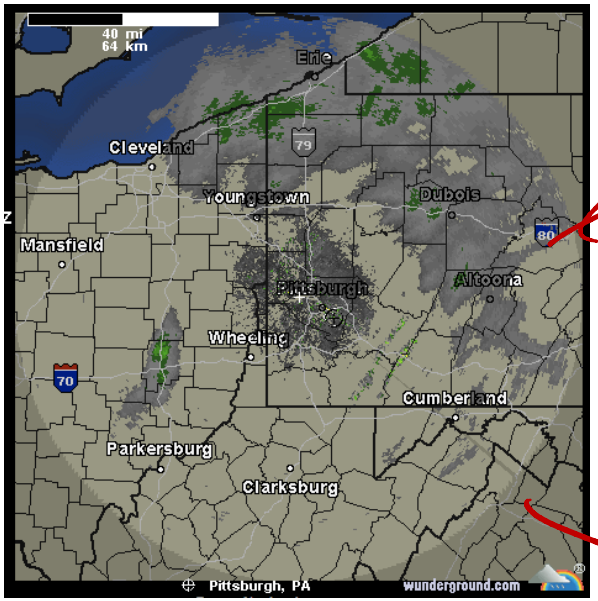
Classification! Logistic "Regression" (linear classification), oh my…



Thus far, regression:

predict a continuous value given some inputs

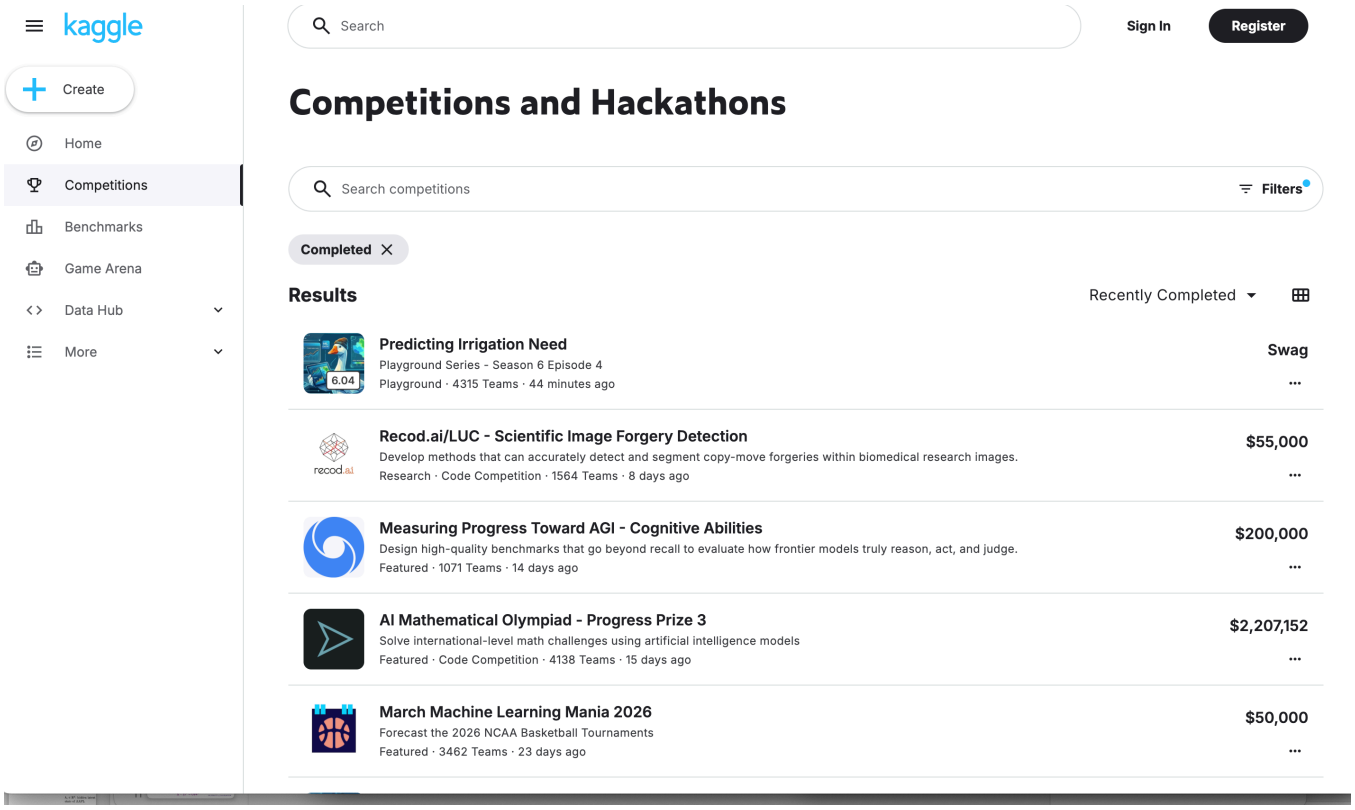
Weather prediction revisted







Recent Kaggle competitions



Classification

- Learn f: X -> Y
  - X - features
  - Y - target classes

- Loss Function
- Expected loss of f:

- Suppose you knew P(YIX) exactly, how should you classify?
  - Bayes-Optimal classifier:

Classification

- Learn f: X -> Y
  - X - features
  - Y - target classes

- Loss Function

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

- Expected loss of f:

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

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

$$= 1 - P(Y = f(x)|X = x)$$

- Suppose you knew P(YIX) exactly, how should you classify?
  - Bayes-Optimal classifier:

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

Binary Classification

- Learn f: X -> Y
  - X - features
  - Y - target classes

$$Y \in \{0, 1\}$$

- Loss Function

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

- Expected loss of f:

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

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

$$= 1 - P(Y = f(x)|X = x)$$

- Suppose you knew P(YIX) exactly, how should you classify?
  - Bayes-Optimal classifier:

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

Bayes Optimal Binary Classifier

$$Y \in \{0, 1\}$$

- Suppose you knew P(YIX) exactly, how should you classify?
  - Bayes-Optimal classifier:

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

- Suppose we don't know P(YIX), but have n iid examples

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

- What is a natural estimator for P(Y I X)?

Bayes Optimal Binary Classifier

- **Suppose we don't know P(YIX), but have n iid examples**

$$\{(x_i, y_i)\}_{i=1}^n \qquad Y \in \{0, 1\}$$

- **What is a natural estimator for P(Y I X)?**

$$\text{Fix some } \tilde{x} \in X$$

$$\text{Suppose } x_i = \tilde{x} \text{ for } m \leq n \text{ samples}$$

$$\text{What is a natural estimator for } \theta_* := \mathbb{P}(Y = 1 | X = \tilde{x})?$$

$$\text{If } k \text{ of the } m \text{ labels are equal to } Y = 1 \text{ then}$$

Bayes Optimal Binary Classifier

- **Suppose we don't know P(YIX), but have n iid examples**

$$\{(x_i, y_i)\}_{i=1}^n \qquad Y \in \{0, 1\}$$

- **What is a natural estimator for argmax_y P(Y = y I X)?**

$$\text{If } X = \{0,1\}^d, \text{ or is generally discrete}$$

$$\hat{f}(x) = \arg\max_{y \in \{0,1\}} \frac{\sum_{i=1}^n \mathbf{1}[\mathbf{x_i}=\mathbf{x}, \mathbf{y_i}=\mathbf{y}]}{\sum_{i=1}^n \mathbf{1}[\mathbf{x_i}=\mathbf{x}]}$$

Issues?

Bayes Optimal Binary Classifier

- **What is a natural estimator for argmax_y P(Y = y I X)?**

If $X = \{0,1\}^d$, or is generally discrete $\qquad Y \in \{0,1\}$

$$\hat{f}(x) = \arg\max_{y \in \{0,1\}} \frac{\sum_{i=1}^{n} \mathbf{1}[\mathbf{x_i}=\mathbf{x},\mathbf{y_i}=\mathbf{y}]}{\sum_{i=1}^{n} \mathbf{1}[\mathbf{x_i}=\mathbf{x}]}$$

Issues?

2^d possible inputs, for small d requires huge n

To make predictions for unseen inputs (xs),

need a **general** model for $\mathbb{P}(Y = 1|X = x)$

Process

Decide on a **model**

Find the function which fits the data best

- **Choose a loss function**
- **Pick the function which minimizes loss on data**

Use function to make prediction on new examples

©2018 Kevin Jamieson

Decide on a model, Binary Classification

To make predictions for unseen inputs (xs),

need a **general** model for $\mathbb{P}(Y = 1|X = x)$

- **What about standard linear regression model?**

- **Need to map real values to [0,1]**
  - **We call such maps "link functions"**

©2018 Kevin Jamieson

Logistic Regression

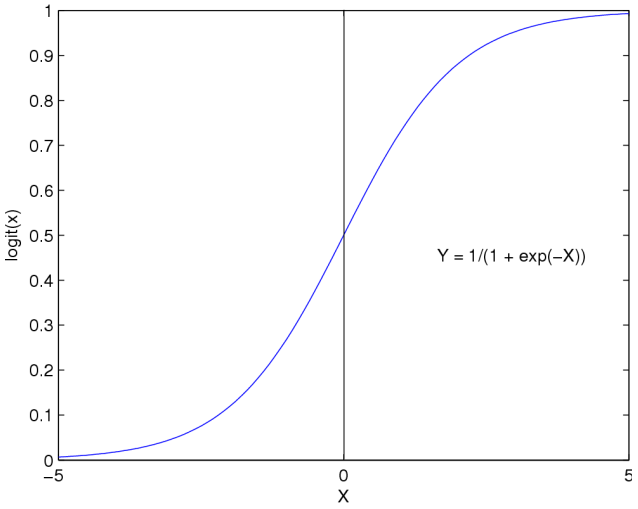
Actually classification, not regression :)

$$\text{Learn } \mathbb{P}(Y = 1|X = x) \text{ using } \sigma(w^T x), \text{ for link function } \sigma =$$

Logistic function(or Sigmoid): $\dfrac{1}{1 + exp(-z)}$

$$\mathbb{P}[Y = 1|X = x, w] = \sigma(w^T x) = \frac{1}{1 + \exp(-w^T x)}$$

$$\mathbb{P}[Y = 0|X = x, w] = 1 - \sigma(w^T x) = \frac{\exp(-w^T x)}{1 + \exp(-w^T x)} = \frac{1}{1 + \exp(w^T x)}$$

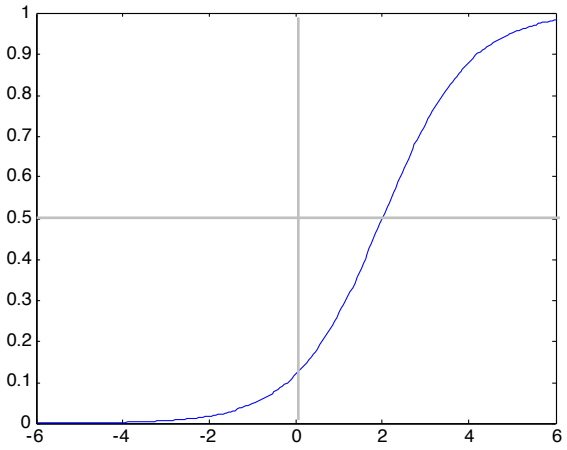


Features can be discrete or continuous!

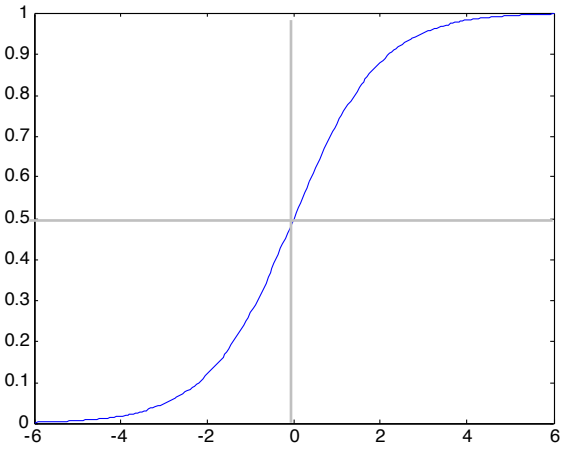
Understanding the sigmoid

$$\sigma(w_0 + \sum_k w_k x_k) = \frac{1}{1 + e^{w_0 + \sum_k w_k x_k}}$$

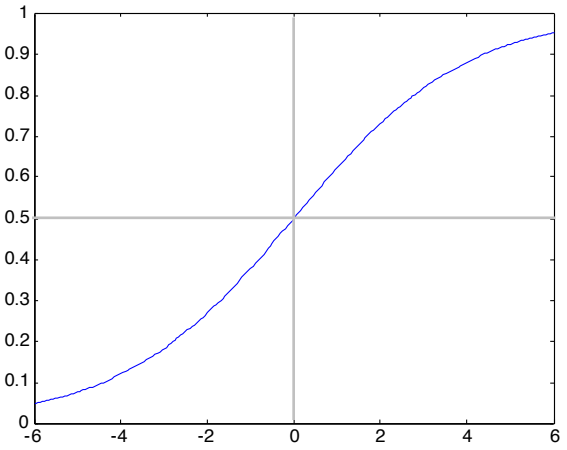
w_0=-2, w_1=-1



w_0=0, w_1=-1



w_0=0, w_1=-0.5



Sigmoid for binary classes

$$\mathbb{P}(Y=0|w,X) = \frac{1}{1+\exp(w_0+\sum_k w_k X_k)}$$

$$\mathbb{P}(Y=1|w,X) = 1-\mathbb{P}(Y=0|w,X) = \frac{\exp(w_0+\sum_k w_k X_k)}{1+\exp(w_0+\sum_k w_k X_k)}$$

$$\frac{\mathbb{P}(Y=1|w,X)}{\mathbb{P}(Y=0|w,X)} =$$

Sigmoid for binary classes

$$\mathbb{P}(Y=0|w,X) = \frac{1}{1+\exp(w_0+\sum_k w_k X_k)}$$

$$\mathbb{P}(Y=1|w,X) = 1-\mathbb{P}(Y=0|w,X) = \frac{\exp(w_0+\sum_k w_k X_k)}{1+\exp(w_0+\sum_k w_k X_k)}$$

