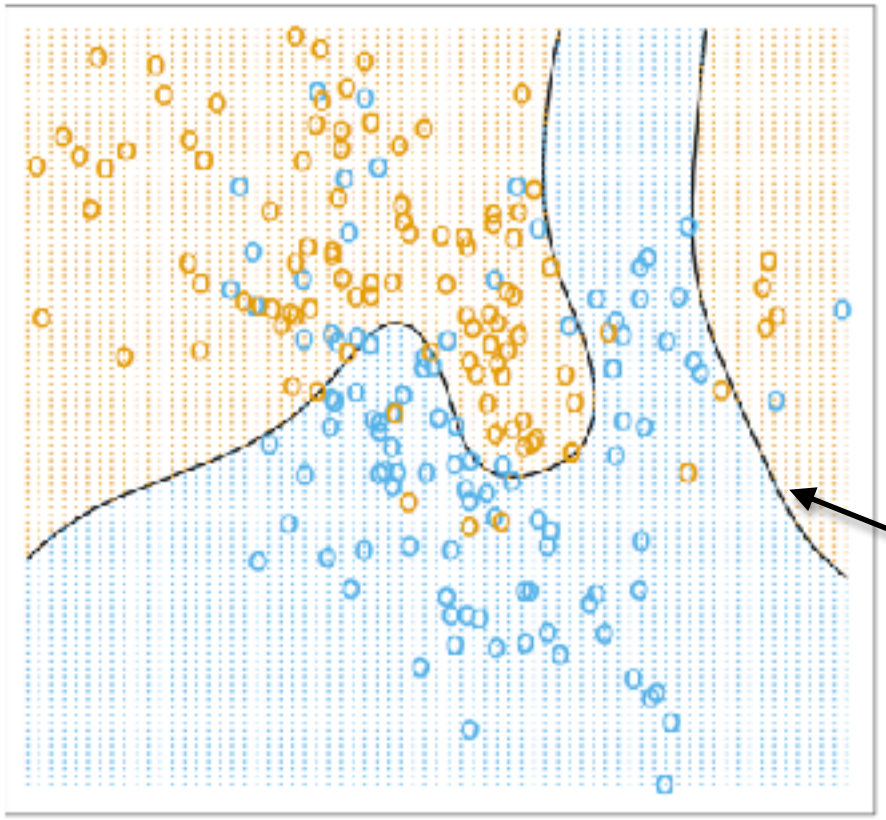
$$\widehat{w} = \sum_{i=1}^{n} \max\{0, 1 - y_i(b + x_i^T w)\} + \lambda ||w||_2^2$$

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



Nearest Neighbor

Some data, Bayes Classifier



Training data:

True label: +1

True label: -1

Optimal "Bayes" classifier:

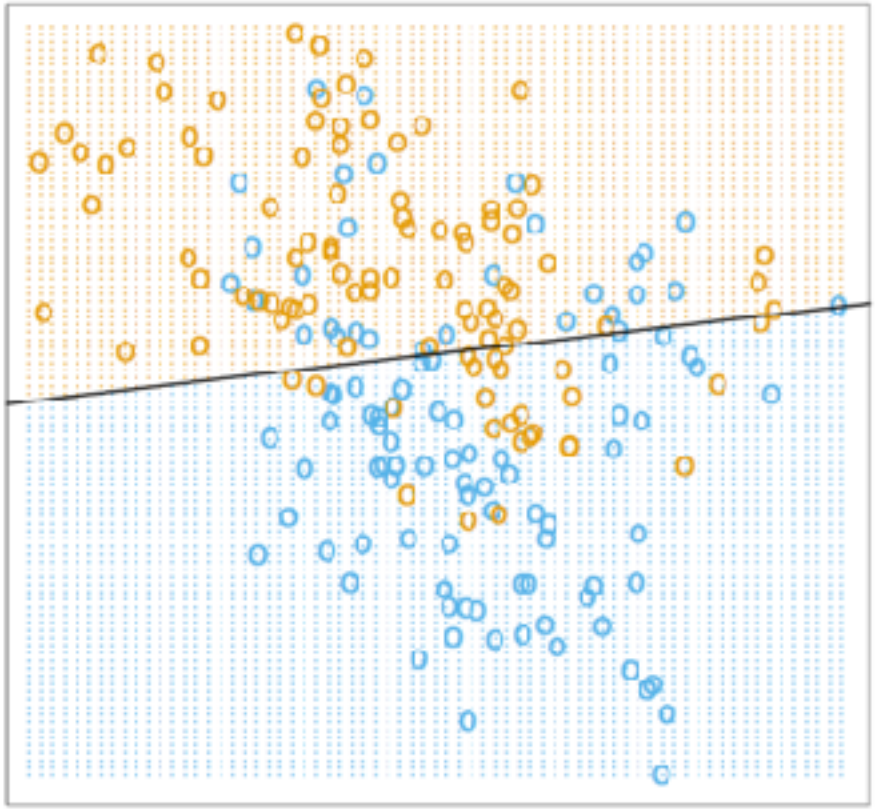
$$\mathbb{P}(Y = 1|X = x) = \frac{1}{2}$$

