

Pattern Recognition
Lecture 02-2
Random Forest &
Linear Classification

Prof. Jongwon Choi
Chung-Ang University
Fall 2022

Standardizing Features

- Features with Different Scales:

Egg (#)	Milk (mL)	Fish (g)	Pasta (cups)
0	250	0	1
1	250	200	1
0	0	0	0.5
2	250	150	0

- Should we convert to some standard ‘unit’?

- It matters for k-nearest neighbours:

- “Distance” will be affected more by large features than small features

- It matters for regularized least squares:

- Penalizing (w) means different things if features ‘ j ’ are on different scales

Standardizing Features

- **It is common to standardize continuous features**

- For each feature:

- 1. Compute mean and standard deviation: $\mu_j = \frac{1}{n} \sum_{i=1}^n x_{ij}$, $\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_{ij} - \mu_j)^2}$

- 2. Subtract mean and divide by standard deviation: $x_{ij} \rightarrow \frac{x_{ij} - \mu_j}{\sigma_j}$

- Now changes in w_j have similar effect for any feature 'j'

- **In the test phase,**

- Use mean and standard deviation of training data (not test data)

- Since the distribution of training and test data can differ


- Furthermore, the entire test data cannot be obtained in an online manner.

Standardizing Labels

- **In regression, we sometimes standardize the targets y_i**
 - Puts targets on the same standard scale as standardized features:
 - $y_i \rightarrow \frac{y_i - \mu_y}{\sigma_y}$
- With the standardized label, $w=0$ predicts the average of y_i
 - High regularization makes us predict closer to the average value
- The test data should be standardized by the training stats.
- Other common transformations of y_i are logarithm/exponent:
 - $\log(y_i)$ or $\exp(y_i)$

Regression?

- The supervised learning classification:

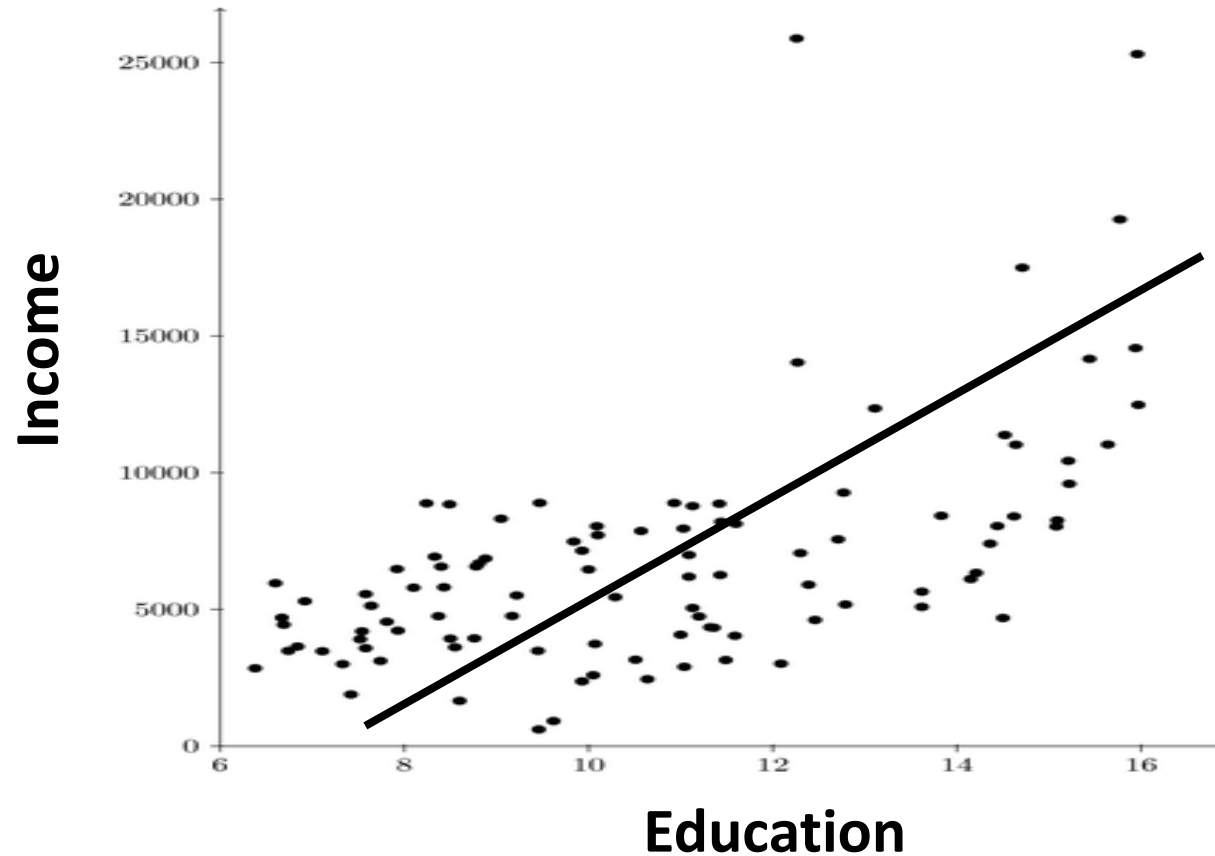
Peanut	Fish	Meat	Wheat	Water	Egg	Milk		Dog eats?
0	0.1	0	0	0.1	0.1	0		Not eat
0.3	0.2	0.9	0	0.9	0.8	0		Eat
0	0.8	0.3	0.5	0.4	0.1	0.2		Not eat
0	0	0.8	0.2	0	0	0.1		Eat
0.5	0.1	0.2	0.9	0.2	0	0.3		Not eat

- where we assume y_i was discrete: $y_i \in \{eat, not\ eat\}$
- Then, how should we handle the learning when
 - y_i is continuous? i.e. $y_i = 1.32\ kg$

Regression?

- Examples

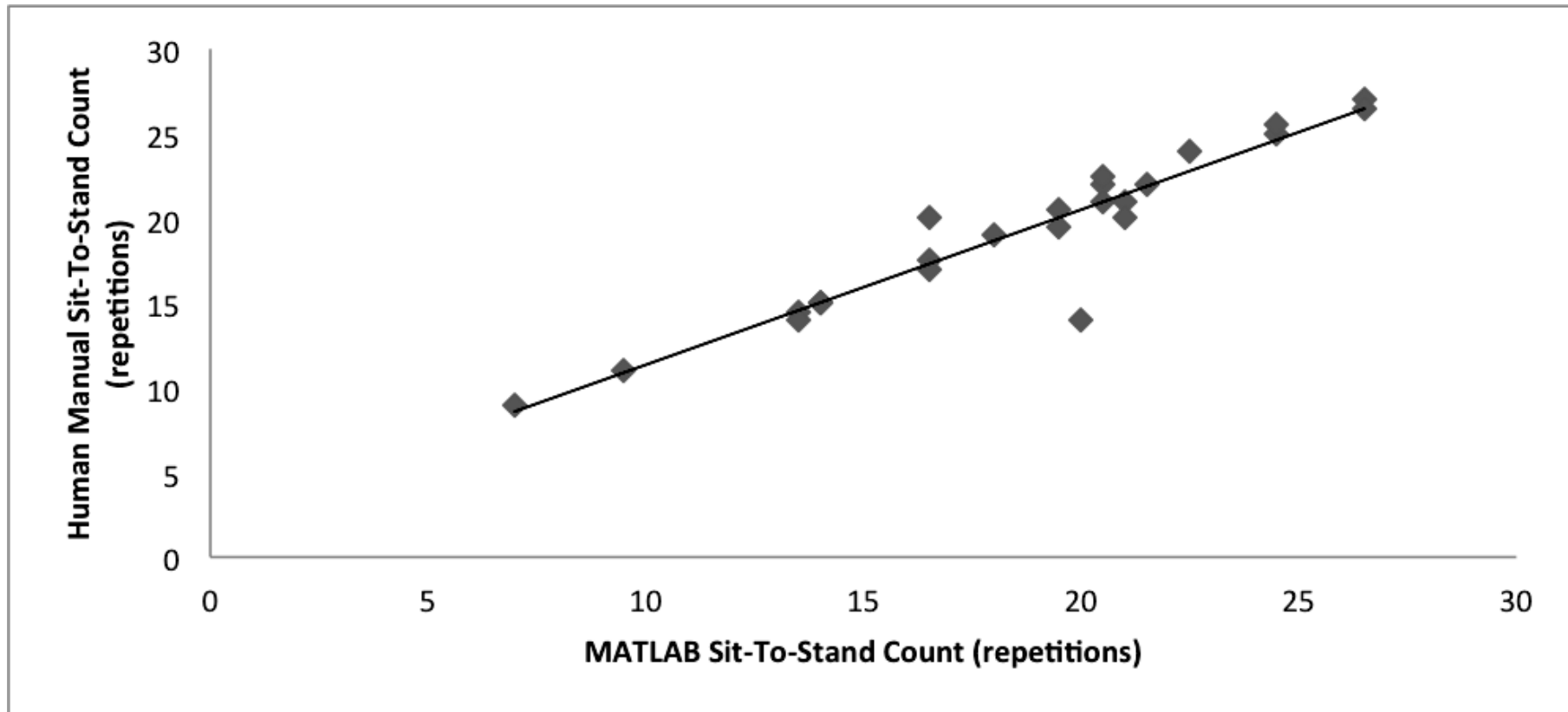
- Does the income change with the length of education ?