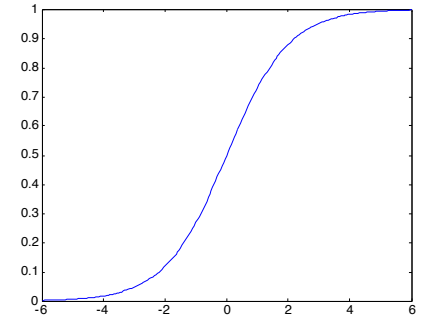
$$\frac{\mathbb{P}(Y=1|w,X)}{\mathbb{P}(Y=0|w,X)} = \exp(w_0+\sum_k w_k X_k)$$

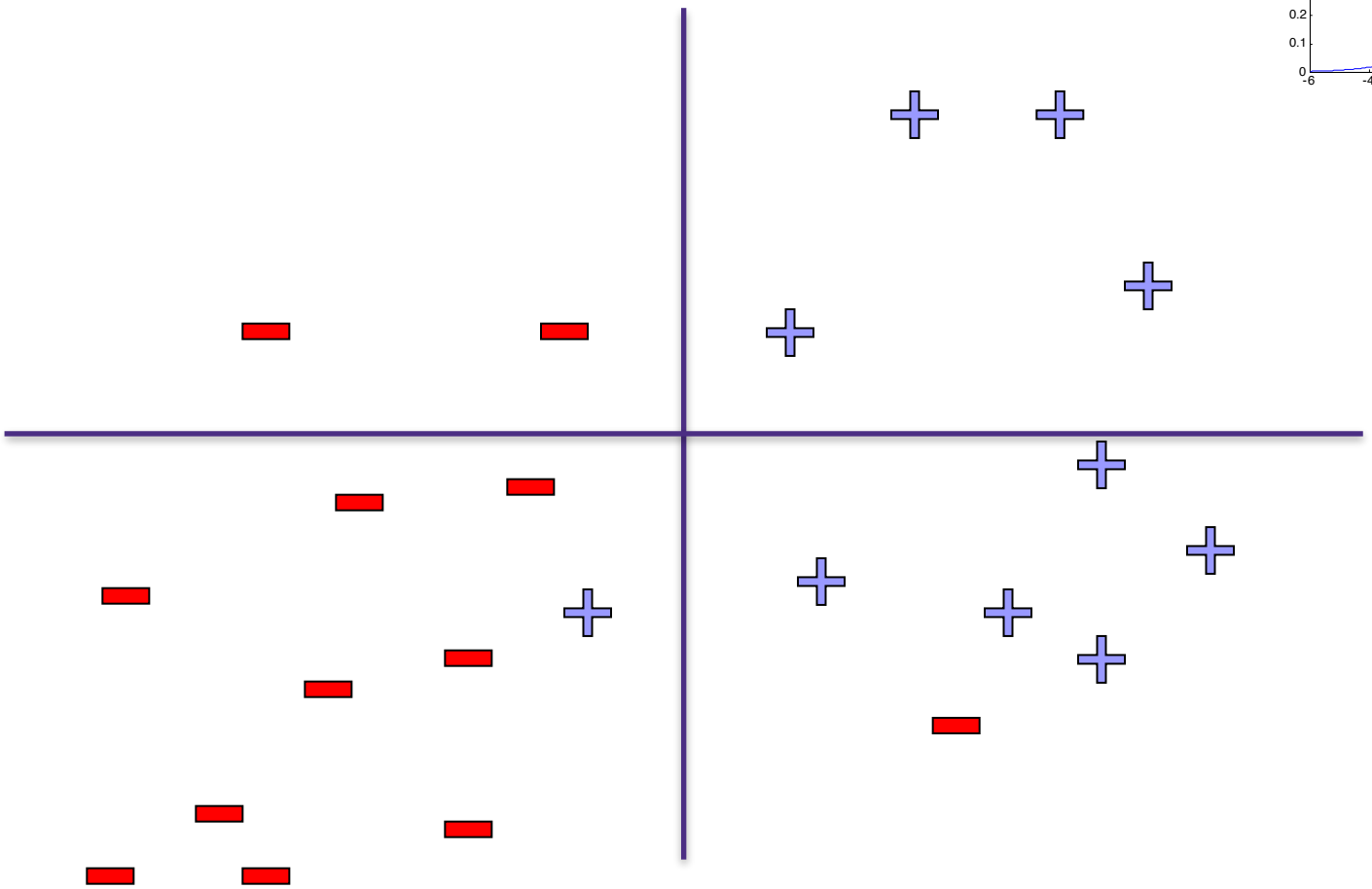
Linear Decision Rule!

$$\log\frac{\mathbb{P}(Y=1|w,X)}{\mathbb{P}(Y=0|w,X)} = w_0+\sum_k w_k X_k$$

Logistic Regression – a Linear classifier

$$\frac{1}{1+exp(-z)}$$





$$\ln \frac{P(Y=0|X)}{P(Y=1|X)} = w_0 + \sum_i w_i X_i$$

Process

Decide on a **model**

Find the function which fits the data best

- **Choose a loss function**
- **Pick the function which minimizes loss on data**

Use function to make prediction on new examples

©2018 Kevin Jamieson

Loss function: Conditional Likelihood

- **Have a bunch of iid data:** $\{(x_i, y_i)\}_{i=1}^{n} \quad x_i \in \mathbb{R}^d, \quad y_i \in \{-1, 1\}$

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

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

- **This is equivalent to:**

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

- **So we can compute the maximum likelihood estimator:**

$$\widehat{w}_{MLE} = \arg\max_{w} \prod_{i=1}^{n} P(y_i|x_i, w)$$

Loss function: Conditional Likelihood

- **Have a bunch of iid data:** $\{(x_i, y_i)\}_{i=1}^{n} \quad x_i \in \mathbb{R}^d, \quad y_i \in \{-1, 1\}$

$$\widehat{w}_{MLE} = \arg\max_w \prod_{i=1}^{n} P(y_i|x_i, w) \qquad P(Y = y|x, w) = \frac{1}{1 + \exp(-y\, w^T x)}$$

$$= \arg\min_w \sum_{i=1}^{n} \log(1 + \exp(-y_i\, x_i^T w))$$

Loss function: Conditional Likelihood

- **Have a bunch of iid data:** $\{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d, \quad y_i \in \{-1, 1\}$

$$\widehat{w}_{MLE} = \arg\max_w \prod_{i=1}^n P(y_i|x_i, w) \qquad P(Y = y|x, w) = \frac{1}{1 + \exp(-y\, w^T x)}$$

$$= \arg\min_w \sum_{i=1}^n \log(1 + \exp(-y_i\, x_i^T w))$$

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

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

Process

Decide on a **model**

Find the function which fits the data best

- **Choose a loss function**
- **Pick the function which minimizes loss on data**

Use function to make prediction on new examples

©2018 Kevin Jamieson

Loss function: Conditional Likelihood

- **Have a bunch of iid data:** $\{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d, \quad y_i \in \{-1, 1\}$

$$\widehat{w}_{MLE} = \arg\max_w \prod_{i=1}^n P(y_i|x_i, w) \qquad P(Y = y|x, w) = \frac{1}{1 + \exp(-y\, w^T x)}$$

$$= \arg\min_w \sum_{i=1}^n \log(1 + \exp(-y_i\, x_i^T w)) = J(w)$$

What does $J(w)$ look like? Is it convex?

Loss function: Conditional Likelihood

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

$$\widehat{w}_{MLE} = \arg\max_w \prod_{i=1}^n P(y_i|x_i, w) \qquad P(Y = y|x, w) = \frac{1}{1 + \exp(-y\, w^T x)}$$

$$= \arg\min_w \sum_{i=1}^n \log(1 + \exp(-y_i\, x_i^T w)) = J(w)$$

Good news: $J(\mathbf{w})$ is convex function of $\mathbf{w}$, no local optima problems

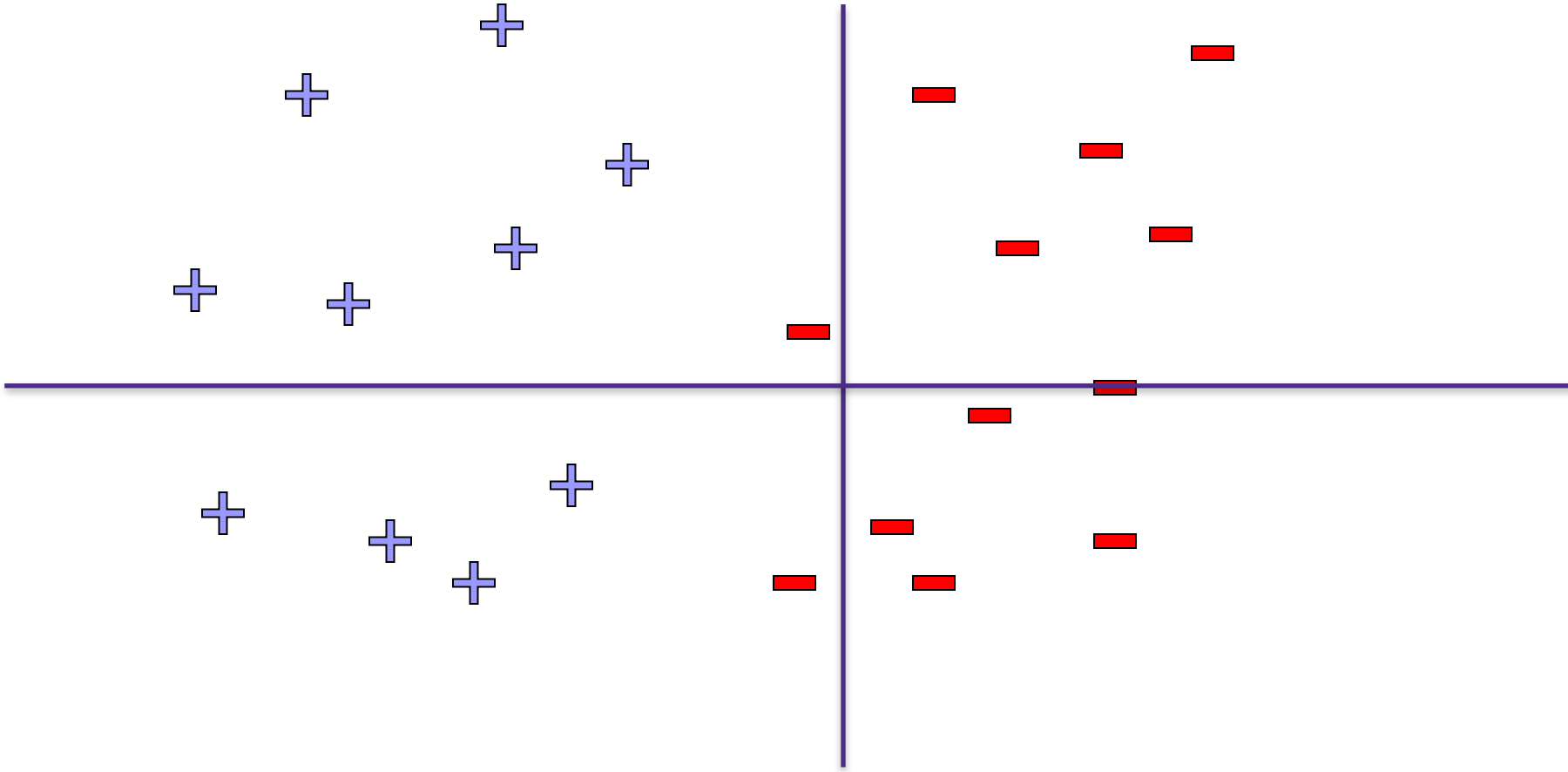
Bad news: no closed-form solution to maximize $J(\mathbf{w})$

Good news: convex functions easy to optimize

Linear Separability

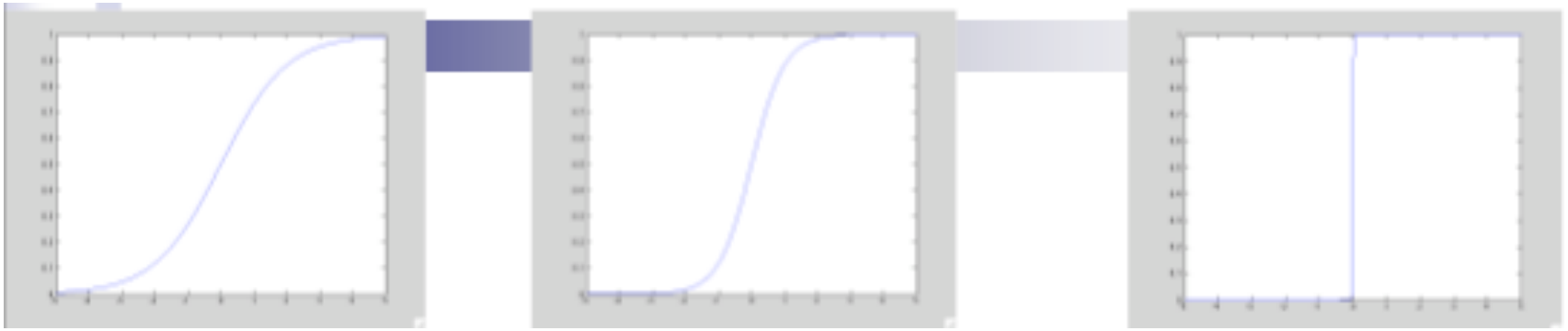
$$\arg\min_{w} \sum_{i=1}^{n} \log(1 + \exp(-y_i \, x_i^T w))$$

When is this loss small?



Large parameters → Overfitting

When data is linearly separable, weights ⇒ ∞



$$\frac{1}{1+e^{-x}} \qquad \frac{1}{1+e^{-2x}} \qquad \frac{1}{1+e^{-100x}}$$

Overfitting

Penalize high weights to prevent overfitting?