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

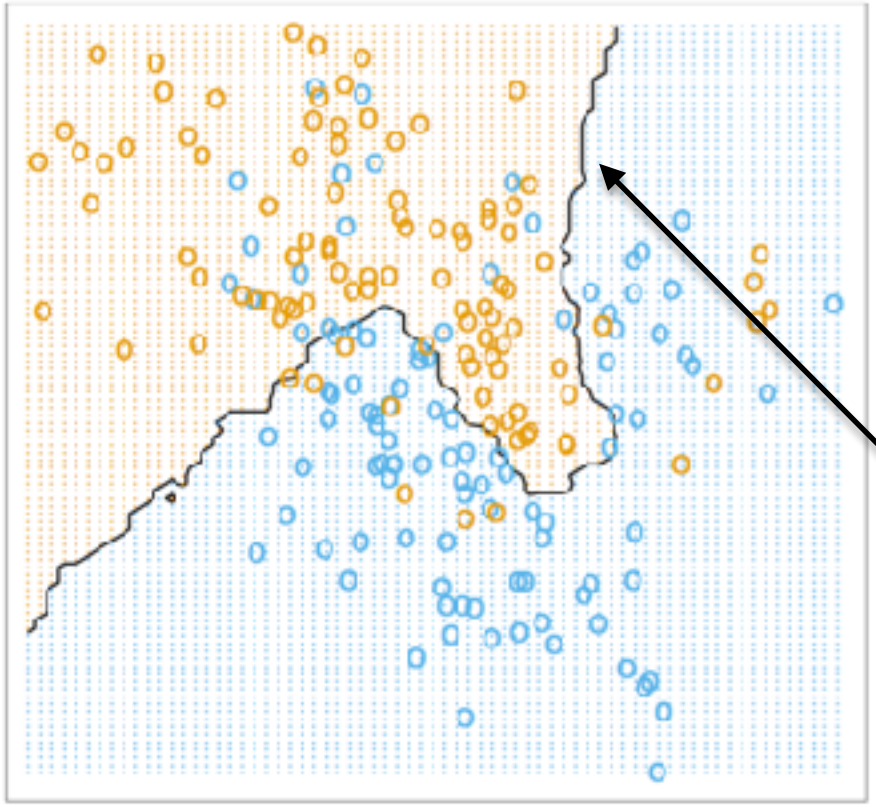
$$x^T w + b = 0$$

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:

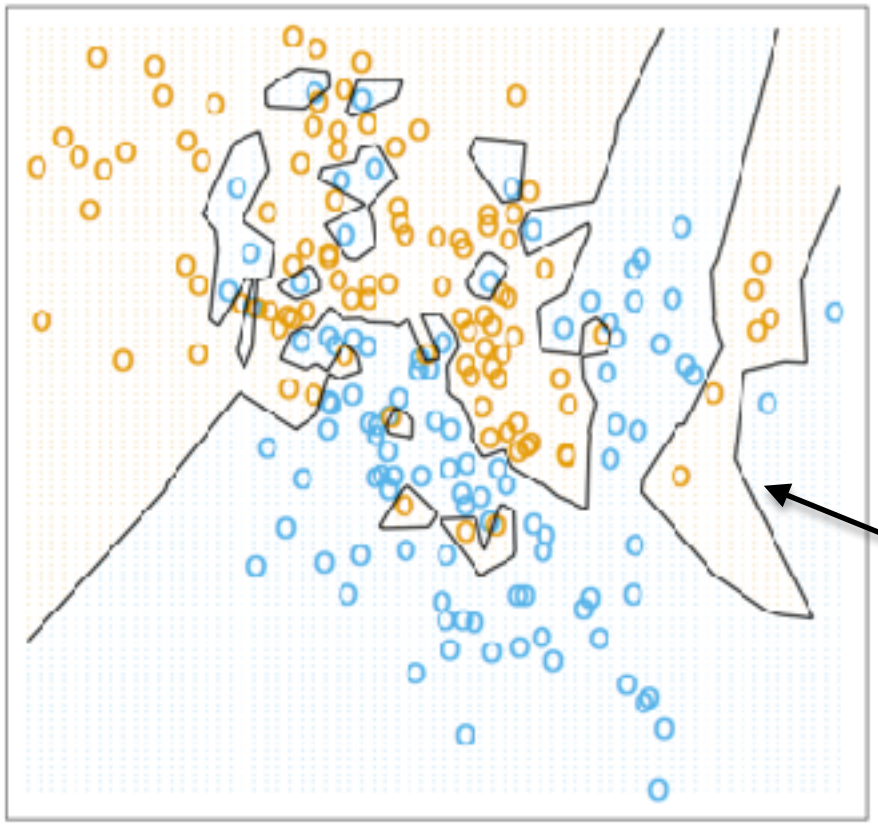
15 nearest neighbor decision boundary (majority vote)

Predicted label: +1

Predicted label: -1

Figures stolen from Hastie et al

1 Nearest Neighbor Boundary



Training data:

True label: +1

True label: -1

Learned:

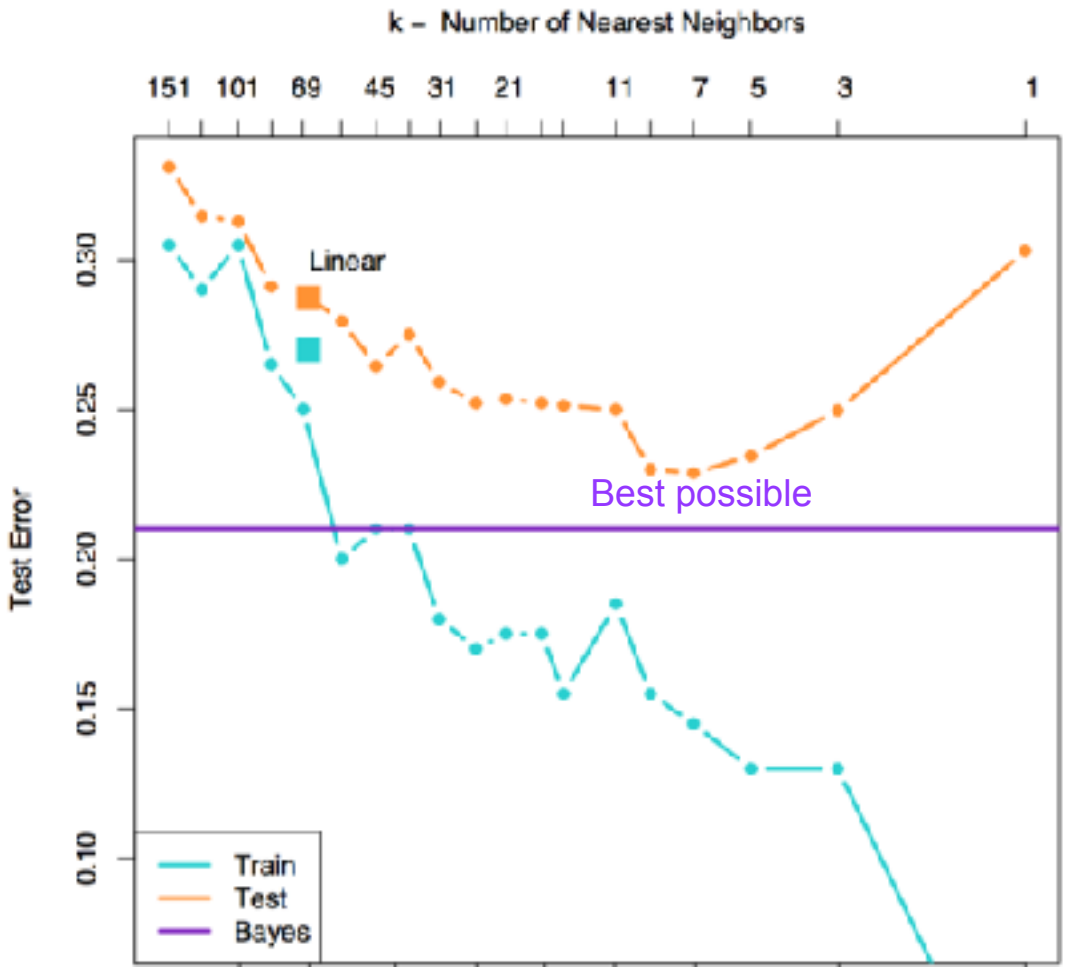
1 nearest neighbor decision boundary (majority vote)

Predicted label: +1

Predicted label: -1

Figures stolen from Hastie et al

k-Nearest Neighbor Error



Bias-Variance tradeoff

As k->infinity?

Bias:

Variance:

As k->1?

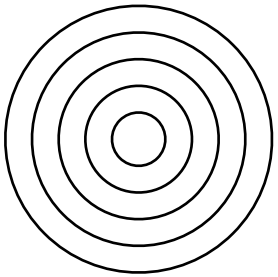
Bias:

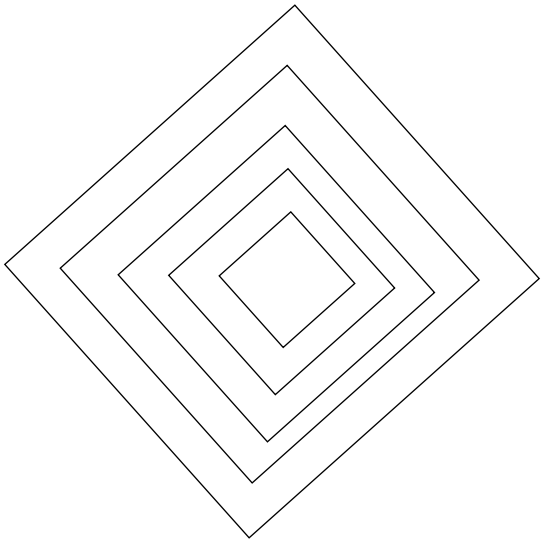
Variance:

Figures stolen from Hastie et al

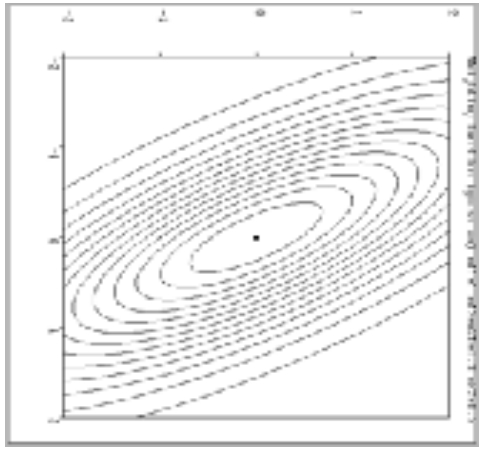
Notable distance metrics (and their level sets)

L_2 norm

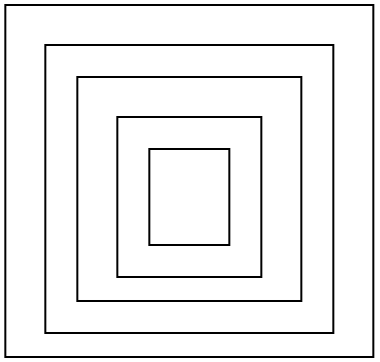




L_1 norm (taxi-cab)