Regression?

- Examples

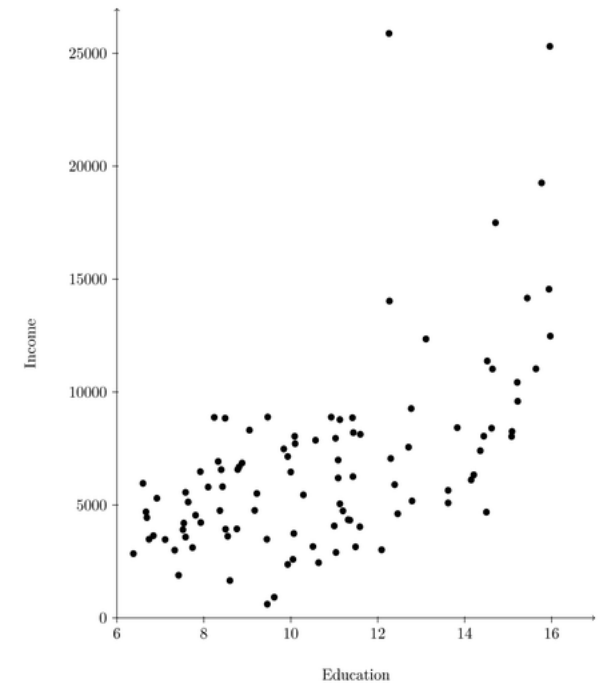
- Does the counts from algorithm change with the real counts?



Regression?

- **IMPORTANT!!**

- We are doing just predict a value of one variable (y_i)
- The supervised learning does not give any CAUSALITY!
 - GOOD - Longer education is correlated with higher income
 - GOOD - Longer education helps predict higher income
 - BAD - Longer education leads to higher income
 - The prediction does not mean the causality & consequence
 - There can be lots of potential reasons for this correlation!
 - In other words, even when higher income leads to longer education, it is possible to predict the income from the length of education.



Regression!

- Definition : The models handling numerical labels (y_i)
- 2 ways
 - 1. Distretize the numerical labels
 - We can utilize the classification methods for regression
 - Coarse discretization loses resolution & fine discretization needs lots of data
 - Ex. regression trees, probabilistic models, non-parametric models
 - 2. Output the numerical labels
 - The best example - Linear regression based on squared error
 - Interpretable & building blocks for complex methods! (i.e. Deep Network)

Linear Regression – 1 Dimension

- Assume that we only have 1 feature ($d=1$):
 - ex. x_i is the length of education and y_i is income
- Linear regression makes predictions \hat{y}_i using a linear function of x_i :

$$\hat{y}_i = wx_i$$

- The parameter 'w' is the weight or regression coefficient of x_i
- As x_i changes, slope 'w' affects the rate that \hat{y}_i increases/decreases:
 - Positive 'w': \hat{y}_i increases as x_i increases
 - Negative 'w': \hat{y}_i decreases as x_i increases

Least Squares Objective

- Our linear model is given by:

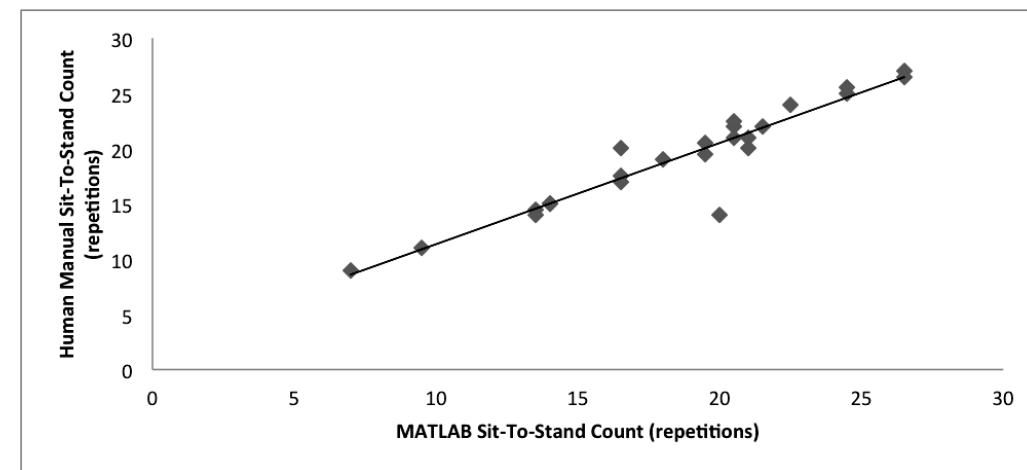
$$\hat{y}_i = wx_i$$

- So we make predictions for a new example by using:

$$\hat{y}_i = w\tilde{x}_i$$

- But we can't use the same error as before:

- It is unlikely to find a line where $\hat{y}_i = y_i$ exactly for many points (i.e. noise)
- “Best” model may have $|\hat{y}_i - y_i|$ is small but not exactly 0.



Least Squares Objective

- Instead of “exact y_i ”, we evaluate the size of error in prediction
- Classic way is setting slope ‘ w ’ to minimize the sum of squared errors:

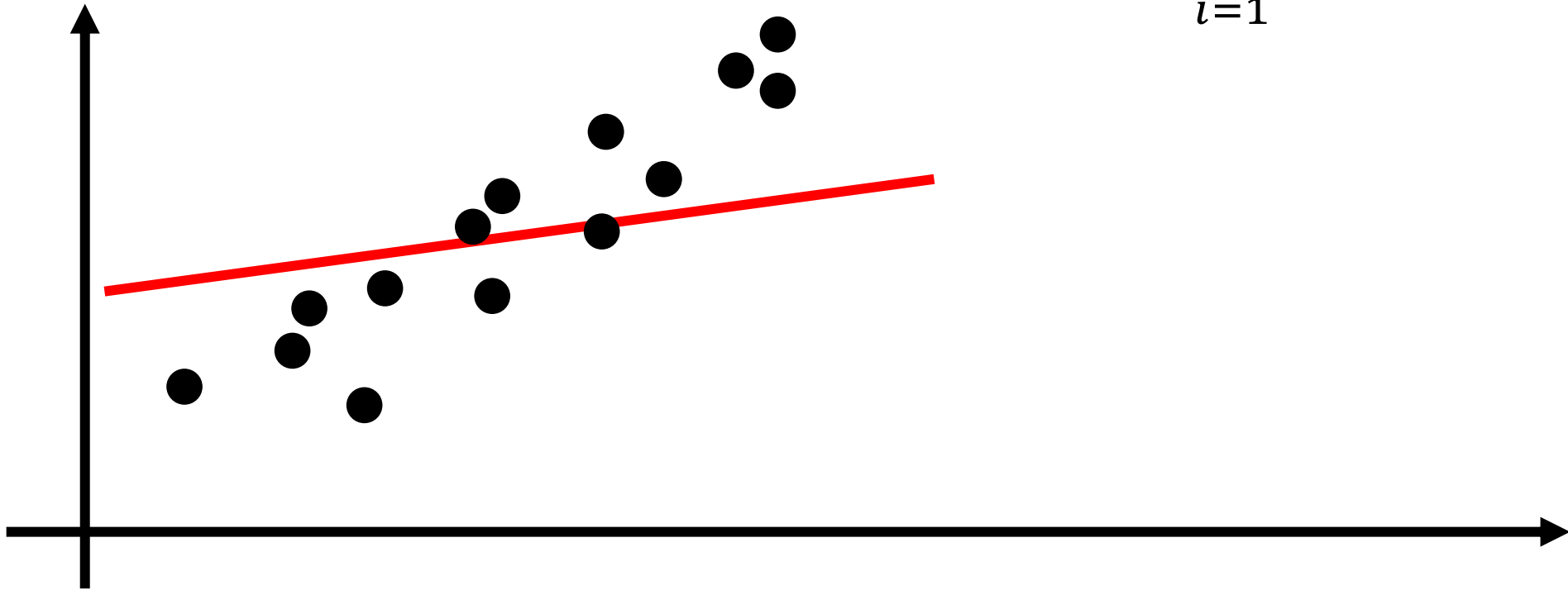
$$f(w) = \sum_{i=1}^n (wx_i - y_i)^2$$

- A probabilistic interpretation is coming later in this course!
- But usually, it is done because it is easy to minimize.