Regularized Conditional Log Likelihood

Add a penalty to avoid high weights/overfitting?:

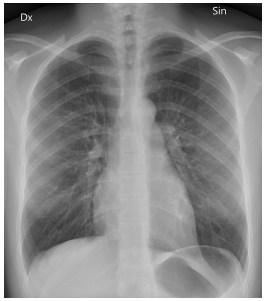
$$\arg\min_{w,b} \sum_{i=1}^{n} \log\left(1 + \exp(-y_i\left(x_i^T w + b\right))\right) + \lambda||w||_2^2$$

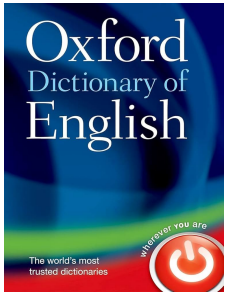
Be sure to not regularize the offset b!

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





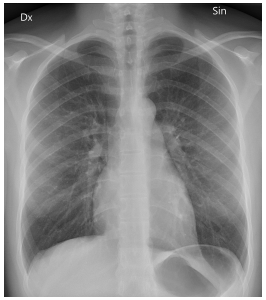


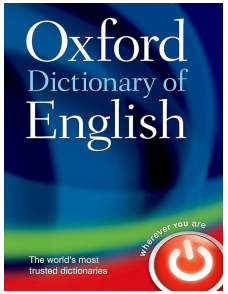
- A k-class classifier predicts y given x

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:

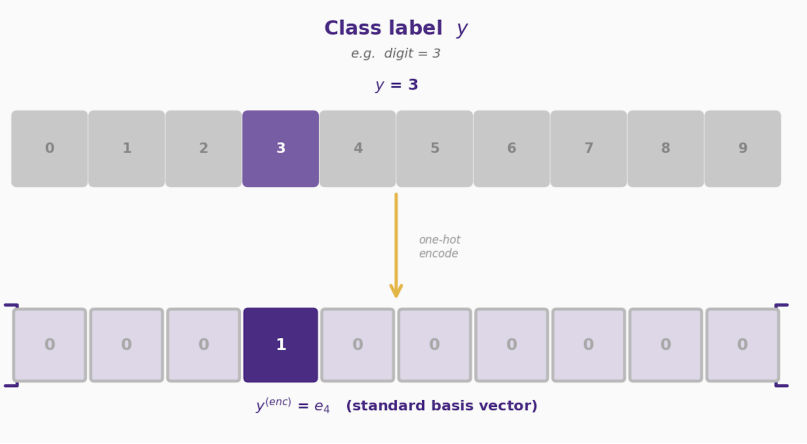
- Try to predict $\mathbb{P}[c_i \mid x]$ versus $\mathbb{P}[\text{not } c_i \mid x]$ for each i
- Try to predict $\mathbb{P}[c_i \mid x]$ as a linear function $w_i \cdot x$ for each i

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 $\mathbb{P}[c_i \mid x]$ 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



$$w_1 \cdot x \qquad \mathbb{P}[c_1 \mid x] \qquad e^{w_1 \cdot x}/N$$
$$w_2 \cdot x \qquad \mathbb{P}[c_2 \mid x] \qquad e^{w_2 \cdot x}/N$$
$$\vdots \qquad \vdots \qquad \vdots$$
$$w_k \cdot x \qquad \mathbb{P}[c_k \mid x] \qquad e^{w_k \cdot x}/N$$

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: $\ell\left(f_w(x),\ y\right) = \log(1 + \exp\left(-y f_w(x)\right))$

Regression and classification

- Can we treat classification as a regression problem?

Yes: could regress to a continuous output y and round to a valid label

Is this a good idea?

No, no smooth relationship between label values

- Can we treat regression as a classification problem?

Yes: classify which "bin" the true number falls into: c_1 = [0-10], c_2 = [11-20], …

Is this a good idea?

You lose some precision, but it actually can be. Sometimes easier to optimize. Can win a kaggle competition this way. Look into "distributional RL"

The ML pipeline

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

Support Vector Machines

UNIVERSITY of WASHINGTON

©2018 Kevin Jamieson



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

©2018 Kevin Jamieson

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:**

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

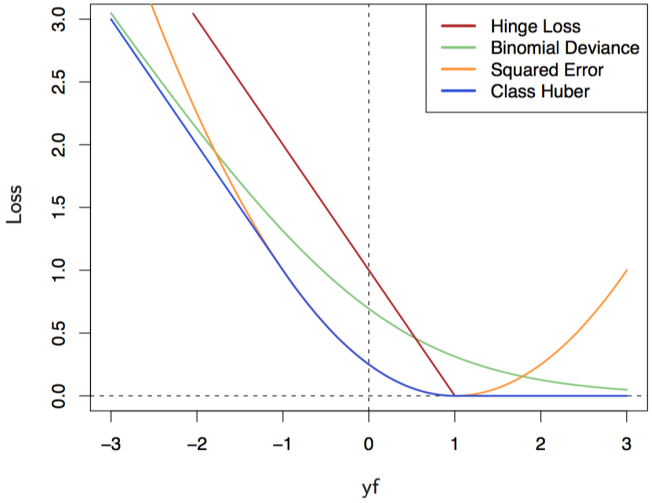
- **Loss function:**

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

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

©2018 Kevin Jamieson

Loss Functions



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

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

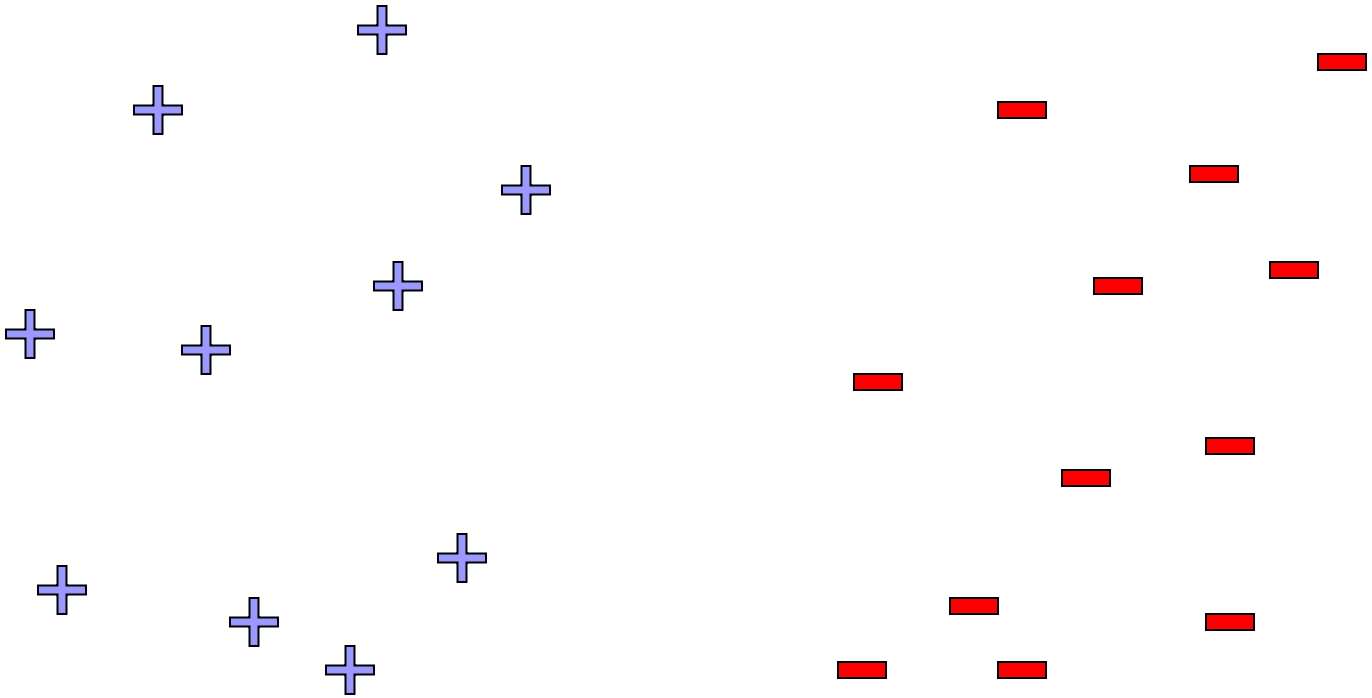
Squared error Loss: $\ell_i(w) = (y_i - x_i^T w)^2$

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

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

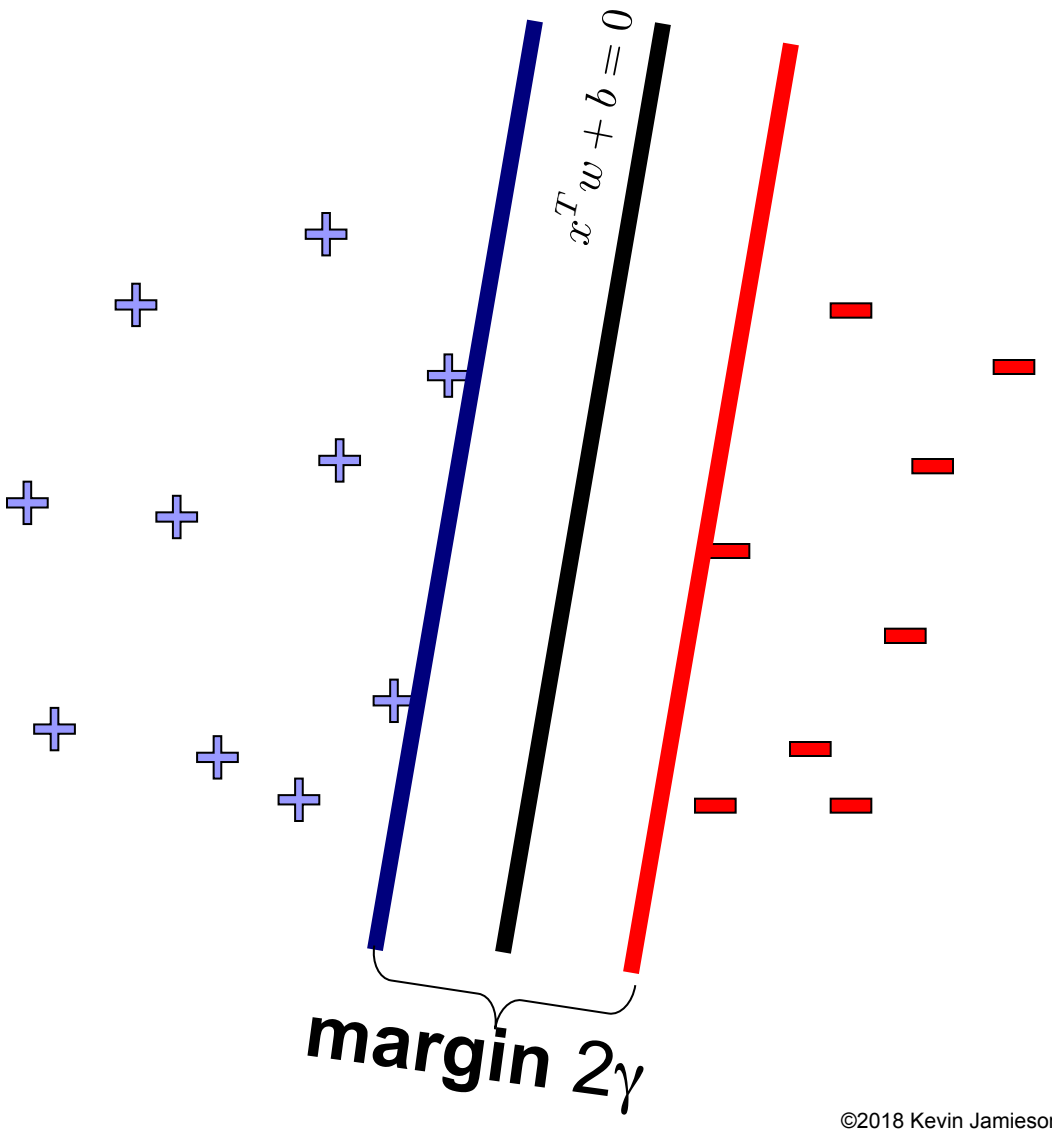
Hinge Loss: $\ell_i(w) = \max\{0, 1 - y_i x_i^T w\}$

Linear classifiers – Which line is better?



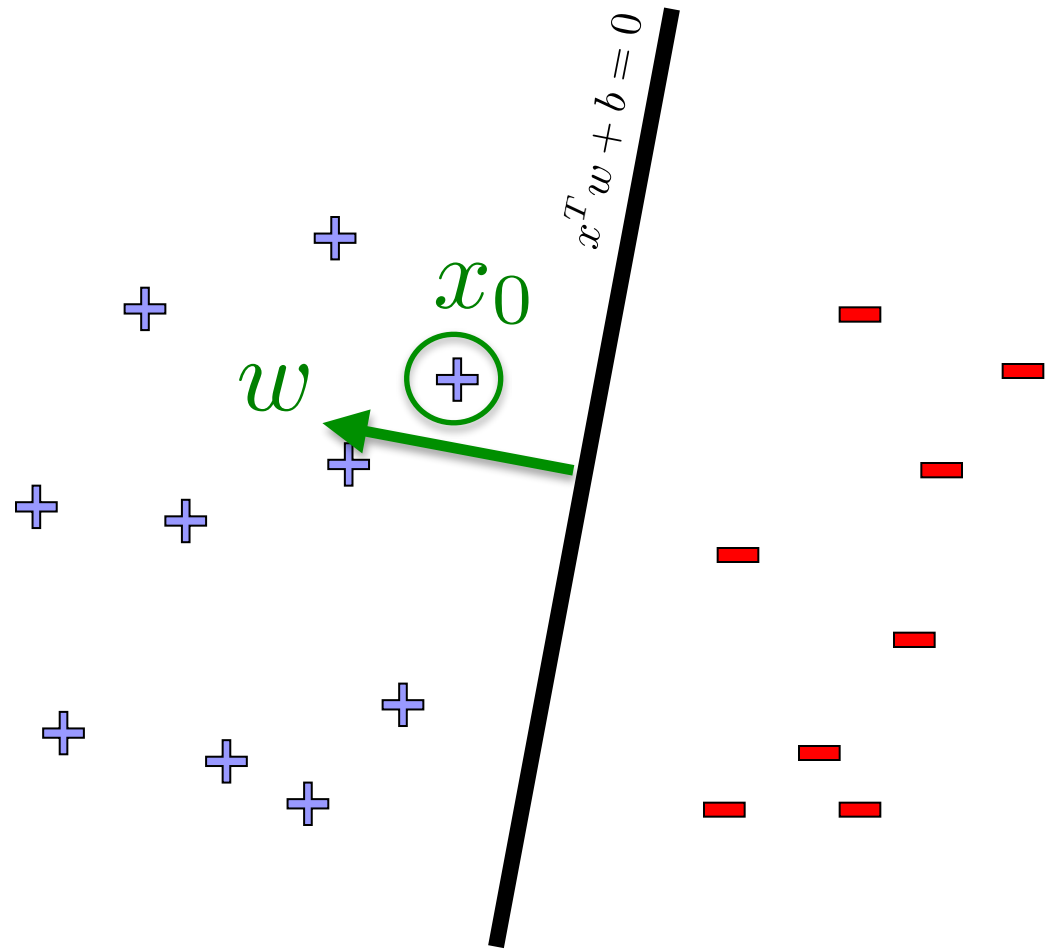
©2018 Kevin Jamieson

Pick the one with the largest margin!