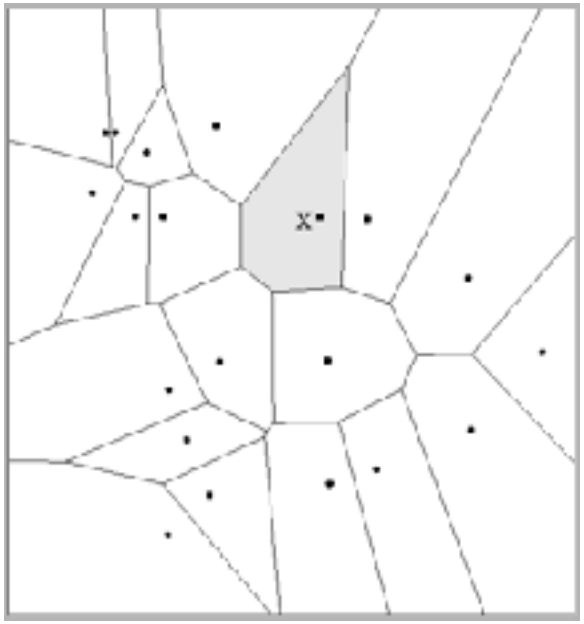
Mahalanobis



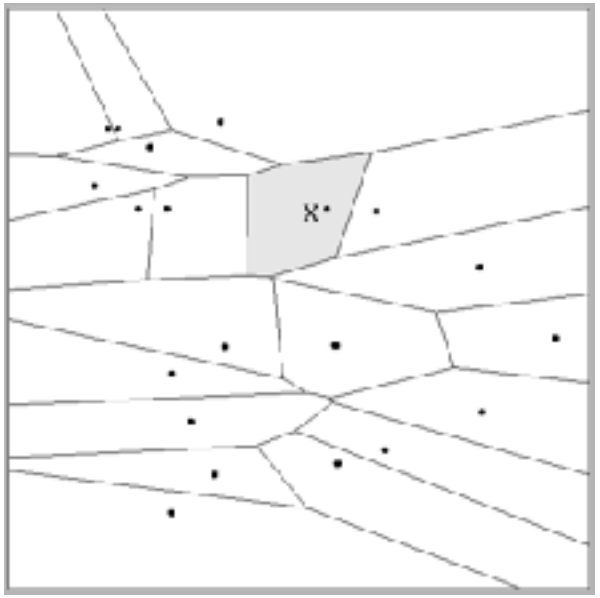
L-infinity *(max) norm*

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 - classification

$$\{(x_i, y_i)\})_{i=1}^n \qquad x_i \in \mathbb{R}^d, \quad y_i \in \{0, 1\} \qquad (x_i, y_i) \overset{iid}{\sim} P_{XY}$$

Theorem[Cover, Hart, 1967] If P_X is supported everywhere in $\mathbb{R}^d$ and $P(Y = 1|X = x)$ is smooth everywhere, then as $n \to \infty$ the 1-NN classification rule has error at most twice the Bayes error rate.

1 nearest neighbor guarantee - classification

$$\{(x_i, y_i)\})_{i=1}^n \qquad x_i \in \mathbb{R}^d, \quad y_i \in \{0,1\} \qquad (x_i, y_i) \overset{iid}{\sim} P_{XY}$$

Theorem[Cover, Hart, 1967] If P_X is supported everywhere in $\mathbb{R}^d$ and $P(Y = 1|X = x)$ is smooth everywhere, then as $n \to \infty$ the 1-NN classification rule has error at most twice the Bayes error rate.

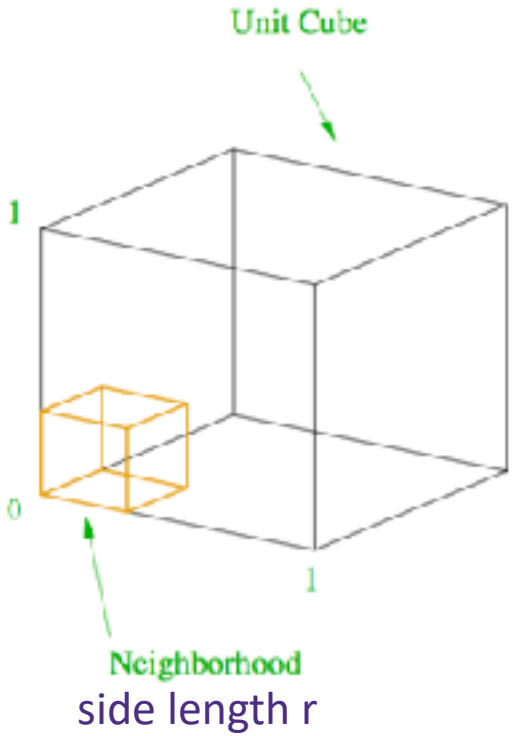
As $x_a \to x_b$ we have $\mathbb{P}(Y_a = 1|X_a = x_a) \to \mathbb{P}(Y_b = 1|X_b = x_b)$

If $p_* = \max_{y=0,1} \mathbb{P}(Y_b = y|X_b = x_b)$ then the Bayes Error $= 1 - p_*$