Least Squares Objective

- Example – Large $f(w)$

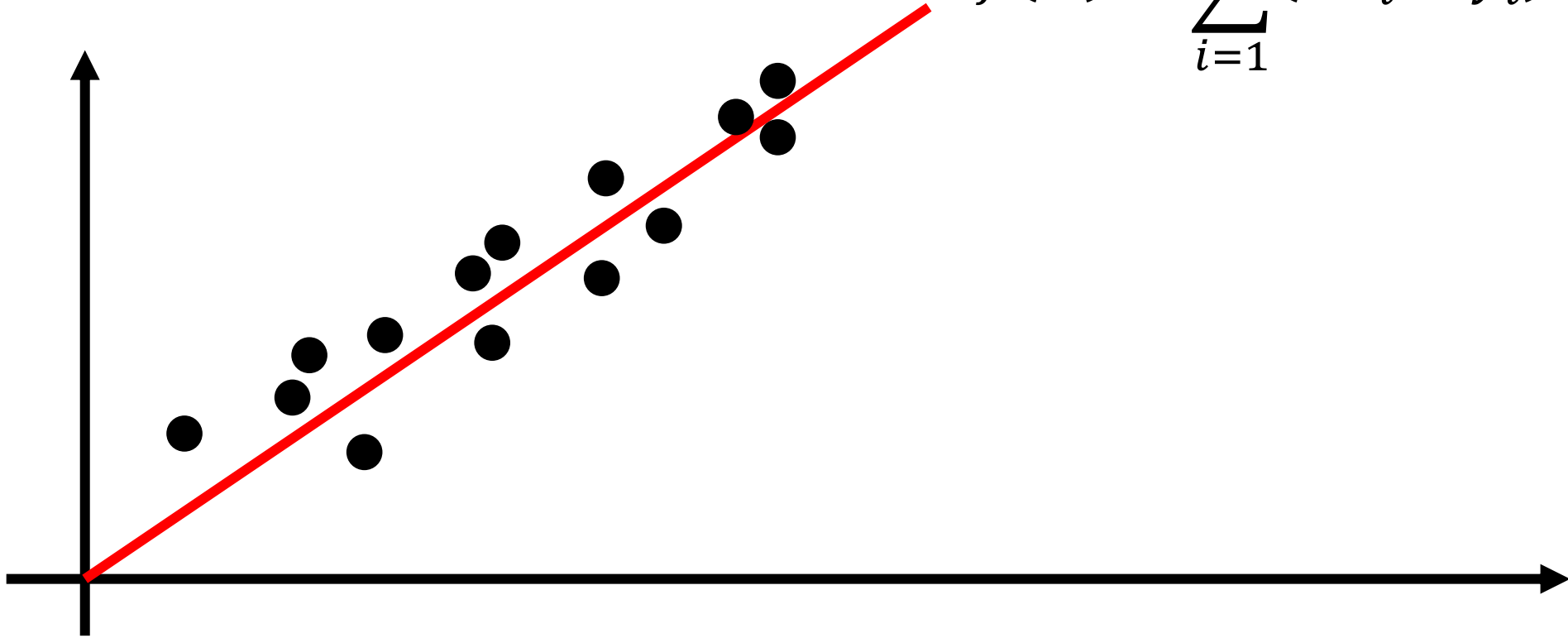
$$f(w) = \sum_{i=1}^n (wx_i - y_i)^2$$



Least Squares Objective

- Example – Small $f(w)$

$$f(w) = \sum_{i=1}^n (wx_i - y_i)^2$$



Finding Least Squares Solution

- Not change the solution!
 - Multiply 'f' by any positive constant
 - Add some constants to 'f'

$$f'(w) = C_1 \sum_{i=1}^n (wx_i - y_i)^2 + C_2$$

- Finding 'w' that minimizes sum of squared errors:

$$\begin{aligned} f(w) &= \frac{1}{2} \sum_{i=1}^n (wx_i - y_i)^2 = \frac{1}{2} \sum_{i=1}^n [w^2 x_i^2 - 2wx_i y_i + y_i^2] \\ &= \frac{w^2}{2} \sum_{i=1}^n x_i^2 - w \sum_{i=1}^n x_i y_i + \frac{1}{2} \sum_{i=1}^n y_i^2 = \frac{w^2}{2} a - wb + c \end{aligned}$$

$$f'(w) = wa - b = 0$$

$$w = \frac{\sum_{i=1}^n x_i y_i}{\sum_{i=1}^n x_i^2}$$

Multiple Dimension Linear Function

- A simple way is with a d-dimensional linear model
 - $\hat{y}_i = w_1x_{i1} + w_2x_{i2} + w_3x_{i3} + \cdots + w_dx_{id}$
 - In words, our model is that the output is a weighted sum of the inputs
- We can re-write this in summation notation:
 - $\hat{y}_i = \sum_{j=1}^d w_jx_{ij}$
- We can also re-write this in vector notation: (inner product)
 - $\hat{y}_i = \mathbf{w}^T \mathbf{x}_i$
 - In this course, a vector is a column vector

Least Squares in d-Dimensions

- The linear least squares model in d-dimensions minimizes:

$$f(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^n (\mathbf{w}^T \mathbf{x}_i - y_i)^2$$

- Least Squares Partial Derivatives for 1 sample

$$f(w_1, w_2, \dots, w_d) = \frac{1}{2} \left(\sum_{j=1}^d w_j x_{ij} \right)^2 - \left(\sum_{j=1}^d w_j x_{ij} \right) y_i + \frac{1}{2} y_i^2$$

$$\frac{\partial}{\partial w_k} f(w_1, w_2, \dots, w_d) = \left(\sum_{j=1}^d w_j x_{ij} \right) x_{ik} - y_i x_{ik} = (\mathbf{w}^T \mathbf{x}_i - y_i) x_{ik}$$

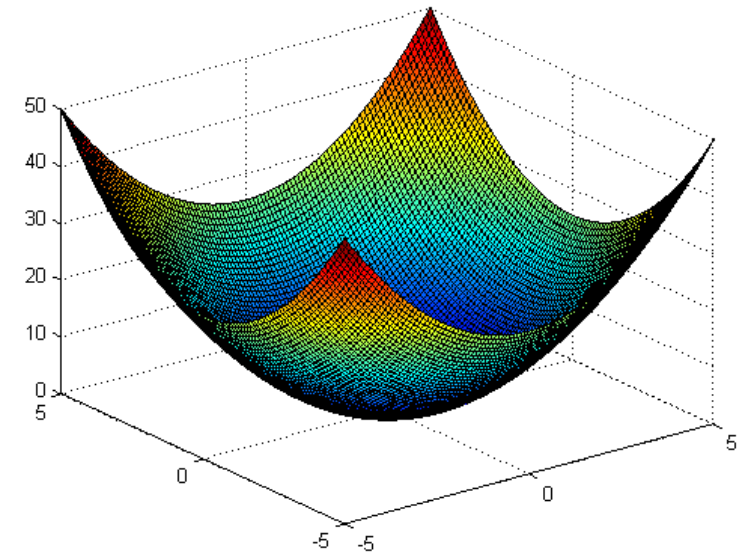
Least Squares in d-Dimensions

- Least Squares Partial Derivatives for all samples

$$\frac{\partial}{\partial w_k} f(w_1, w_2, \dots, w_d) = \sum_{i=1}^n (\mathbf{w}^T \mathbf{x}_i - y_i) x_{ik}$$

- Unfortunately, the partial derivative for w_j depends on all $\{w_1, w_2, \dots, w_d\}$
 - Thus, we can't just set equal to 0 and solve for w_j
 - **We need to find 'w' where the gradient vector equals the zero vector!**

$$\nabla f(w_1, w_2, \dots, w_d) = \left[\frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}, \frac{\partial f}{\partial w_3}, \dots, \frac{\partial f}{\partial w_d} \right]^T$$



Linear and Quadratic Gradients

$$f(w) = aw^2 \quad \Rightarrow \quad \frac{\partial}{\partial w} f(w) = 2aw$$

$$f(\mathbf{w}) = \mathbf{w}^T \mathbf{A} \mathbf{w} \quad \Rightarrow \quad \nabla f(\mathbf{w}) = \mathbf{A} \mathbf{w}$$

If A is symmetric

$$g(w) = bw \quad \Rightarrow \quad \frac{\partial}{\partial w} g(w) = b$$

$$g(\mathbf{w}) = \mathbf{w}^T \mathbf{b} \quad \Rightarrow \quad \nabla g(\mathbf{w}) = \mathbf{b}$$

$$h(w) = c \quad \Rightarrow \quad \frac{\partial}{\partial w} h(w) = 0$$

$$h(\mathbf{w}) = \mathbf{c} \quad \Rightarrow \quad \nabla h(\mathbf{w}) = 0$$

Linear and Quadratic Gradients

- We can re-write the d-dimensional quadratic:

- $f(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^n (\mathbf{w}^T \mathbf{x}_i - y_i)^2 = \frac{1}{2} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 = \frac{1}{2} \mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w} - \mathbf{w}^T \mathbf{X}^T \mathbf{y} + \frac{1}{2} \mathbf{y}^T \mathbf{y}$

- $f(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{A} \mathbf{w} + \mathbf{w}^T \mathbf{b} + \mathbf{c}$

- Thus, the gradient is given by:

- $\nabla f(\mathbf{w}) = \mathbf{A} \mathbf{w} - \mathbf{b} = \mathbf{X}^T \mathbf{X} \mathbf{w} - \mathbf{X}^T \mathbf{y}$

- **Normal equations:**

- $\mathbf{X}^T \mathbf{X} \mathbf{w} = \mathbf{X}^T \mathbf{y}$ (in the form of $\mathbf{A} \mathbf{x} = \mathbf{b}$)

- When $\mathbf{X}^T \mathbf{X}$ is invertible, $\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$

Least Squares Cost

- When $\mathbf{X}^T \mathbf{X}$ is invertible, $\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$
 - $\mathbf{X}^T \mathbf{y} : O(nd)$
 - $\mathbf{X}^T \mathbf{X} : O(nd^2)$
 - $(\mathbf{X}^T \mathbf{X})^{-1} : O(d^3)$
 - $(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} : O(d^2)$
- Entire cost = $O(nd + nd^2 + d^3 + d^2) = O(nd^2 + d^3)$

Least Squares Issues

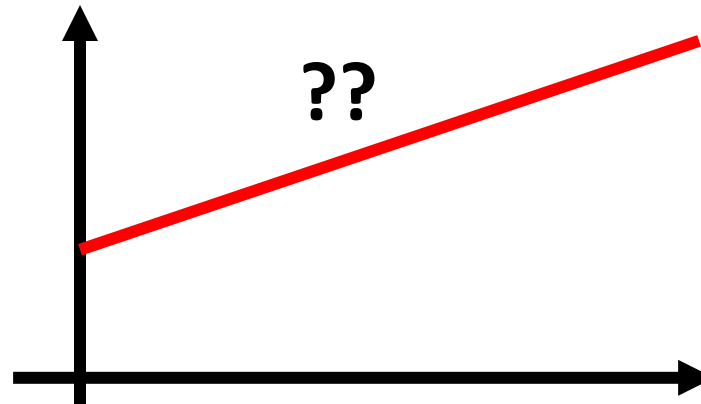
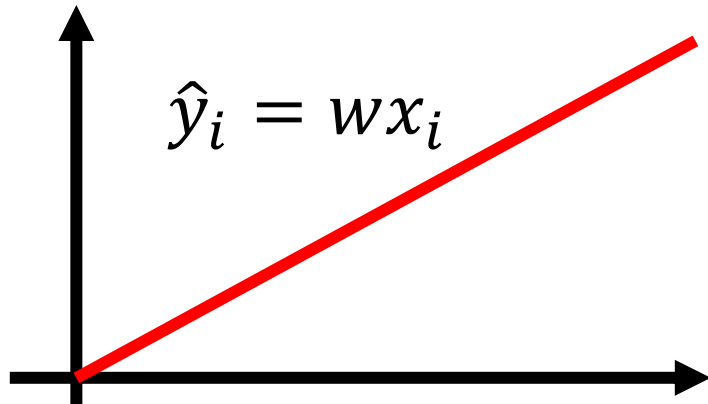
- Issues with least squares model:
 - Solution might not be unique
 - It is sensitive to outliers
 - It always uses all features
 - Data can be so big we can't store $\mathbf{X}^T\mathbf{X}$
 - Or you can't afford the $O(nd^2 + d^3)$ cost
 - It might predict outside range of y_i values
 - It assumes a linear relationship between x_i and y_i

Adding a Bias Variable

- Make a new matrix “Z” with an extra feature that is always “1”

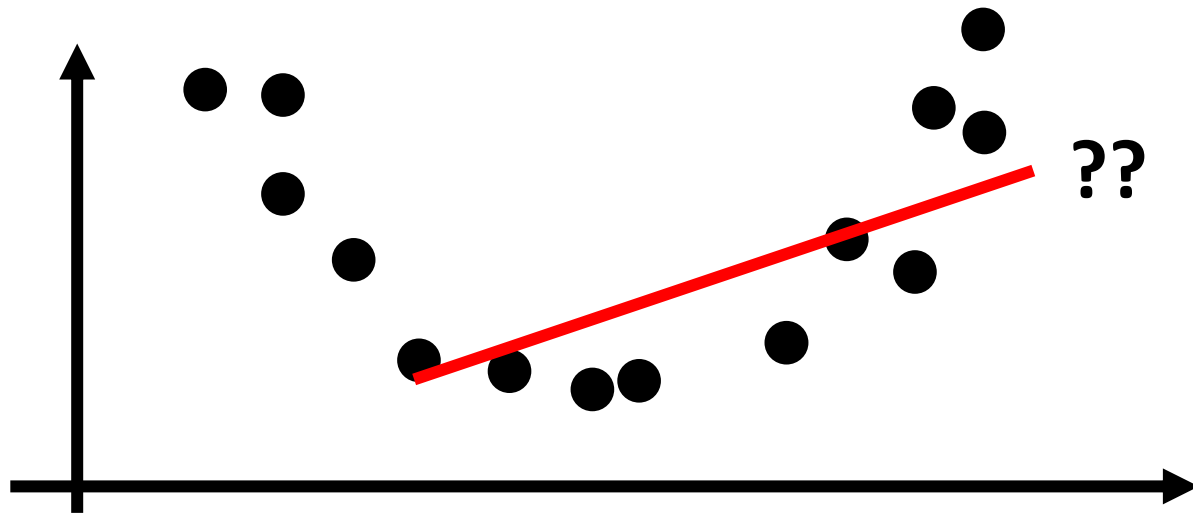
- $\mathbf{X} = \begin{bmatrix} -0.1 \\ 0.3 \\ 0.2 \end{bmatrix} \Rightarrow \mathbf{Z} = \begin{bmatrix} 1 & -0.1 \\ 1 & 0.3 \\ 1 & 0.2 \end{bmatrix}$

- $\hat{y}_i = v_1 z_{i1} + v_2 z_{i2} = w_0 + w_1 x_{i1}$



Limitation of Linear Models

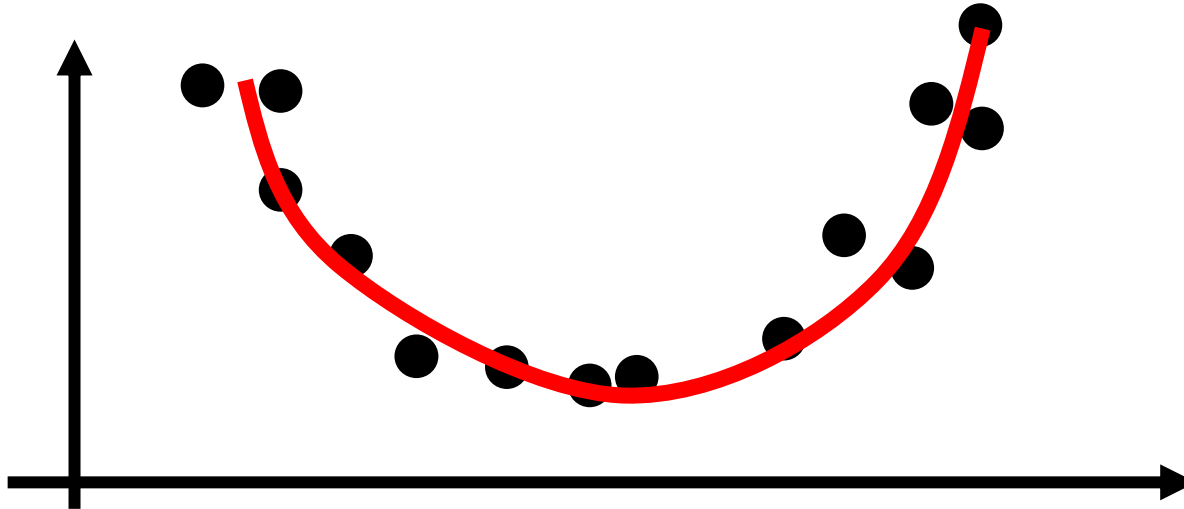
- On many situations, y_i is not a linear function of x_i



- Can we use least square to find non-linear models??