©2018 Kevin Jamieson

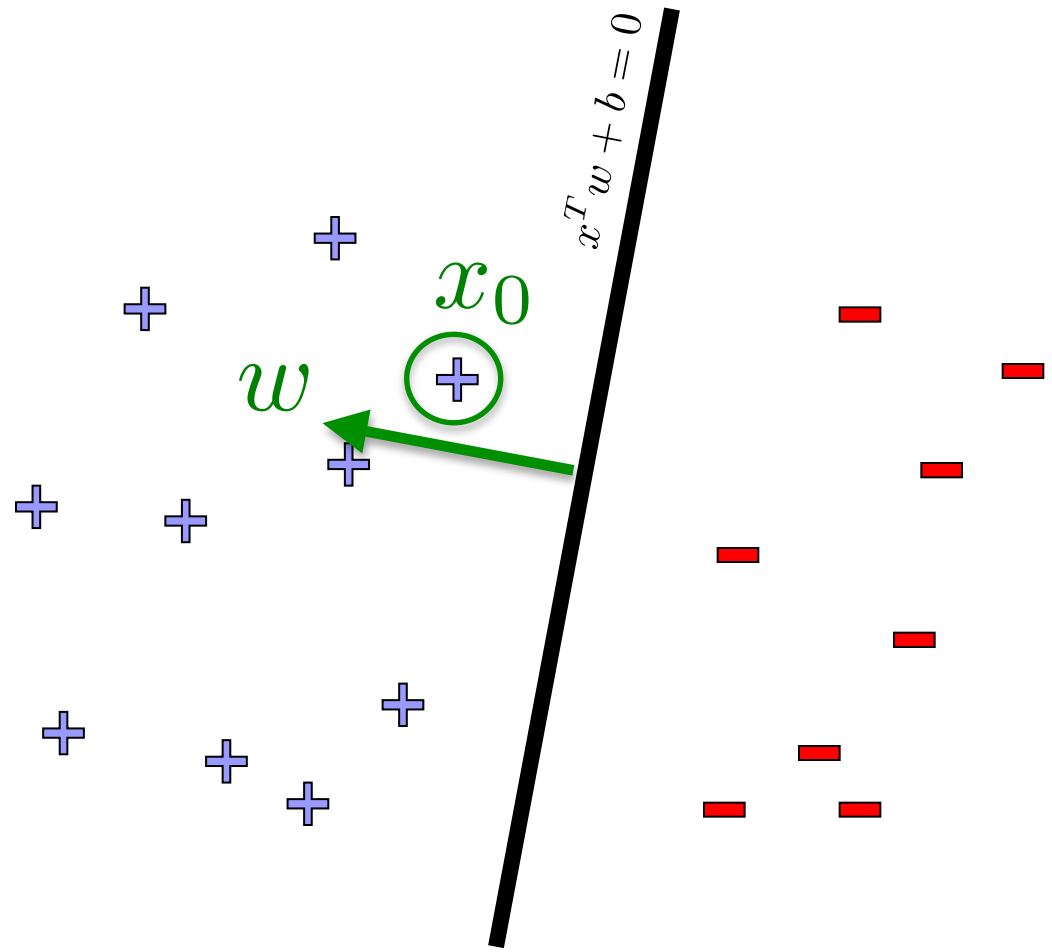
Pick the one with the largest margin!



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

©2018 Kevin Jamieson

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

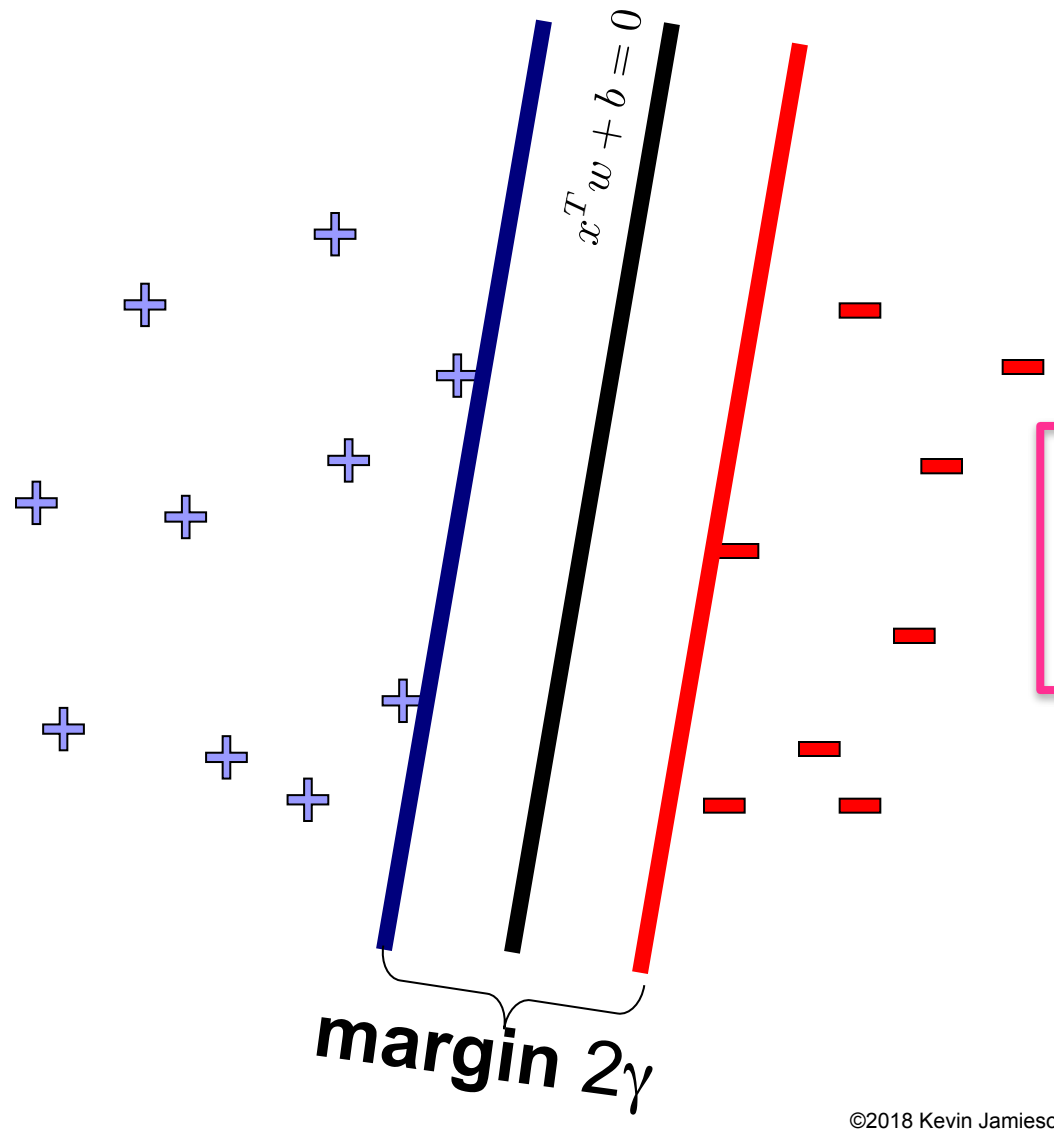
$$||x_0 - \widetilde{x}_0||_2 = |(x_0^T - \widetilde{x}_0)^T \frac{w}{||w||_2}|$$

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

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

©2018 Kevin Jamieson

Pick the one with the largest margin!



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

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

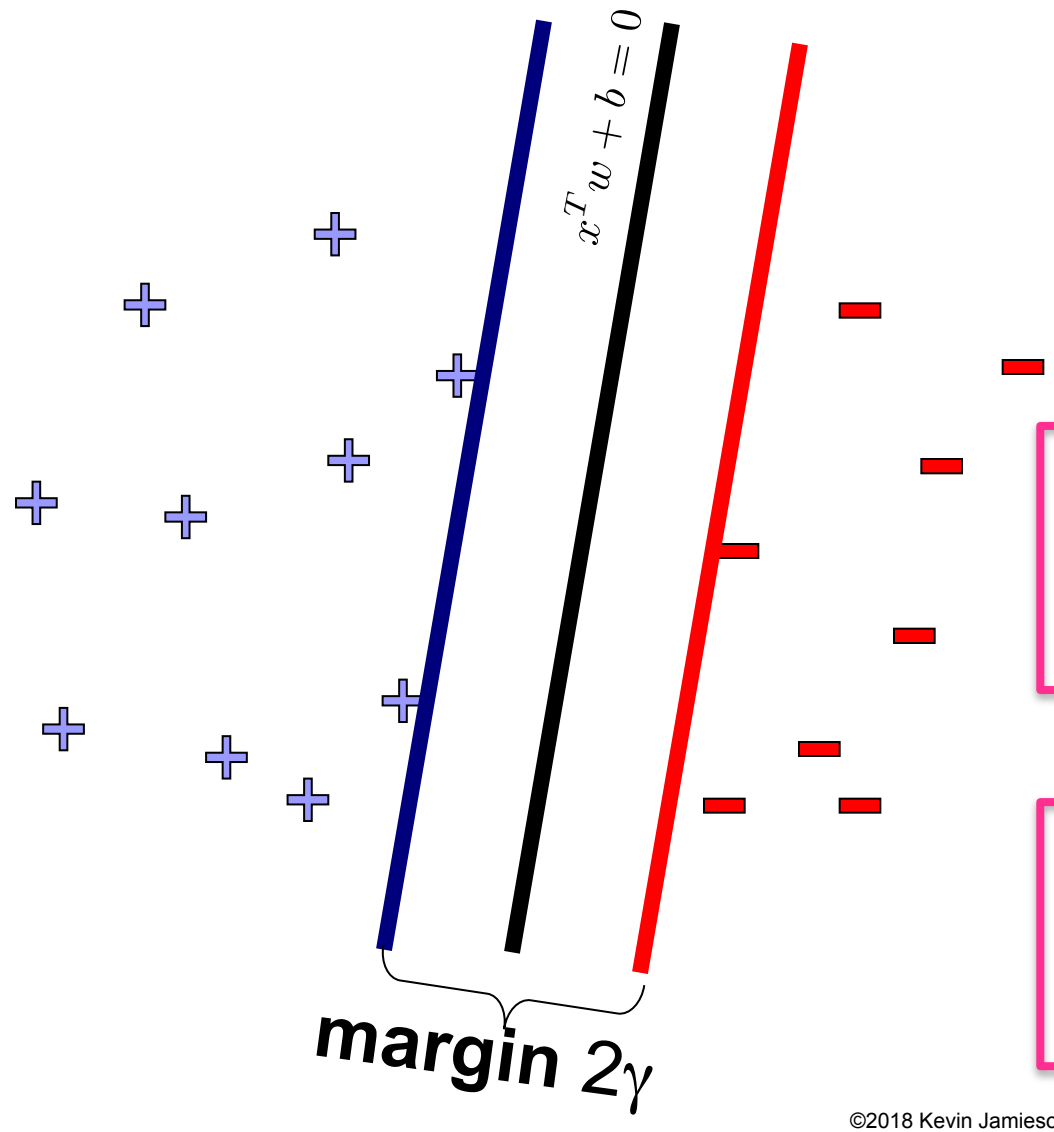
Optimal Hyperplane

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

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

©2018 Kevin Jamieson

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

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

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

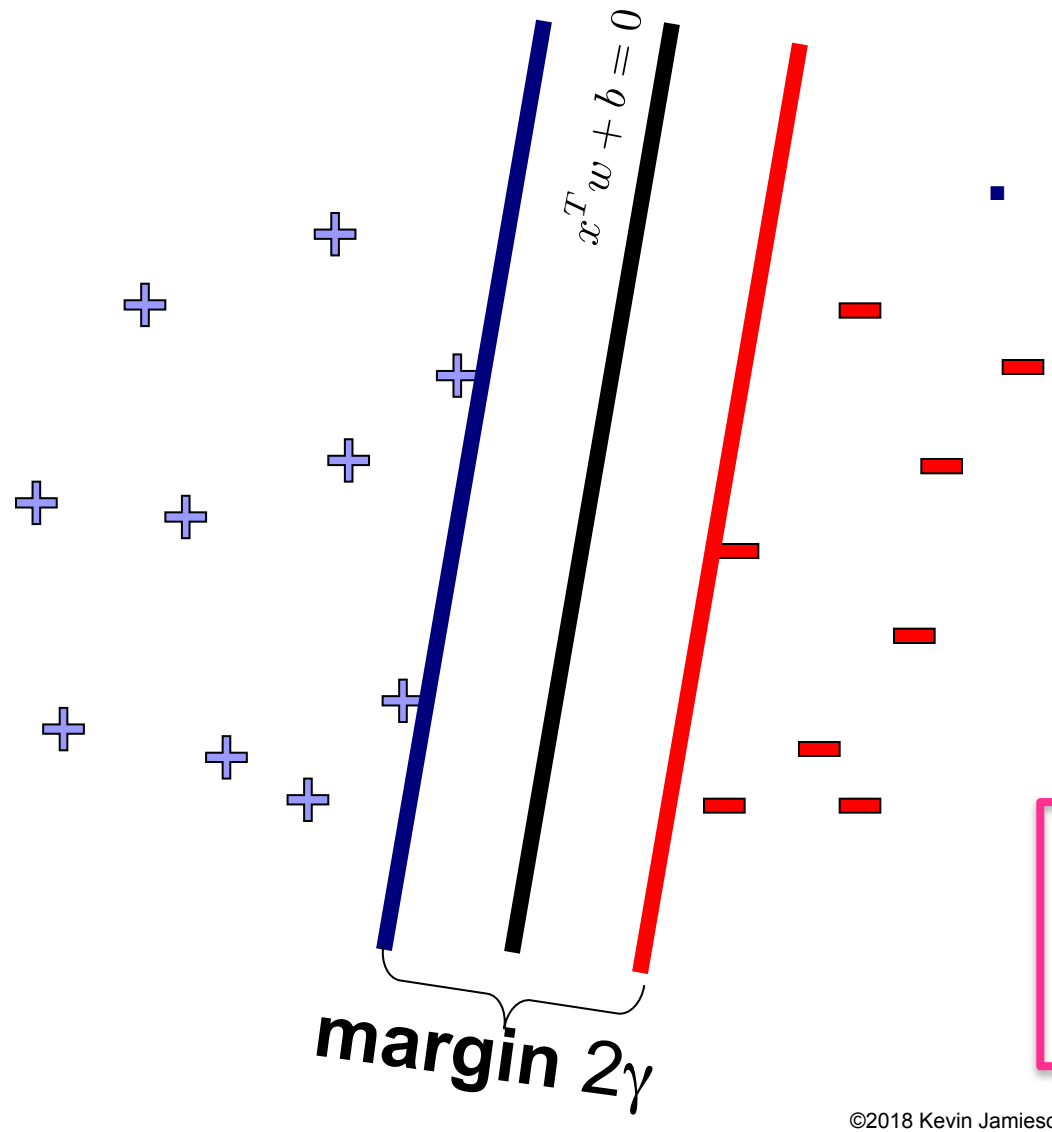
Optimal Hyperplane (reparameterized)