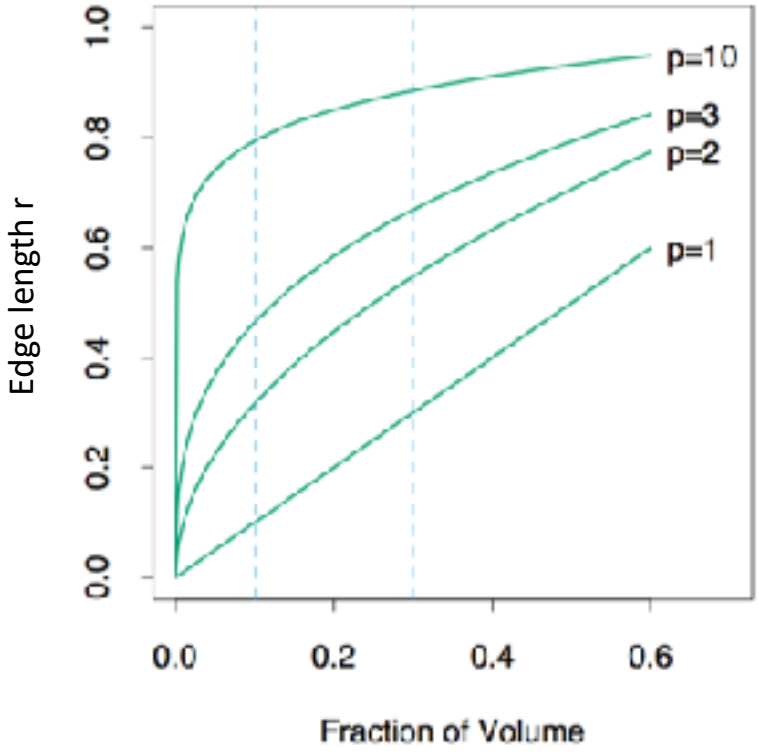
1-nearest neighbor error $=$

$$\lim_{n\to\infty} \mathbb{P}(\widehat{f}_{1NN}(x_a) \neq Y_a|X_a = x_a) = \mathbb{P}(Y_b \neq Y_a|X_a = x_b, X_b = x_b)$$
$$= \mathbb{P}(Y_b = 1|X_b = x_b)\mathbb{P}(Y_a = 0|X_a = x_b) + \mathbb{P}(Y_b = 0|X_b = x_b)\mathbb{P}(Y_a = 1|X_a = x_b)$$
$$= 2p_*(1 - p_*) \leq 2(1 - p_*)$$

Curse of dimensionality Ex. 1



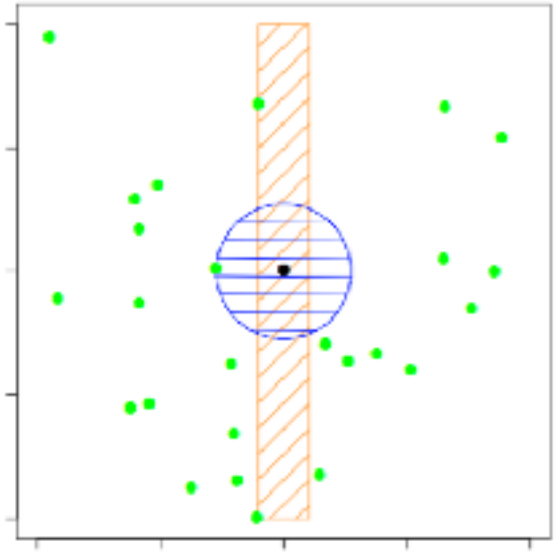
side length r

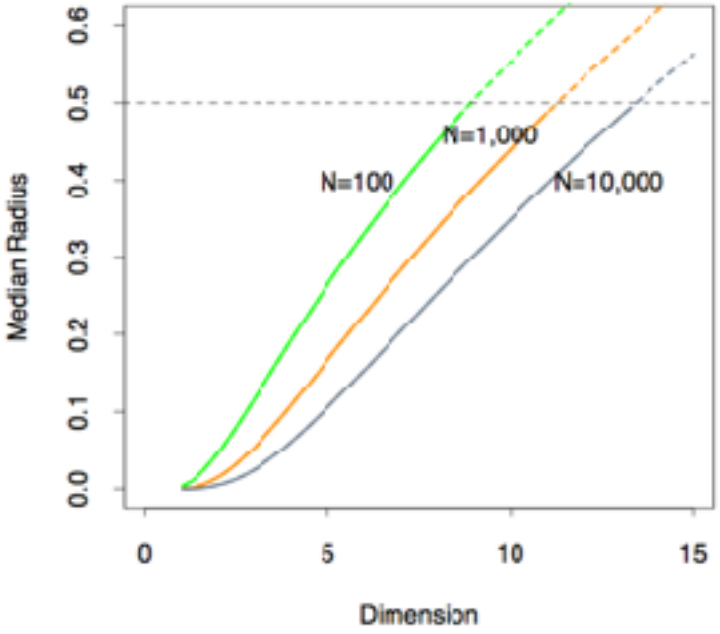


X is uniformly distributed over $[0,1]^p$. What is $\mathbb{P}(X \in [0,r]^p)$?