Non-linear Feature Transforms

- Can we use linear least squares to fit a quadratic model?
 - Yes! By adding a non-linear feature as:
 - $\hat{y}_i = w_0 + w_1x_i + w_2x_i^2$
 - It's a linear function of w , but a quadratic function of x_i
 - $\hat{y}_i = \mathbf{v}^T \mathbf{z}_i = v_1z_{i1} + v_2z_{i2} + v_3z_{i3}$



General Polynomial Features (d=1)

- We can have a polynomial of degree 'p' by using these features:

$$Z = \begin{bmatrix} 1 & x_1 & (x_1)^2 & \dots & (x_1)^p \\ 1 & x_2 & (x_2)^2 & \dots & (x_2)^p \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_n & (x_n)^2 & \dots & (x_n)^p \end{bmatrix}$$

- There are polynomial basis functions that are numerically nicer!
 - e.g. Lagrange polynomials
- Instead of polynomial basis functions, radial basis functions are also conventional

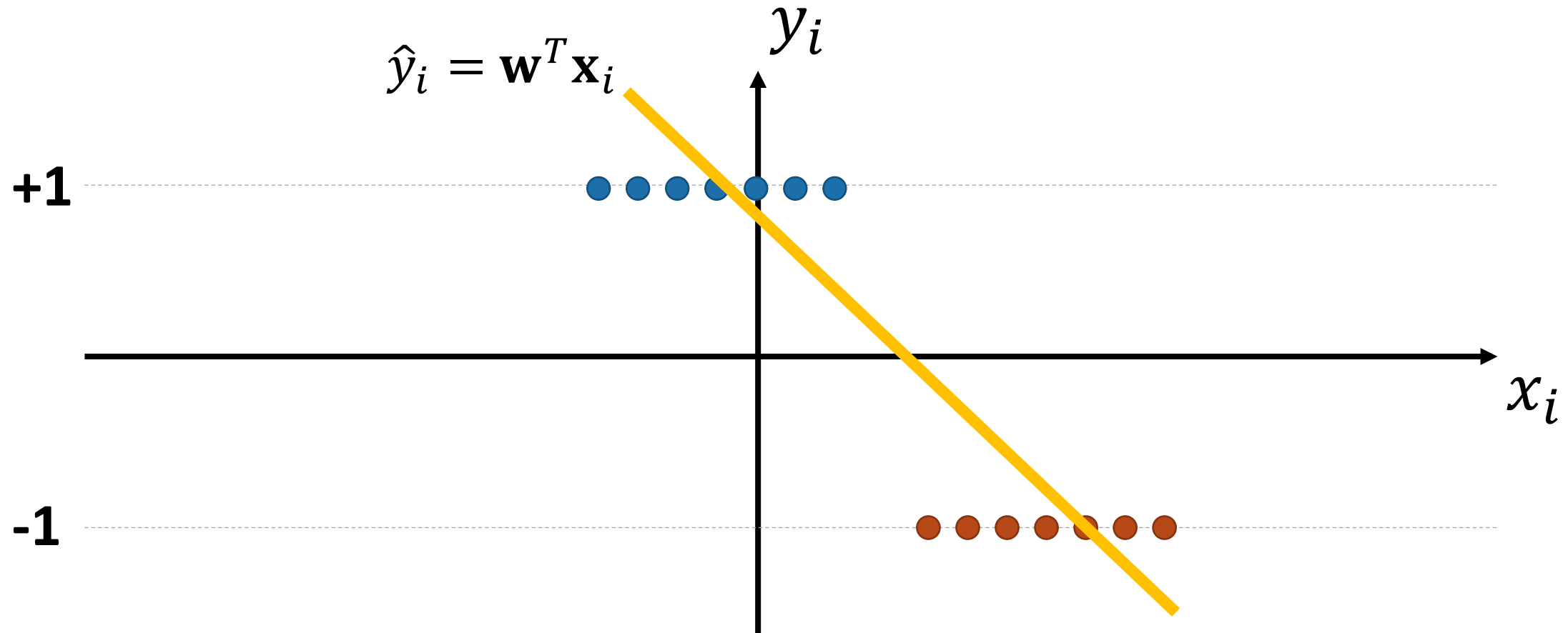
Linear Model Classification

- Binary classification using regression
 - Set $y_i = +1$ for one class
 - Set $y_i = -1$ for the other class
- At training time, fit a linear regression model:
 - $\hat{y}_i = w_1x_{i1} + w_2x_{i2} + \dots + w_dx_{id} = \mathbf{w}^T \mathbf{x}_i$
 - Then, the model will try to make $\mathbf{w}^T \mathbf{x}_i = +1$ for one class, and $\mathbf{w}^T \mathbf{x}_i = -1$ for the other class

Linear Model Classification

- Binary classification using regression
 - Set $y_i = +1$ for one class
 - Set $y_i = -1$ for the other class
- But, the linear model gives real numbers like 0.9, -1.1, and so on.
 - So, we look at whether $\mathbf{w}^T \mathbf{x}_i$ is closer to +1 or -1.
 - $\mathbf{w}^T \mathbf{x}_i = 0.9$, then $\hat{y}_i = +1$
 - $\mathbf{w}^T \mathbf{x}_i = -1.1$, then $\hat{y}_i = -1$
 - $\mathbf{w}^T \mathbf{x}_i = 0.1$, then $\hat{y}_i = +1$
 - This operation is $\hat{y}_i = \text{sign}(\mathbf{w}^T \mathbf{x}_i)$

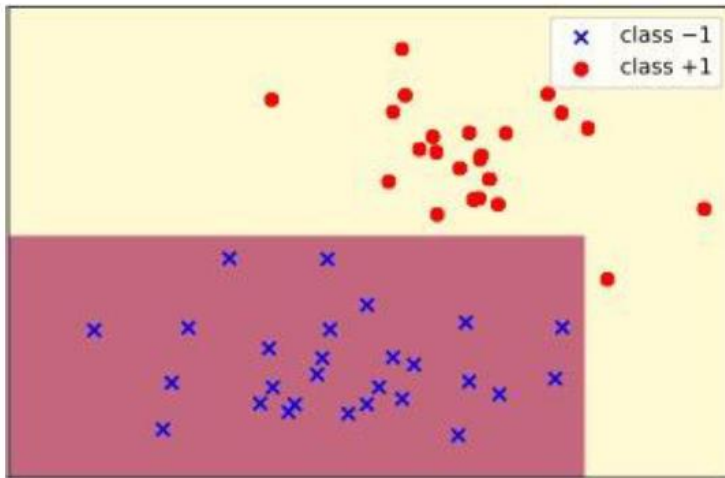
Decision Boundary in 1D



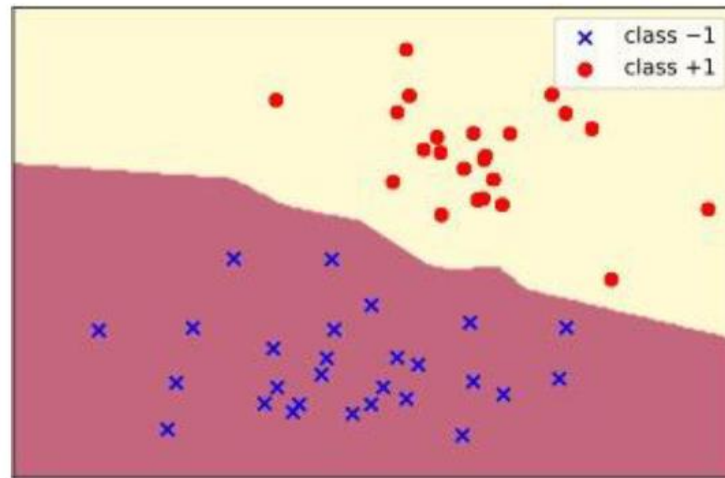
- We can interpret 'w' as a hyperplane separating x into sets:
 - set where $\mathbf{w}^T \mathbf{x}_i > 0$ and set where $\mathbf{w}^T \mathbf{x}_i < 0$