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

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

©2018 Kevin Jamieson

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)

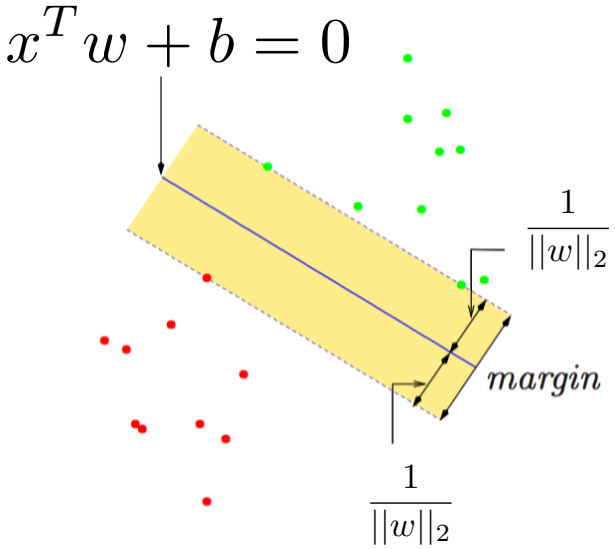
Optimal Hyperplane (reparameterized)

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

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

©2018 Kevin Jamieson

What if the data is not linearly separable?



If data is linearly separable

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

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

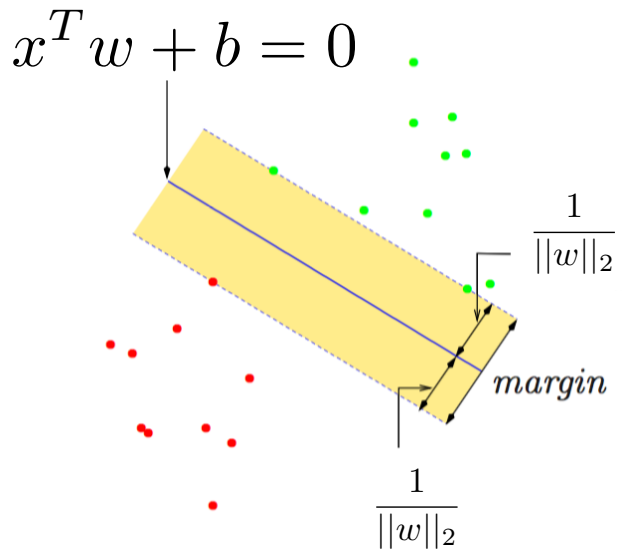
If data is not linearly separable, some points don't satisfy margin constraint:

Two options:

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

©2018 Kevin Jamieson

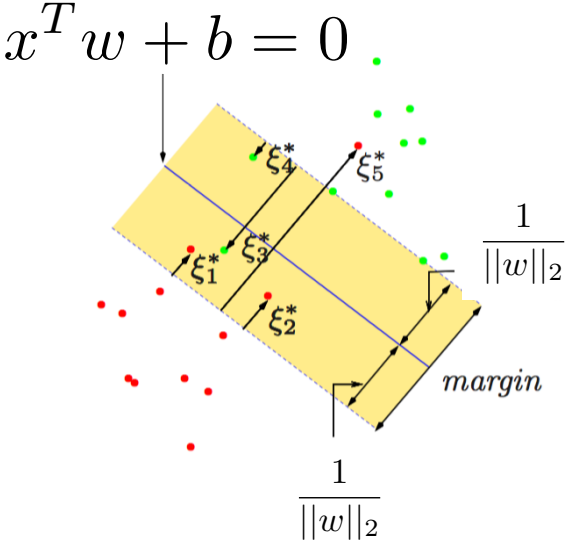
What if the data is not linearly separable?



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:

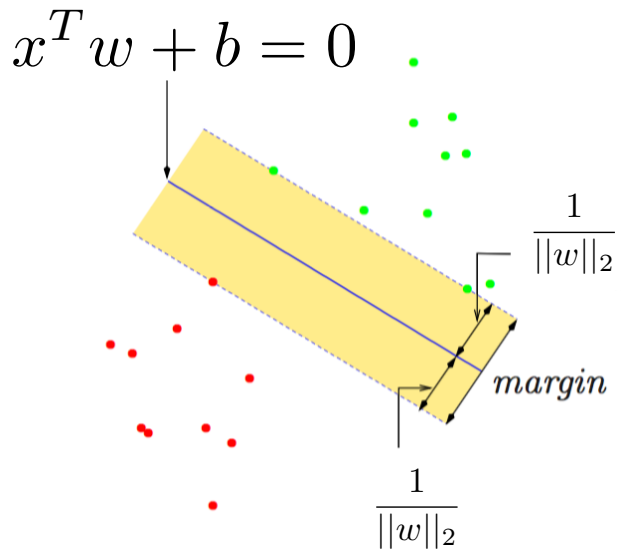
$$\min_{w,b} ||w||_2^2$$

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

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

©2018 Kevin Jamieson

What if the data is not linearly separable?



If data is linearly separable:

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

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

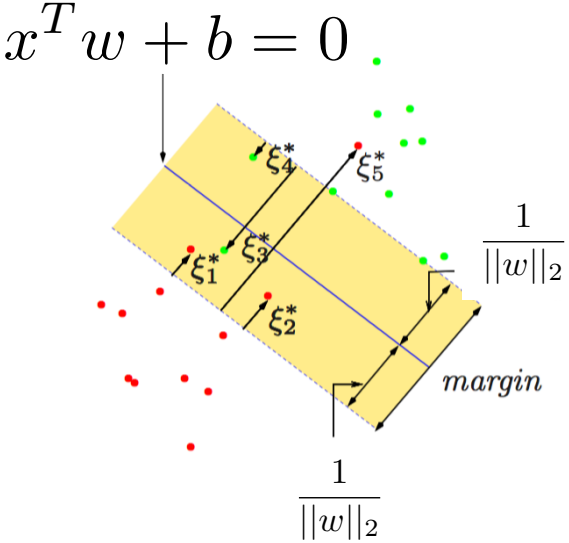
If data is not linearly separable,

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?"

©2018 Kevin Jamieson

SVM as penalization method

- Original quadratic program with linear constraints:

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

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

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

©2018 Kevin Jamieson

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:

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

©2018 Kevin Jamieson

SVMs: optimizing what?

SVM objective:

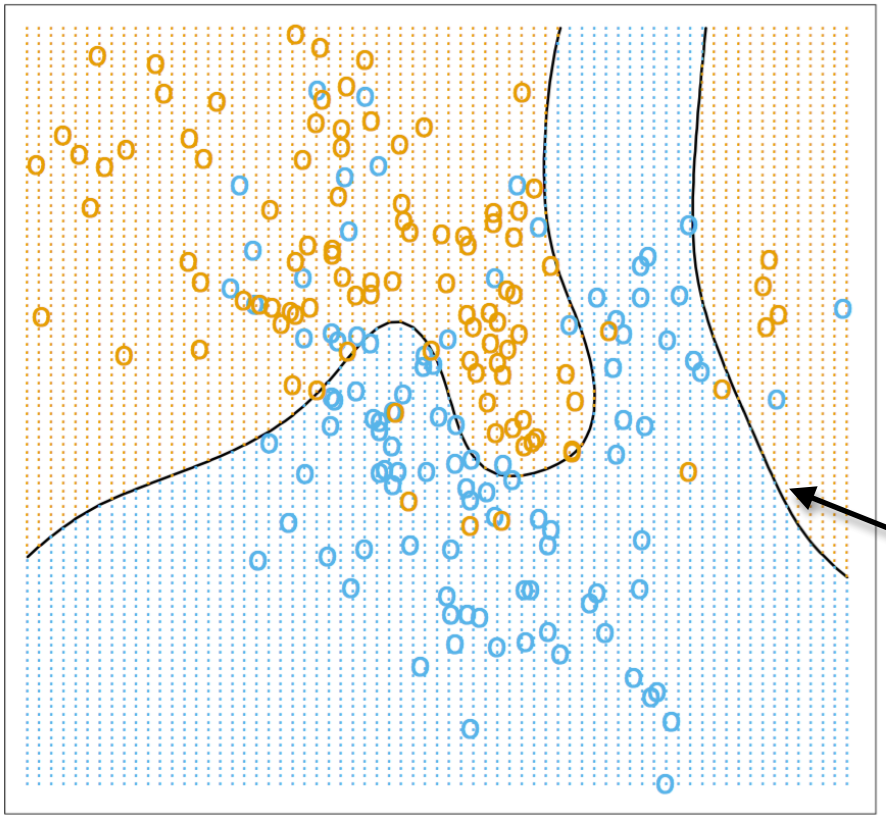
$$\sum_{i=1}^{n} \max\{0, 1 - y_i(b + x_i^T w)\} + \lambda ||w||_2^2 \quad = \sum_{i=1}^{n} \ell_i(w, b)$$

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

©2018 Kevin Jamieson

Nearest Neighbor

Some data, Bayes Classifier



Training data:

True label: +1

True label: -1

Optimal "Bayes" classifier:

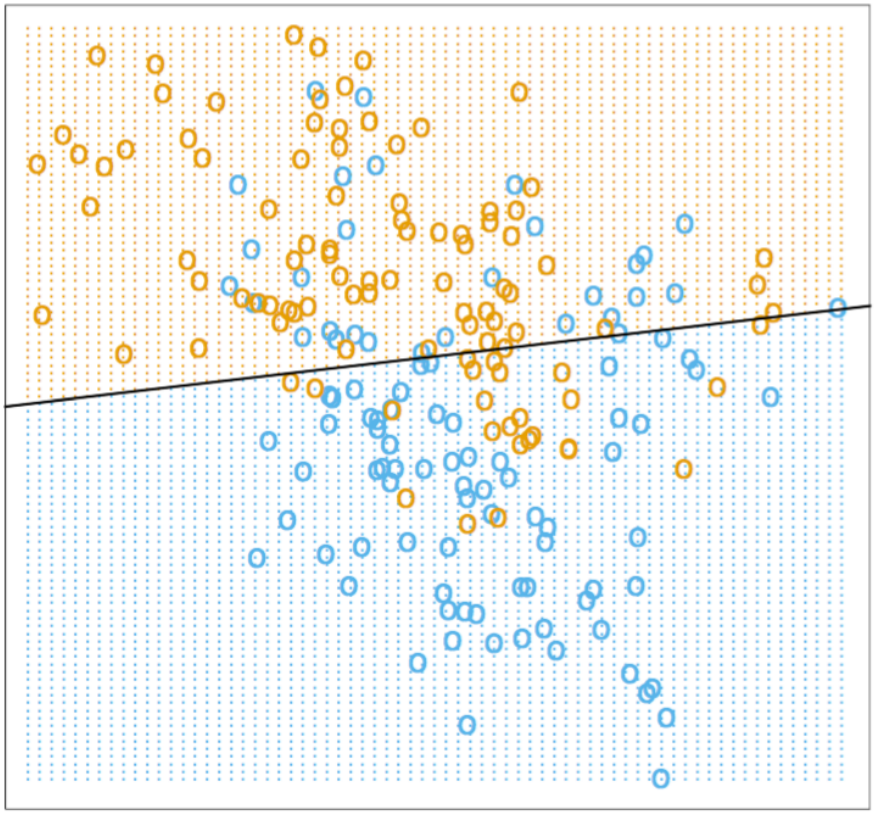
$$\mathbb{P}(Y = 1|X = x) = \frac{1}{2}$$

Predicted label: +1

Predicted label: -1

Figures stolen from Hastie et al

Linear Decision Boundary



Training data:

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

Learned:

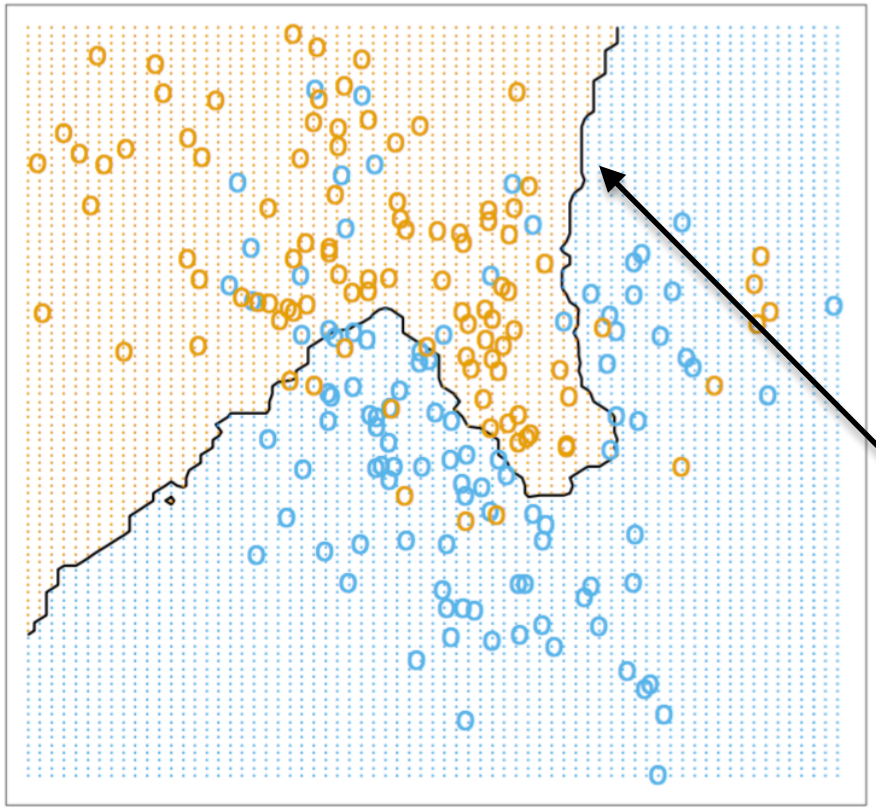
Linear Decision boundary

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

- 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:

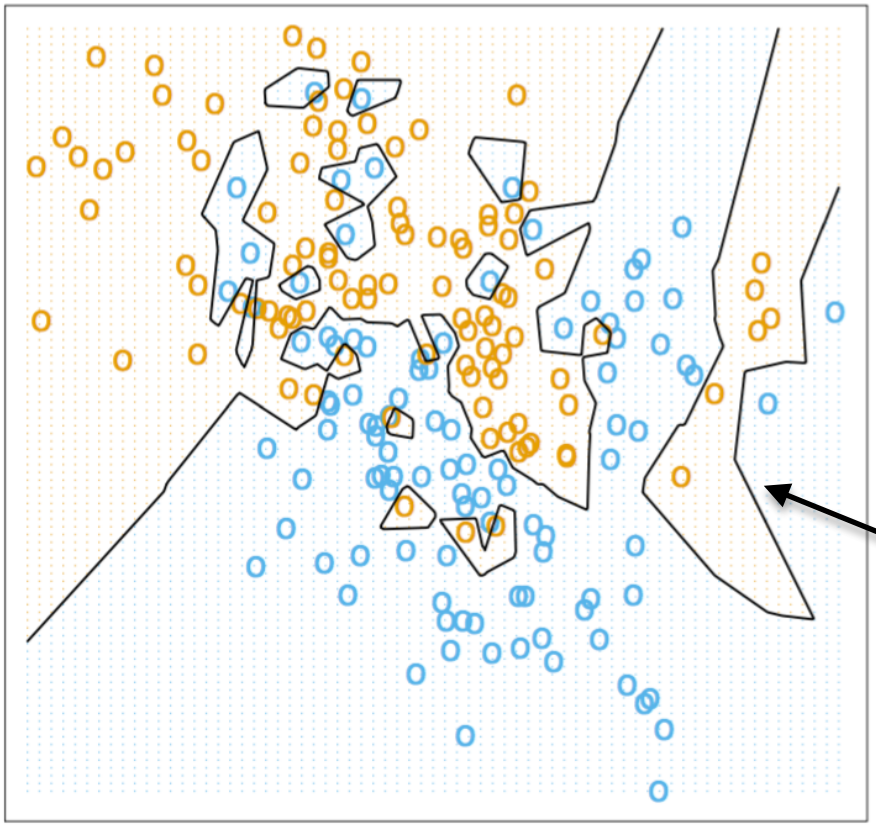
15 nearest neighbor decision boundary (majority vote)

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:

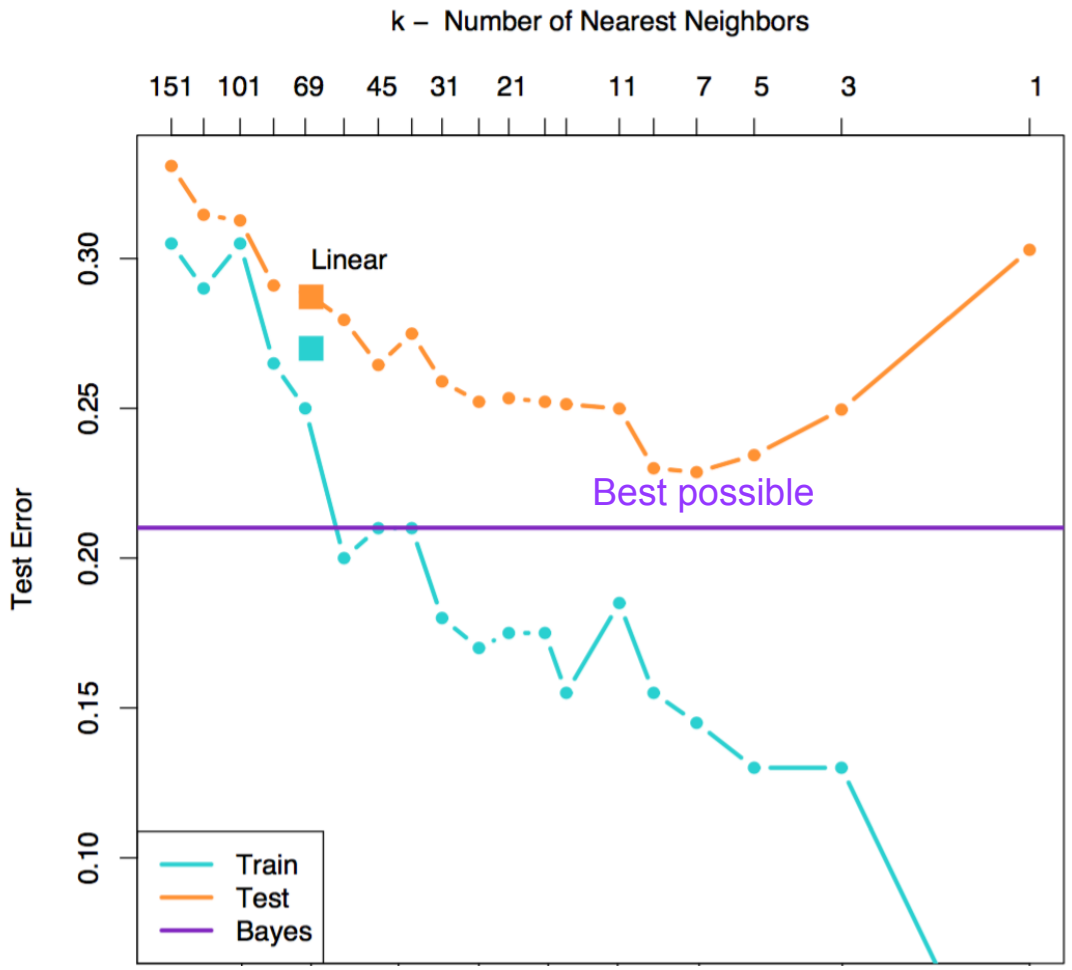
1 nearest neighbor decision boundary (majority vote)

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?

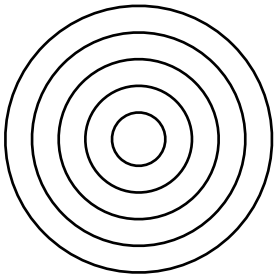
Bias:

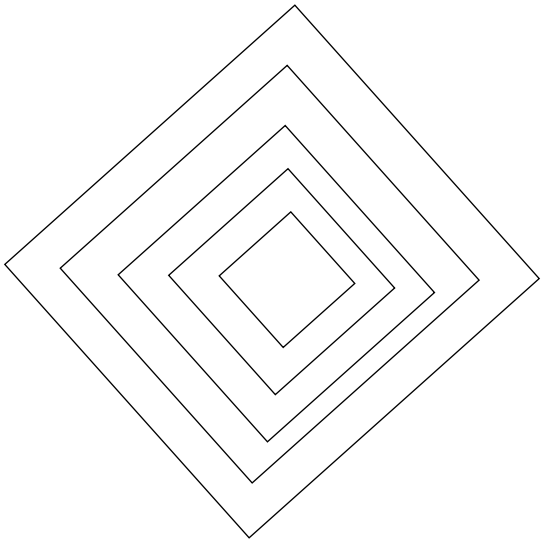
Variance:

Figures stolen from Hastie et al

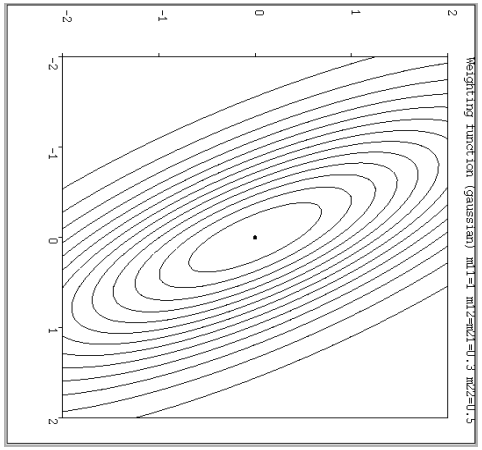
Notable distance metrics (and their level sets)

L_2 norm

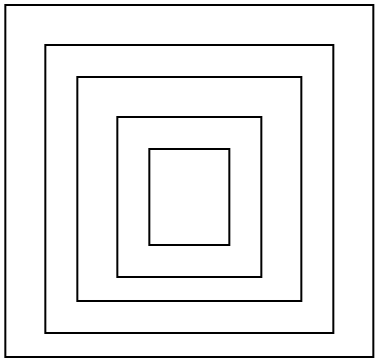




L_1 norm (taxi-cab)



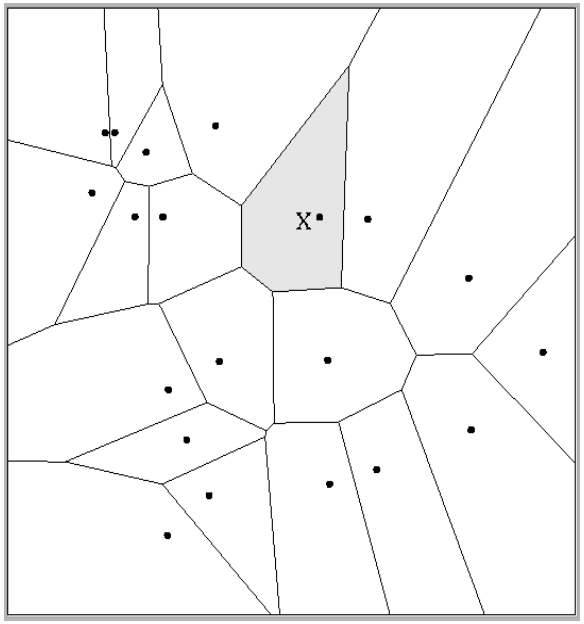
Mahalanobis



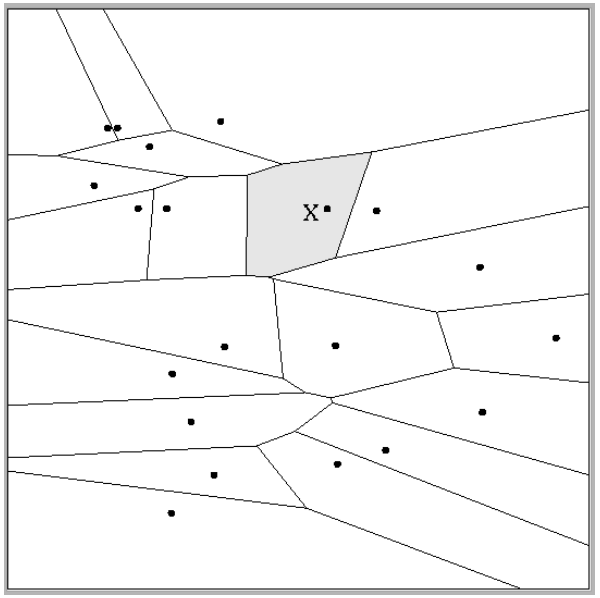
L-infinity *(max) norm*

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.