Curse of dimensionality Ex. 2

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

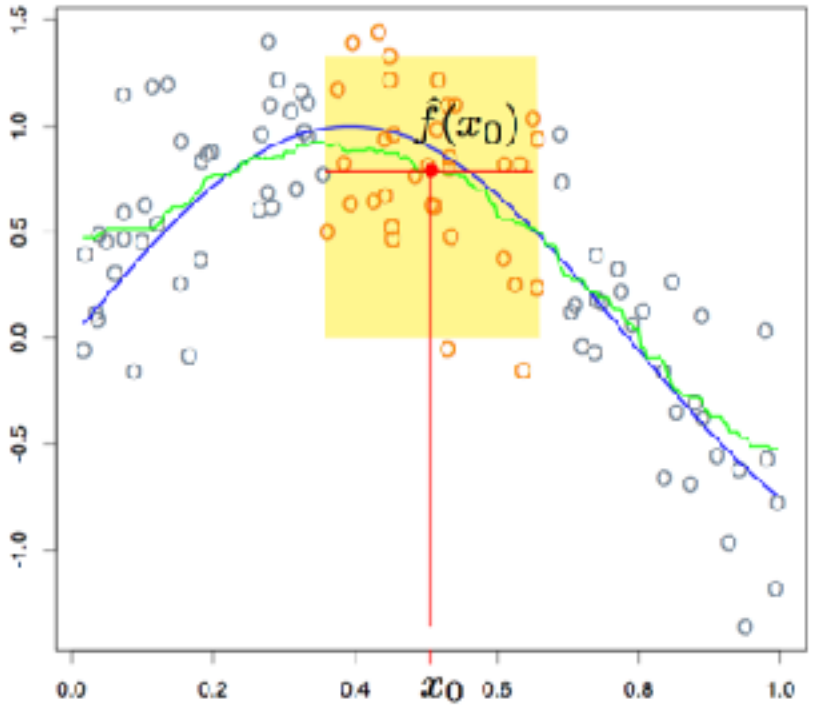




What is the median distance from a point at origin to its 1NN?