Decision Boundary in 2D

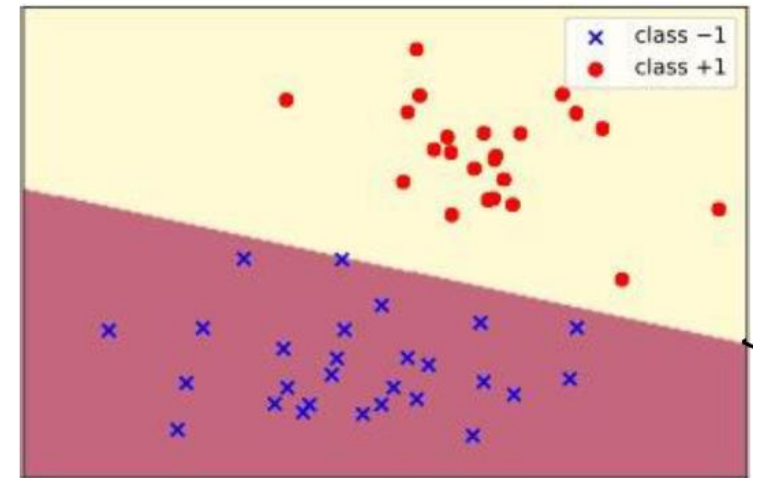
decision tree



KNN



linear classifier



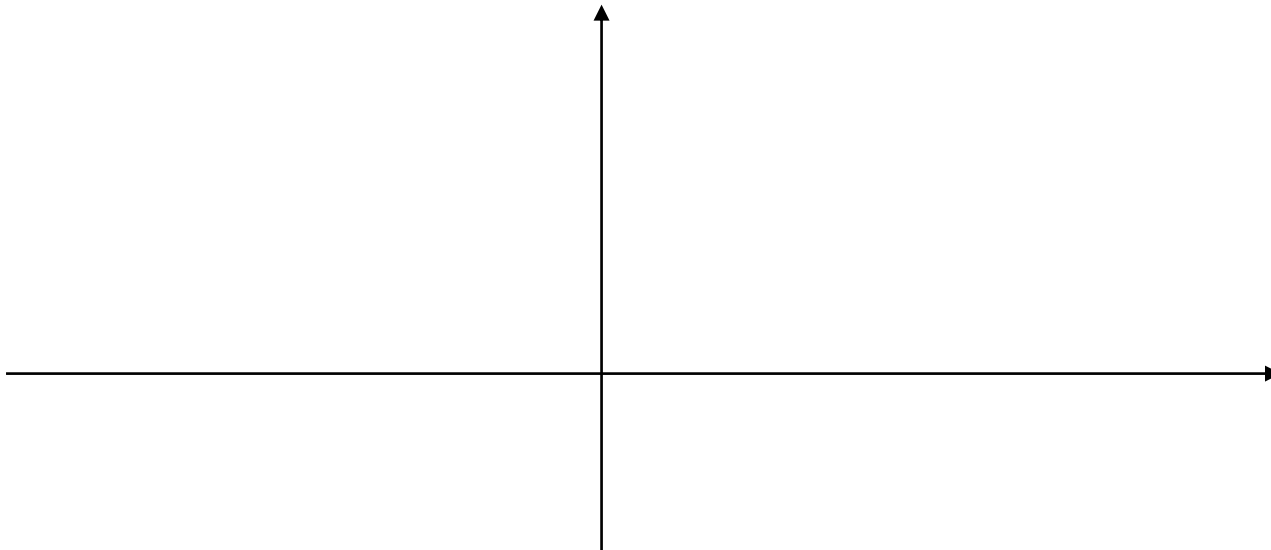
- Then, the decision boundary is at $\hat{y}_i = 0$

Linear Model Classification with Least Squares

- Let's consider training by minimizing squared error with y_i
 - $f(w) = \frac{1}{2} \|Xw - y\|^2$, where $y_i \in \{-1, +1\}$
 - If we predict $\mathbf{w}^T \mathbf{x}_i = +0.9$ and $y_i = +1$, error is small: $(0.9 - 1)^2 = 0.01$
 - If we predict $\mathbf{w}^T \mathbf{x}_i = -0.9$ and $y_i = +1$, error is small: $(-0.8 - 1)^2 = 3.24$
 - If we predict $\mathbf{w}^T \mathbf{x}_i = +100$ and $y_i = +1$, error is large: $(100 - 1)^2 = 9801$
 - However, this is correct answer!
- Thus, the least squares penalized for being “too right”

0-1 Loss Function

- Minimize the number of classification errors
 - By using L0-norm,
 - $\|\hat{y} - y\|_0$
 - Unlike regression, in classification it is reasonable that $\hat{y}_i = y_i$



0-1 Loss Function

- Minimize the number of classification errors
 - By using L0-norm,
 - $\|\hat{y} - y\|_0$
 - Unlike regression, in classification it is reasonable that $\hat{y}_i = y_i$
- Unfortunately, 0-1 loss is non-convex in 'w'
- Gradient is zero everywhere – Can't apply the gradient descent
- We need to approximate the 0-1 loss function

Degenerate Convex Approximation of 0-1 Loss

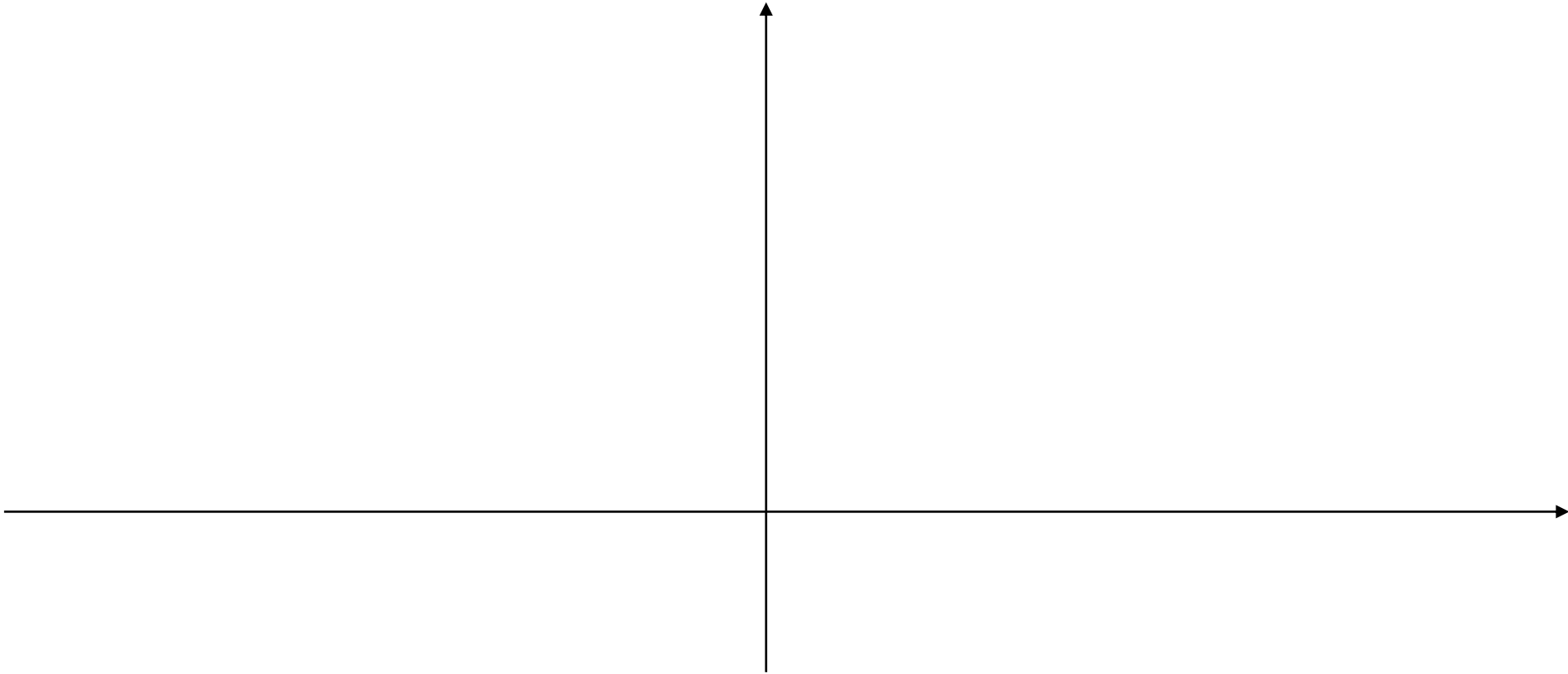
- Our convex approximation of the error for one example is:
 - $\max\{0, -y_i \mathbf{w}^T \mathbf{x}_i\}$
- We could train by minimizing sum over all examples:
 - $f(\mathbf{w}) = \sum_{i=1}^n \max\{0, -y_i \mathbf{w}^T \mathbf{x}_i\}$
- But this has a degenerate solution:
 - We have $f(\mathbf{0})=0$, and this is the lowest possible value of 'f'
- There are two standard fixes: hinge loss and logistic loss

Hinge Loss

- We saw that we classify examples 'I' correctly if $y_i \mathbf{w}^T \mathbf{x}_i > 0$
 - Our convex approximation is the amount this inequality is violated
- Consider replacing $y_i w^T x_i > 0$ with $y_i w^T x_i > 1$
- The violation of this constraint is now given by:
 - $\max\{0, 1 - y_i \mathbf{w}^T \mathbf{x}_i\}$
 - It's convex: $\max(\text{constant}, \text{linear})$
 - It's not degenerate: $w=0$ now gives an error of 1 instead of 0