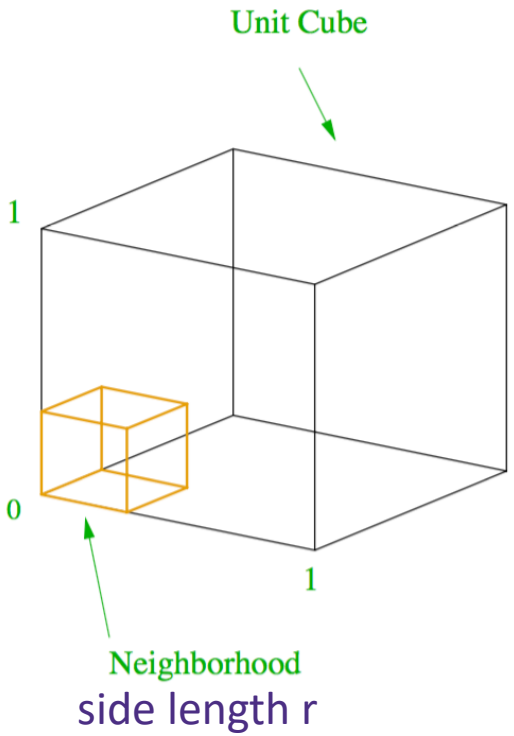
As $x_a \to x_b$ we have $\mathbb{P}(Y_a = 1|X_a = x_a) \to \mathbb{P}(Y_b = 1|X_b = x_b)$

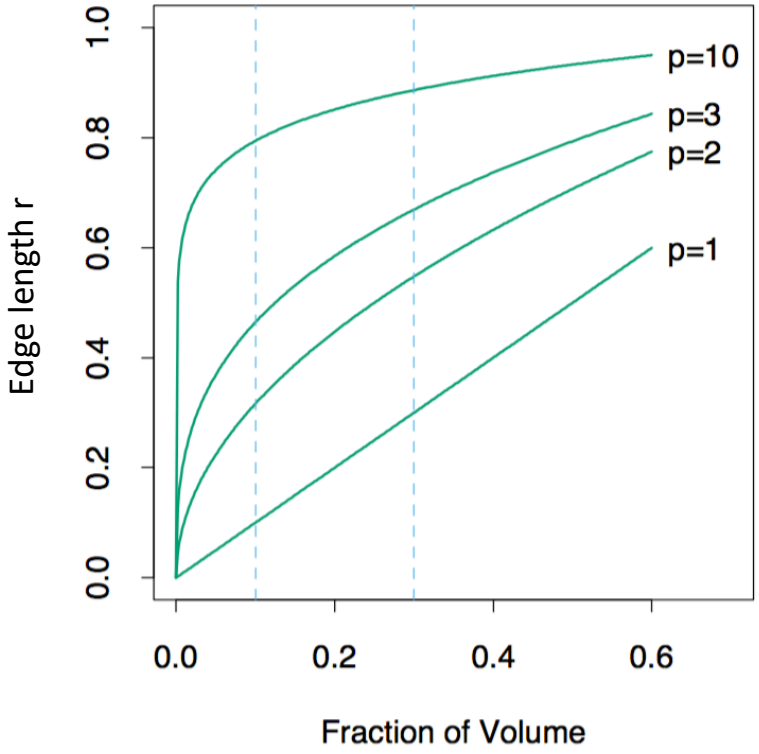
If $p_* = \max_{y=0,1} \mathbb{P}(Y_b = y|X_b = x_b)$ then the Bayes Error $= 1 - p_*$

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

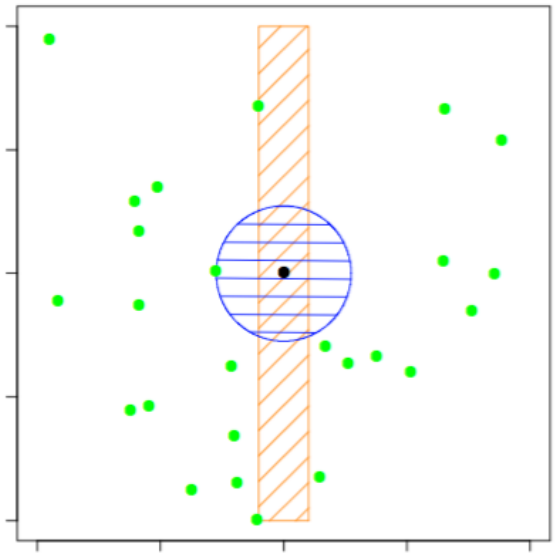


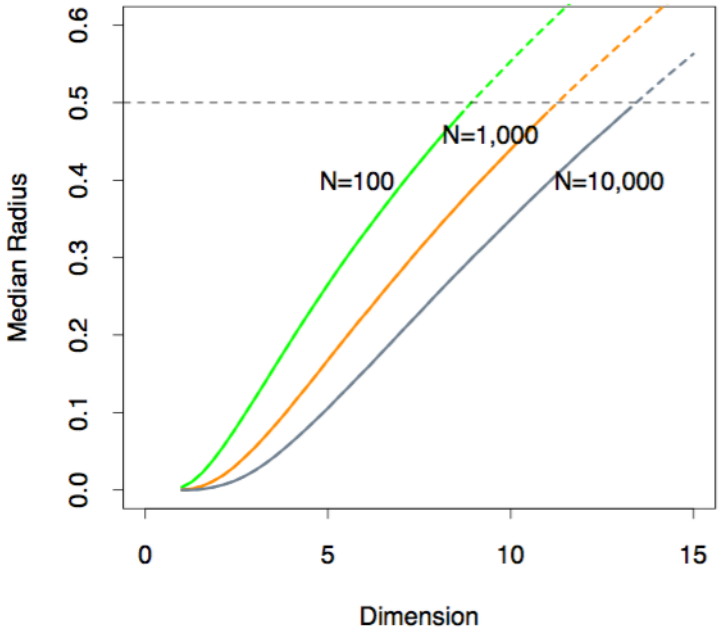


X is uniformly distributed over $[0,1]^p$. What is $\mathbb{P}(X \in [0,r]^p)$?

Curse of dimensionality Ex. 2

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

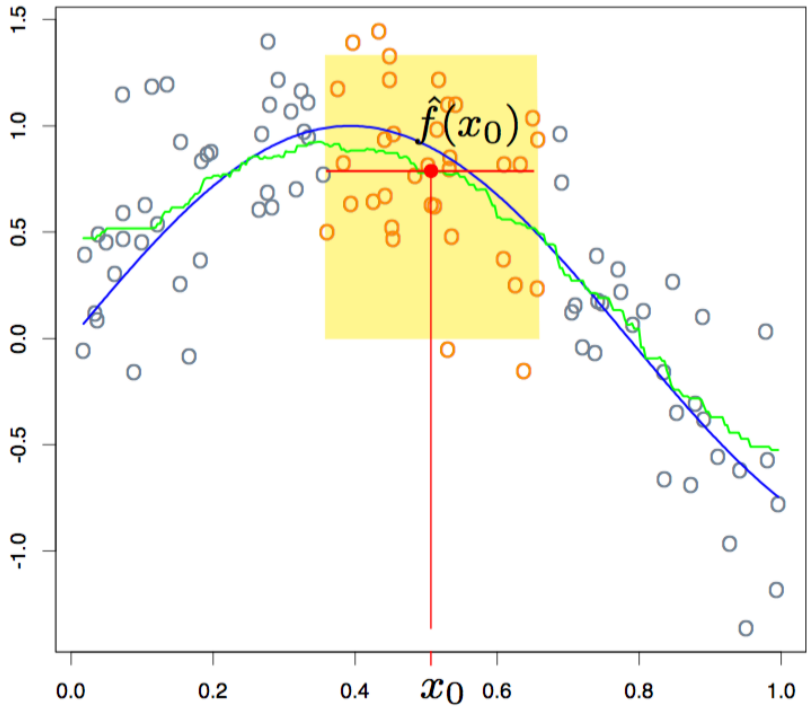




What is the median distance from a point at origin to its 1NN?

Nearest neighbor regression

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

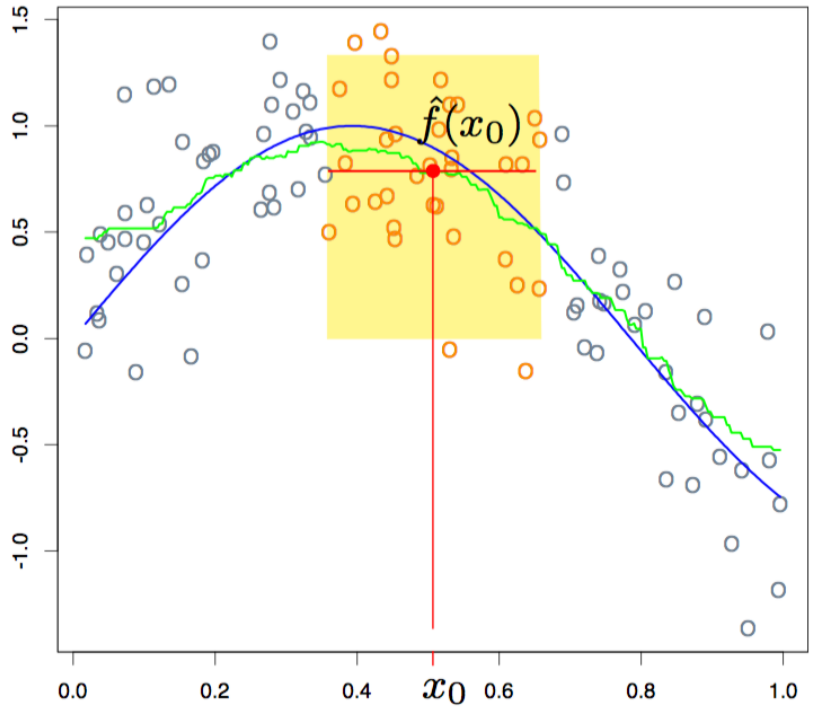


$$\mathcal{N}_k(x_0) = k\text{-nearest neighbors of } x_0$$

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

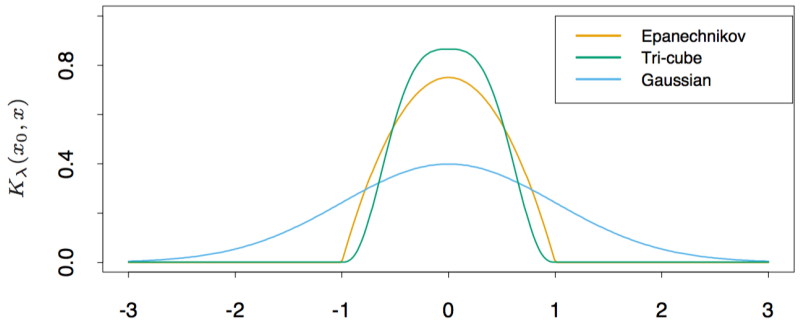
Nearest neighbor regression

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



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

Kernel smoothing: $K(x, y)$



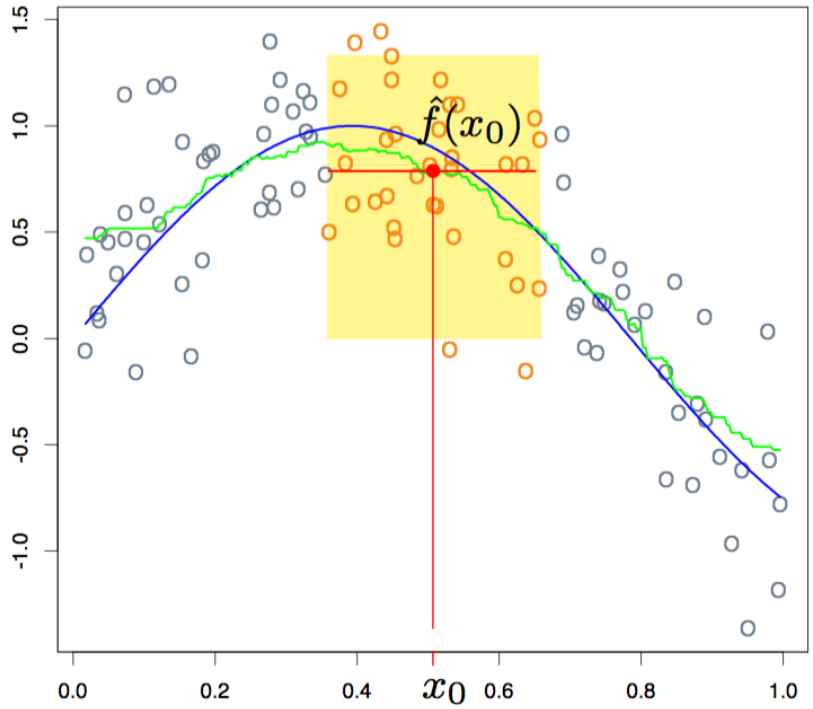
$\mathcal{N}_k(x_0) = k$-nearest neighbors of x_0

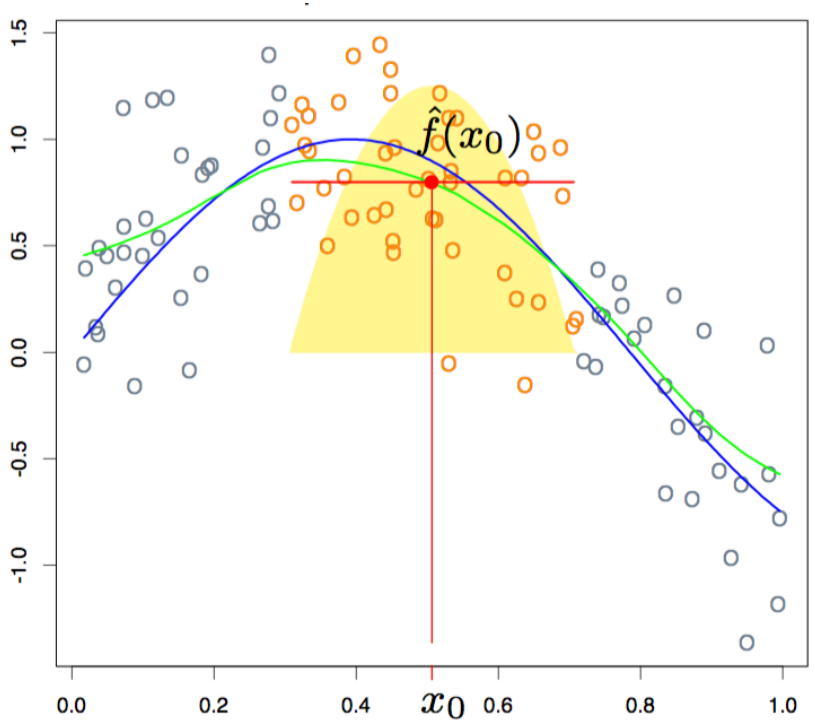
$$\hat{f}(x_0) = \sum_{x_i \in \mathcal{N}_k(x_0)} \frac{1}{k} y_i$$