Nearest neighbor regression

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

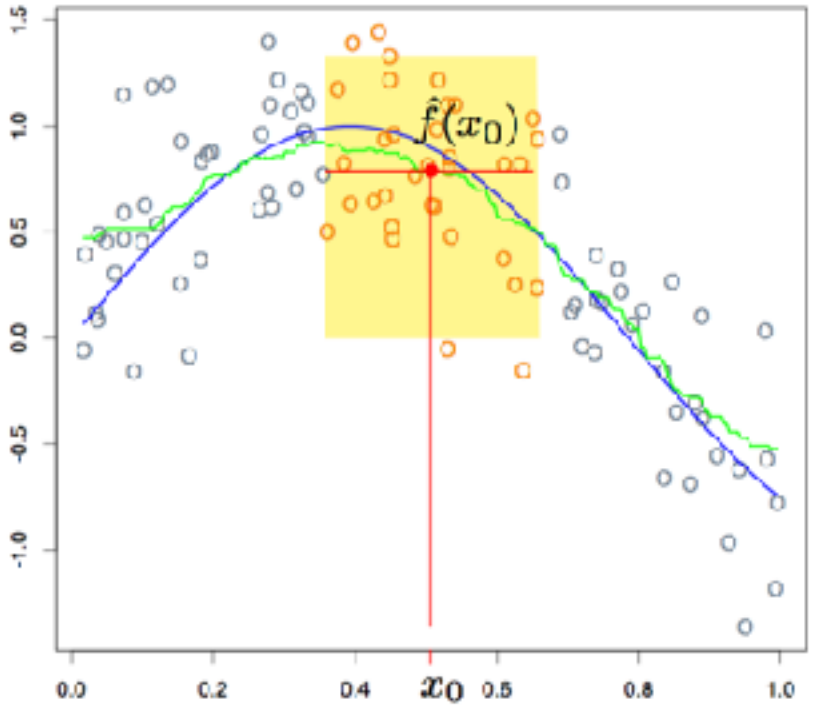


$$\mathcal{N}_k(x_0) = k\text{-nearest neighbors of } x_0$$

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

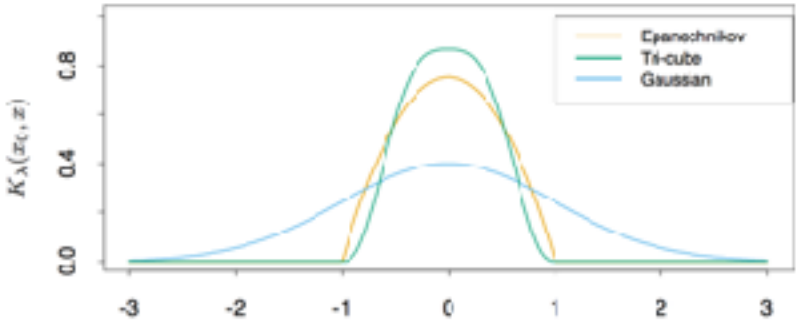
Nearest neighbor regression

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



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

Kernel smoothing: $K(x, y)$



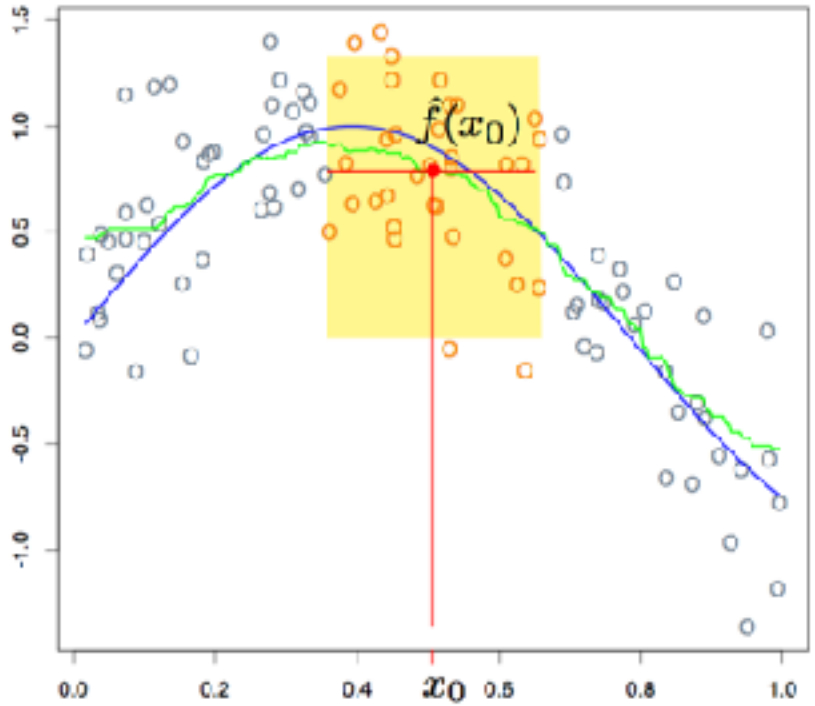
$$\mathcal{N}_k(x_0) = k\text{-nearest neighbors of } x_0$$

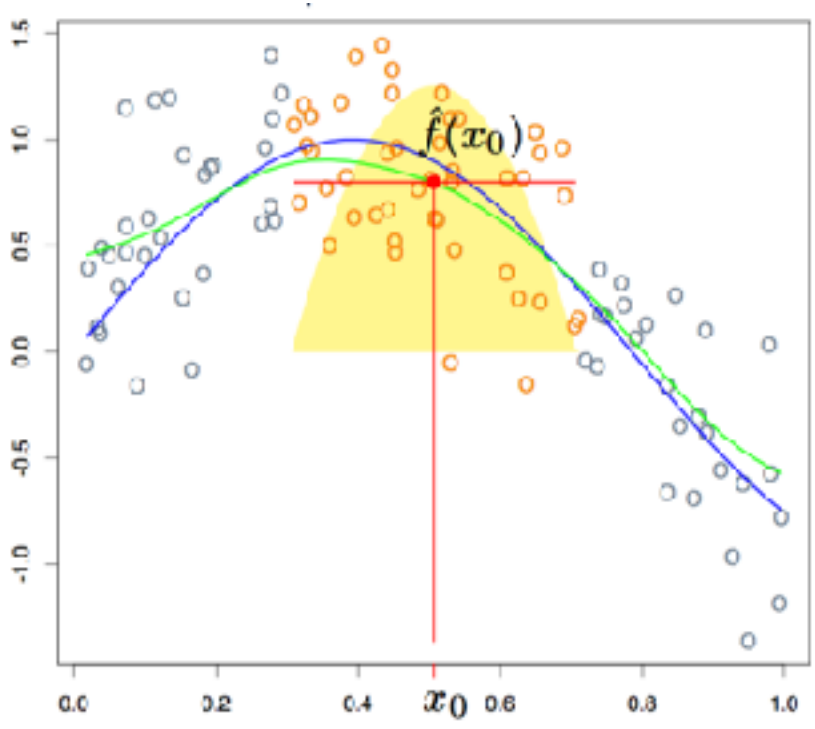
$$\widehat{f}(x_0) = \sum_{x_i \in \mathcal{N}_k(x_0)} \frac{1}{k}\, y_i$$

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

Nearest neighbor regression

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





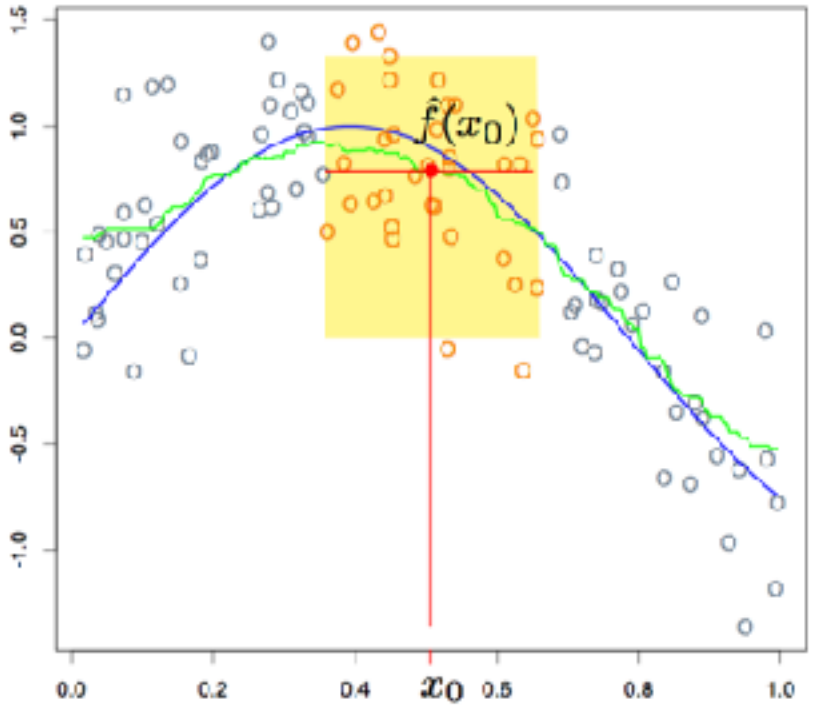
$\mathcal{N}_k(x_0) = k$-nearest neighbors of x_0