Hinge Loss

- $\max\{0, 1 - y_i \mathbf{w}^T \mathbf{x}_i\}$



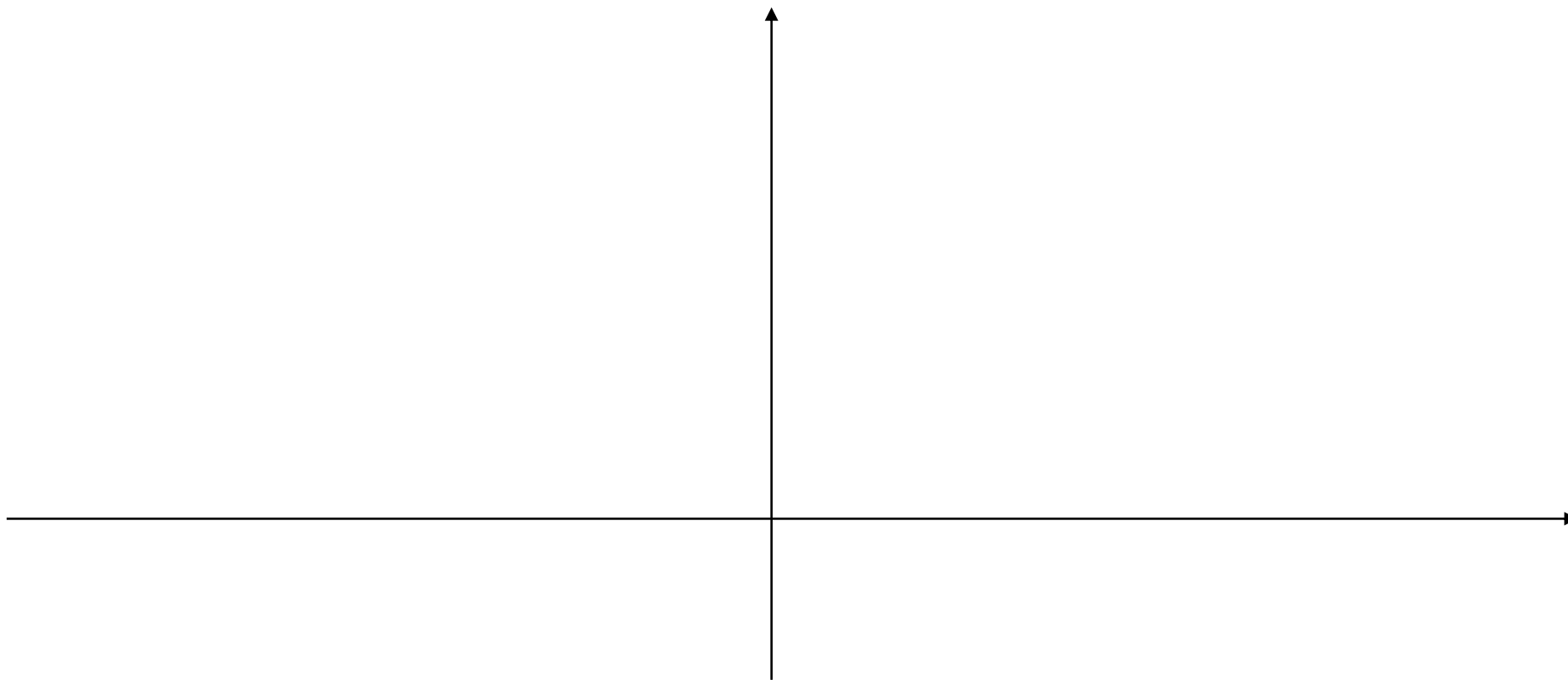
Hinge Loss

- $f(\mathbf{w}) = \sum_{i=1}^n \max\{0, 1 - y_i \mathbf{w}^T \mathbf{x}_i\}$
 - Convex upper bound on 0-1 loss
 - If the hinge loss is 18.3, then number of training errors is “at most” 18
 - So minimizing hinge loss indirectly tries to minimize training error
- Support vector machine (SVM) is hinge loss with L2-regularization!!
 - $f(\mathbf{w}) = \sum_{i=1}^n \max\{0, 1 - y_i \mathbf{w}^T \mathbf{x}_i\} + \frac{\lambda}{2} \|\mathbf{w}\|^2$
 - “Maximizing the margin”

Logistic Loss

- We can smooth max in degenerate loss with log-sum-exp:
 - $\max\{0, -y_i \mathbf{w}^T \mathbf{x}_i\} \approx \log(\exp(0) + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$
- Summing over all examples gives:
 - $f(w) = \sum_{i=1}^n \log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$
- This is the “logistic loss” and model is called “logistic regression”
 - It’s not degenerate: $w=0$ now gives an error of $\log(2)$ instead of 0
 - Convex and differentiable: can be minimized by gradient descent!
 - You should also add regularization

Convex Approximation to 0-1 Loss



Logistic Regression and SVMs

- Logistic regression and SVMs are used EVERYWHERE!
 - Fast training and testing
 - Training on huge datasets using “stochastic” gradient descent (deep learning)
 - Prediction is just computing $\mathbf{w}^T \mathbf{x}_i$
 - Weight \mathbf{w}_j are easy to understand
 - It’s how much \mathbf{w}_j changes the prediction and in what direction
 - We can often get a good test error
 - With low-dimensional features using RBFs and regularization
 - With high-dimensional features and regularization
 - Smoother predictions than random forests

Multi-class Classification – one-vs-all

- Suppose you only know how to do binary classification:
 - “One vs all” is a way to turn a binary classifier into a multi-class method
- Training phase
 - For each class ‘c’, train binary classifier to predict whether example is a ‘c’
 - If we have ‘k’ classes, this gives ‘k’ binary classifiers
- Prediction phase
 - Apply the ‘k’ binary classifiers to get a ‘score for each class ‘c’
 - Predict the ‘c’ with the highest score

“One vs All” Linear Classification

- “One vs all” logistic regression for classifying as cat/dog/person
 - Train a separate classifier for each class
 - Classifier 1 tries to predict +1 for “cat” images and -1 for “dog” and “person” images
 - Classifier 2 tries to predict +1 for “dog” images and -1 for “cat” and “person” images
 - ...
 - Results in a weight vector w_c for each class ‘c’:
 - Weights w_c try to predict +1 for class ‘c’ and -1 for all others
 - We’ll use ‘W’ as a matrix with the w_c as rows

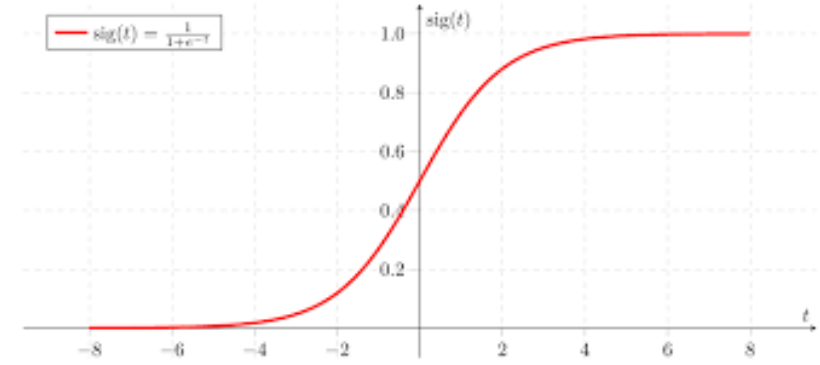
Multi-Class SVMs

- We need to encourage the largest $w_c^T x_i$ to be $w_{y_i}^T x_i$
 - We want $w_{y_i}^T x_i > w_c^T x_i$ for all 'c' that are not correct label y_i
 - We use $w_{y_i}^T x_i \geq w_c^T x_i + 1$ for all $c \neq y_i$ to avoid strict inequality
- Thus, there are two ways to measure constraint violation:
 - $\sum_{c \neq y_i} \max\{0, 1 - w_{y_i}^T x_i + w_c^T x_i\}$
 - $\max_{c \neq y_i} \max\{0, 1 - w_{y_i}^T x_i + w_c^T x_i\}$
 - By adding L2-regularization, both are called multi-class SVMs

Predictions vs. Probabilities

- For predictions, “sign” maps from $w^T x_i$ to the elements $\{-1, +1\}$
 - If $w^T x_i$ is positive we predict +1, if it's negative we predict -1
- For probabilities, we want to map from $\mathbf{w}^T \mathbf{x}_i$ to the range $[0, 1]$
 - If $\mathbf{w}^T \mathbf{x}_i$ is very positive, we output a value close to +1
 - If $\mathbf{w}^T \mathbf{x}_i$ is very negative, we output a value close to 0.0
 - If $\mathbf{w}^T \mathbf{x}_i$ is close to 0, we output a value close to 0.5

Sigmoid function



- The most common choice is the sigmoid function:

- $h(z_i) = \frac{1}{1 + \exp(-z_i)}$

- We can output probabilities for linear models using:

- $p(y_i = 1|w, x_i) = \frac{1}{1 + \exp(-w^T x_i)}$

- $p(y_i = -1|w, x_i) = 1 - \frac{1}{1 + \exp(-w^T x_i)} = \frac{1}{1 + \exp(w^T x_i)}$

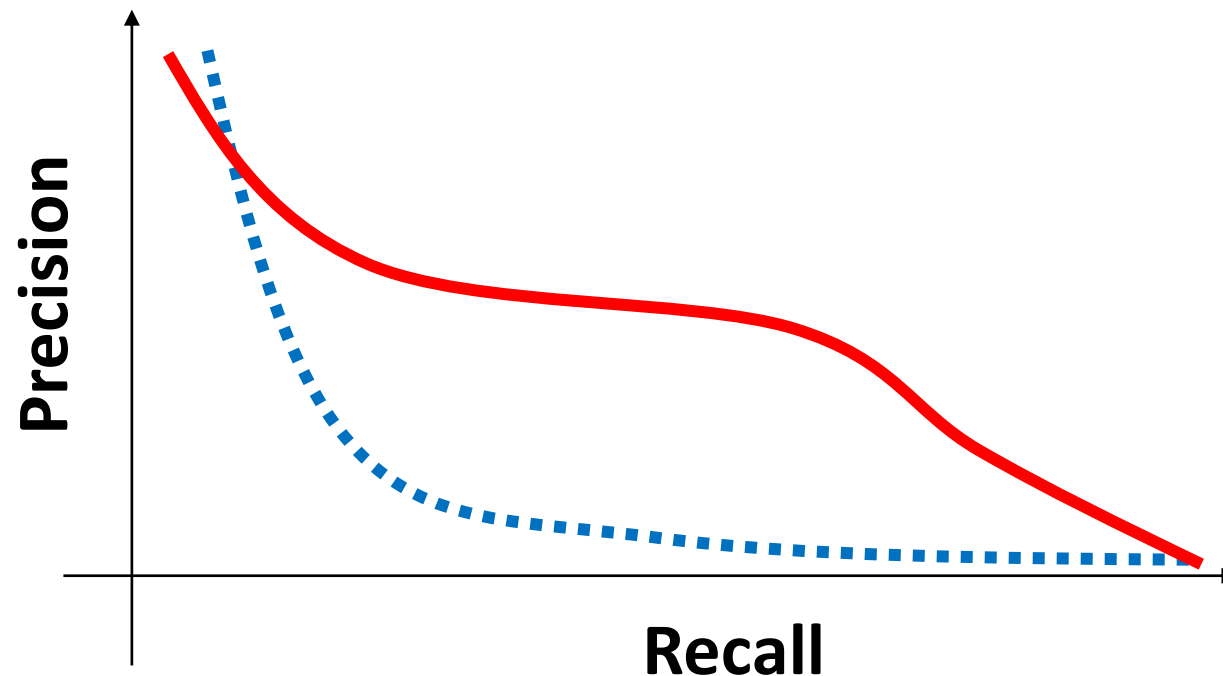
Performance Measure – Precision/Recall

- **To report the classification performance, we use precision and recall**
 - Precision: $TP/(TP+FP)$
 - “Real” true among all the “Predict” true
 - Recall: $TP/(TP+FN)$
 - “Predict” true among all the “Real” true

Predict / True	True ‘spam’	True ‘not spam’
Predict ‘spam’	True Positive	False Positive
Predict ‘not spam’	False Negative	True Negative

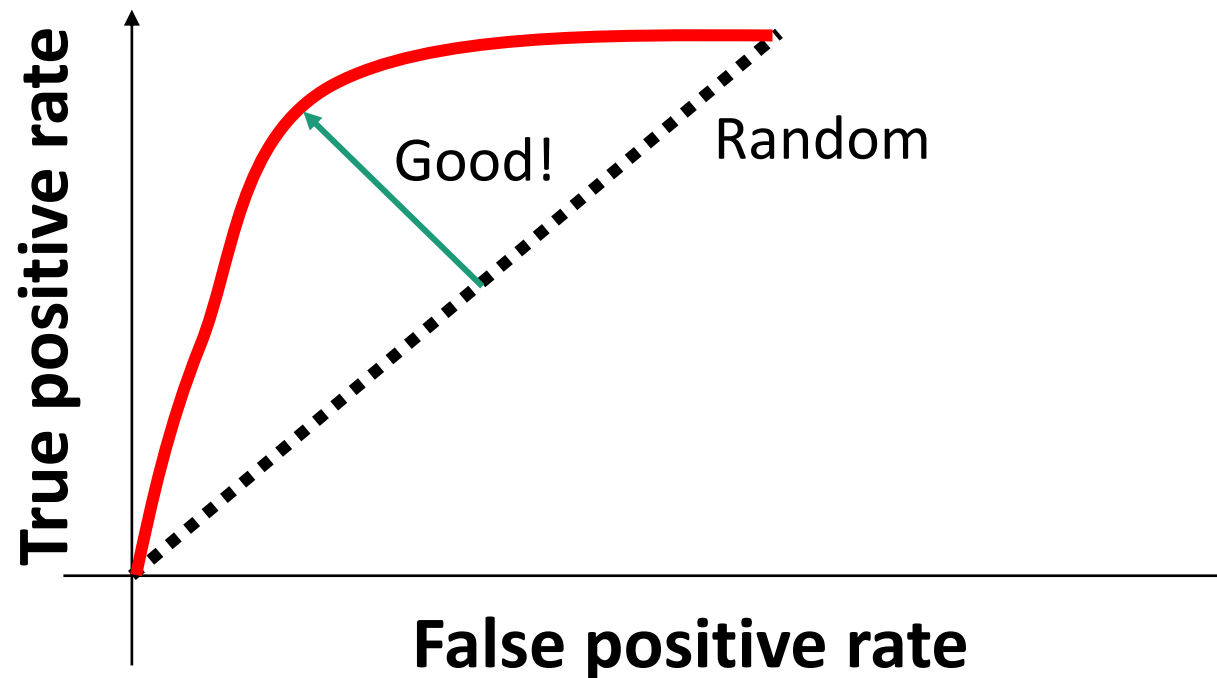
Performance Measure – Precision-Recall Curve

- When we consider the rule $p(y_i = \textit{spam} \mid x_i) > t$,
- By varying the threshold t to obtain the precision-recall curve



Performance Measure – ROC Curve

- ROC curve – Receiver Operating Characteristic curve
 - Plot true positive rate (recall) vs. false positive rate ($FP/(FP+TN)$)
 - Sometimes, papers report area under curve (AUC)

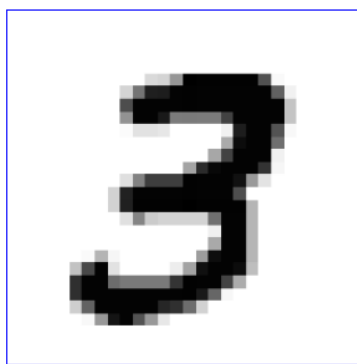


Performance Measure – F1 Score

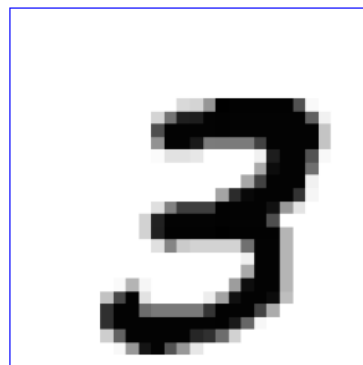
- **For statistical analysis of binary classification**
- **The harmonic mean of precision and recall**
 - $F_1 = \left(\frac{2}{recall^{-1} + precision^{-1}} \right) = 2 \cdot \frac{precision \cdot recall}{precision + recall}$
- **Weighted F1 score**
 - Extension of F1 score for multi-class classification problem
 - Estimate the weighted average of class-wise F1 scores
 - Weighted by the number of real true samples

Encouraging invariance

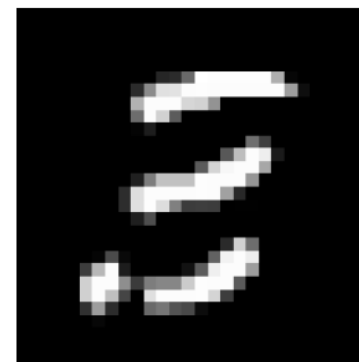
- **The distance for Human vs. Computer**



Sample 1



Sample 2



Distance

- We want the classifiers to be invariant to the feature transforms!
 - Ex. Image translations, rotations, scale changes, etc.
- 2 methods to solve the problem
 - Modify the distance function to be invariant (hard/slow)
 - Add transformed data during training (easy/fast)