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

Nearest neighbor regression

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





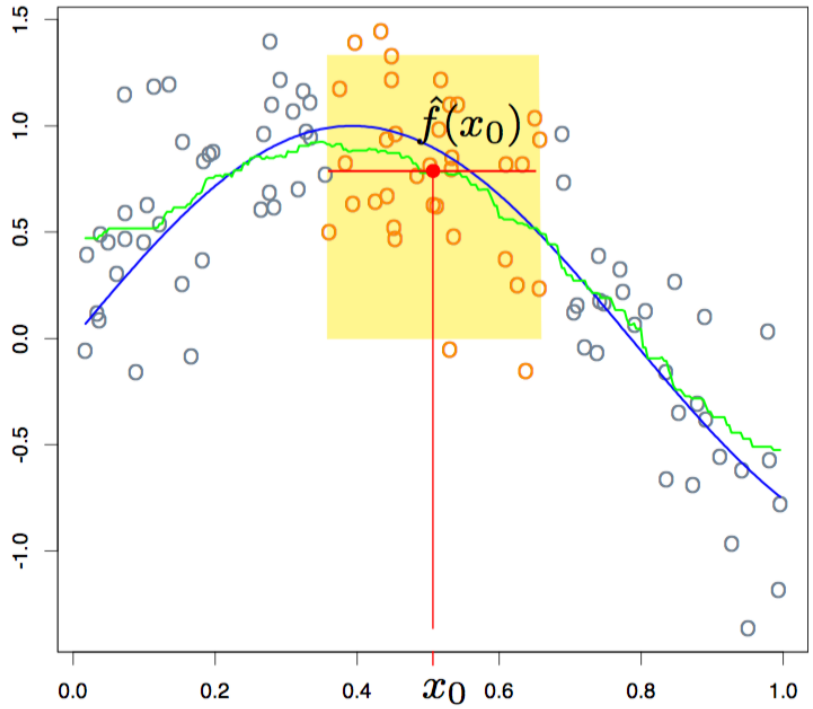
$$\mathcal{N}_k(x_0) = k\text{-nearest neighbors of } x_0$$

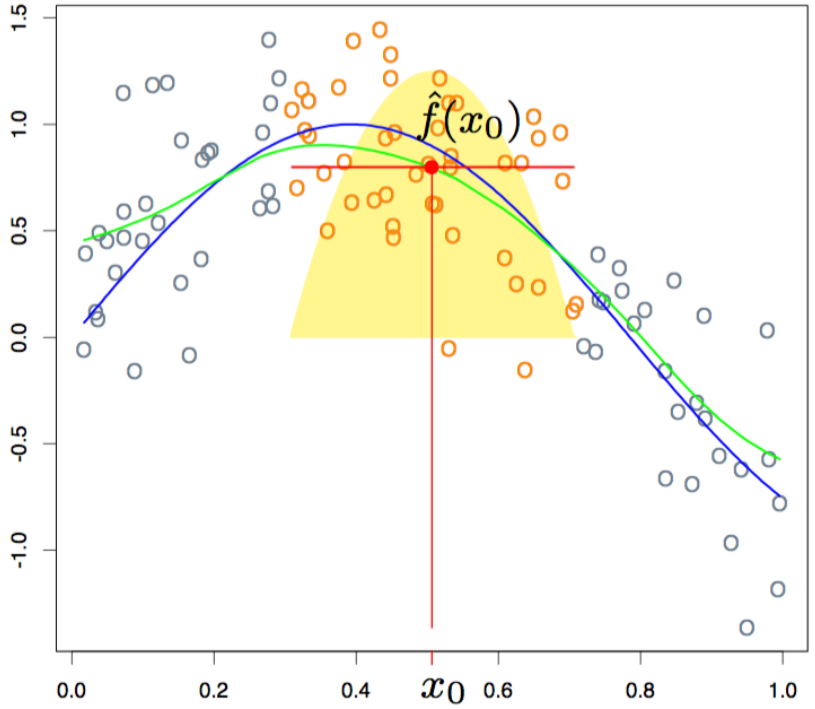
$$\widehat{f}(x_0) = \sum_{x_i \in \mathcal{N}_k(x_0)} \frac{1}{k}\, y_i$$

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

Nearest neighbor regression

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





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

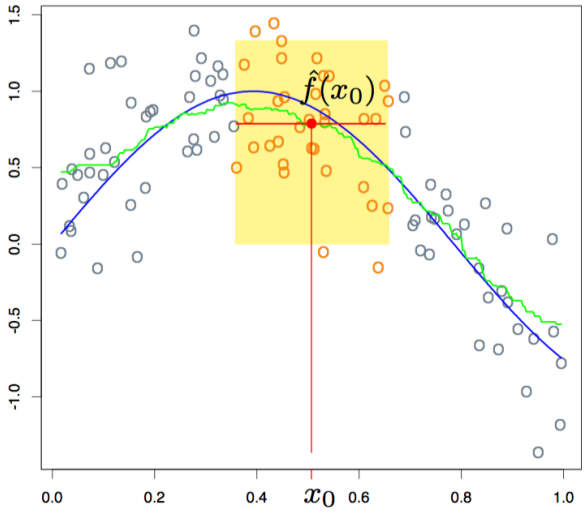
$$\widehat{f}(x_0) = \sum_{x_i \in \mathcal{N}_k(x_0)} \frac{1}{k}\, y_i$$

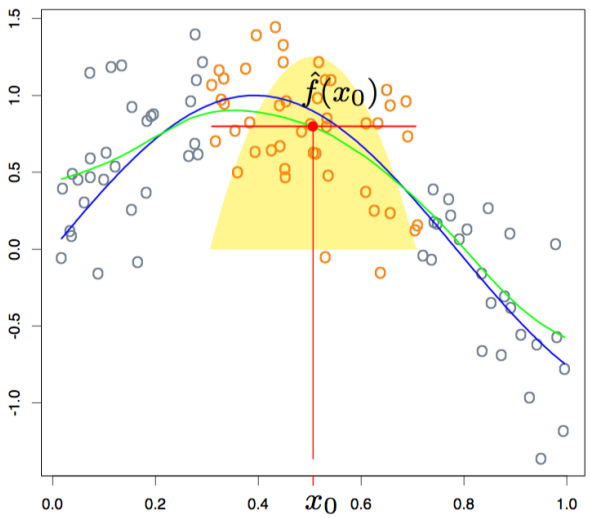
Why just average them?

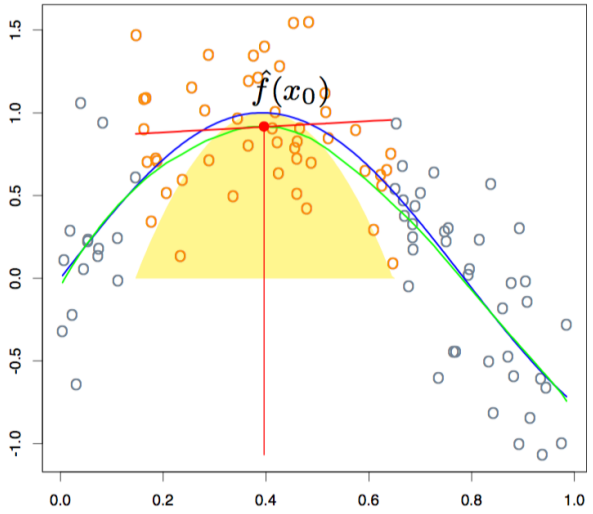
$$\widehat{f}(x_0) = \frac{\sum_{i=1}^n K(x_0, x_i) y_i}{\sum_{i=1}^n K(x_0, x_i)}$$

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.

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: $\widehat{\theta} = t(\mathcal{D})$

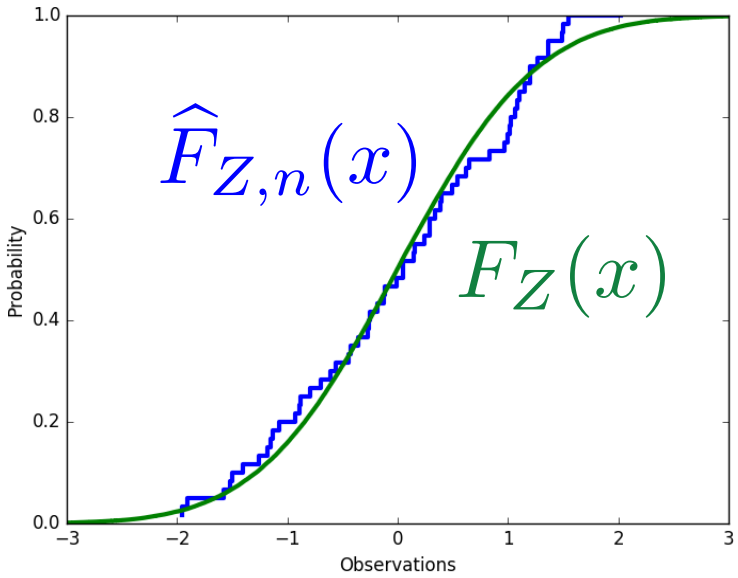
What is the distribution of $\widehat{\theta} = t(\mathcal{D})$

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: $\theta^{*b} = t(\mathcal{D}^{*b})$

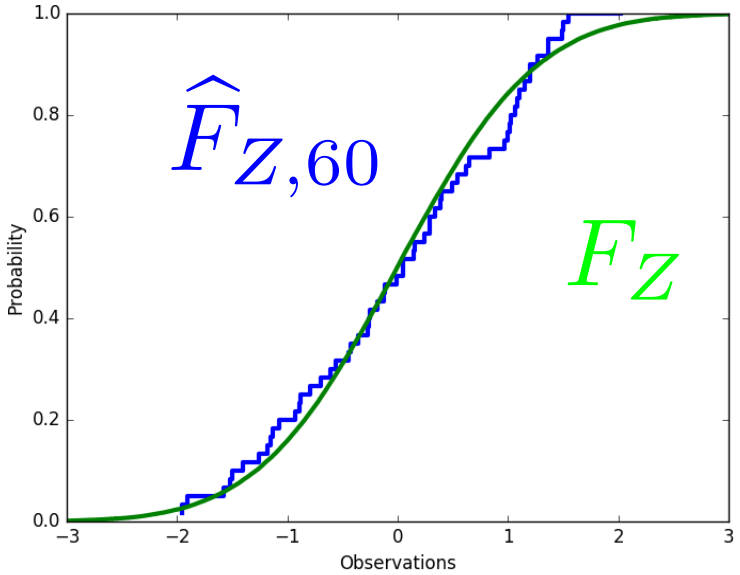
Bootstrap: basic idea

Given dataset drawn iid samples with CDF F_Z:

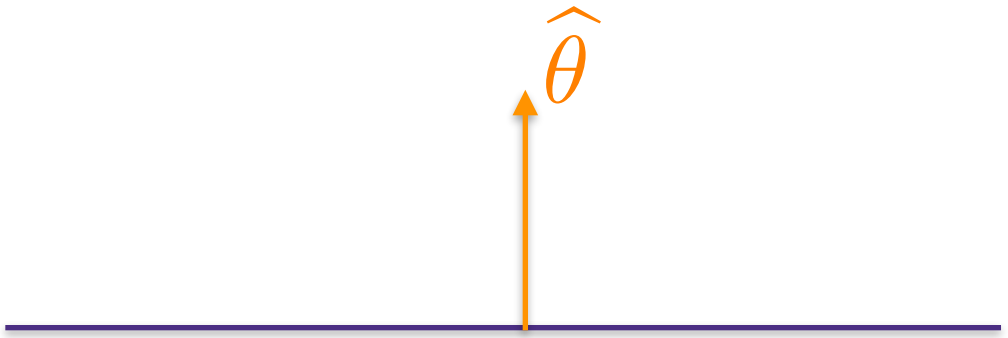
$$\mathcal{D} = \{z_1, \ldots, z_n\} \overset{i.i.d.}{\sim} F_Z$$

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

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



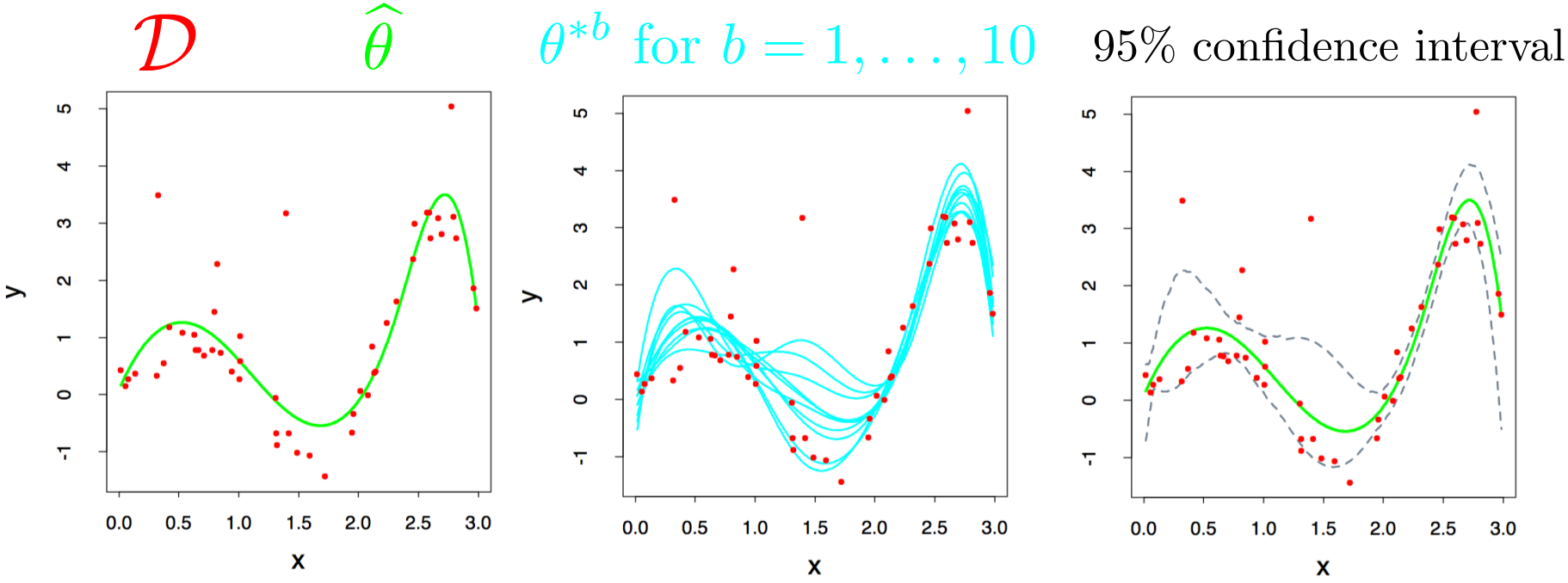
$$|\widehat{F}_{Z,n}(x) - F_Z(x)| \overset{n\to\infty}{\to} 0 \quad \text{a.s.}$$



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.