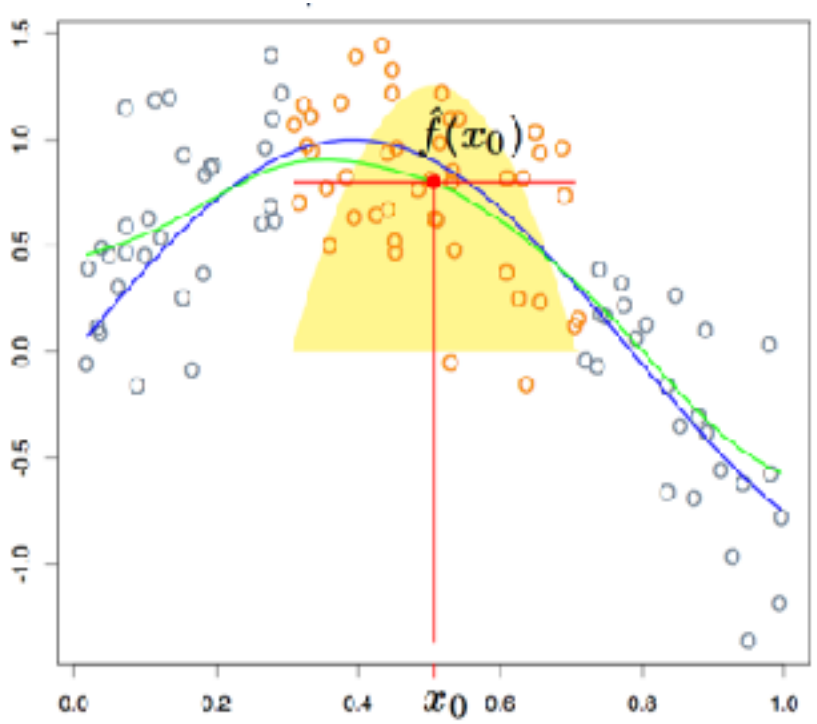
$$\widehat{f}(x_0) = \sum_{x_i \in \mathcal{N}_k(x_0)} \frac{1}{k}\, y_i$$

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

Nearest neighbor regression

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





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

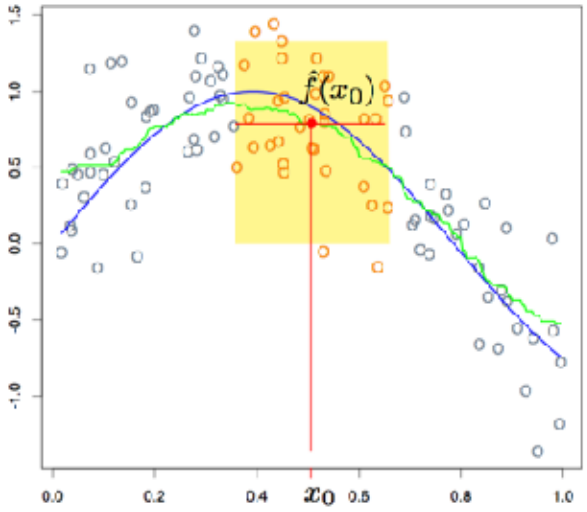
$$\widehat{f}(x_0) = \sum_{x_i \in \mathcal{N}_k(x_0)} \frac{1}{k}\, y_i$$

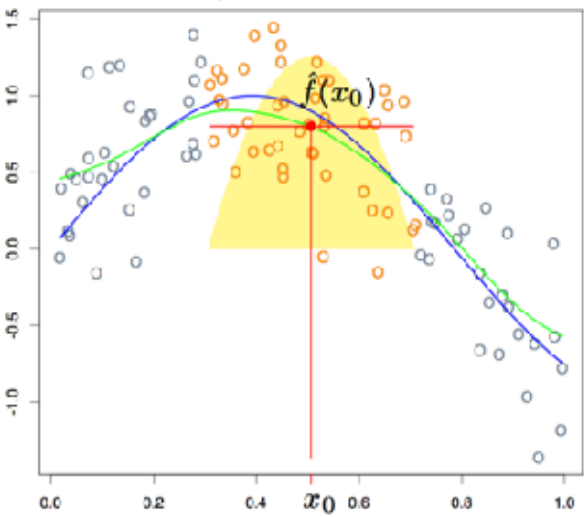
Why just average them?

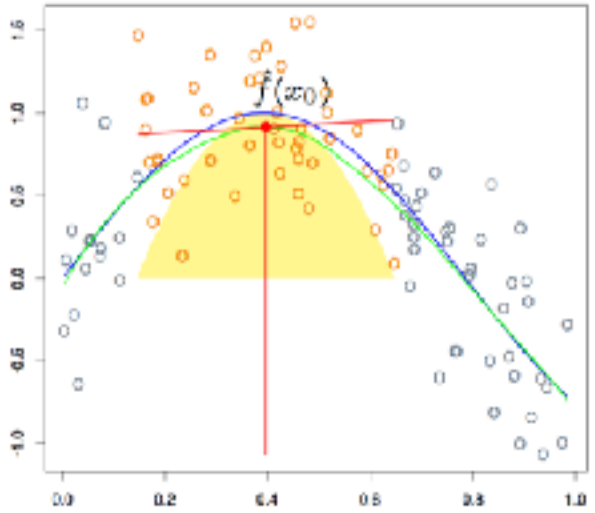
$$\widehat{f}(x_0) = \frac{\sum_{i=1}^n K(x_0, x_i) y_i}{\sum_{i=1}^n K(x_0, x_i)}$$

Nearest neighbor regression

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







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

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

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

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

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

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.
- Without a lot of data, neighborhoods aren’t “local” and methods suffer.