DS 4400

Machine Learning and Data Mining I
Spring 2022

Alina Oprea
Associate Professor
Khoury College of Computer Science
Northeastern University

February 14 2022

Outline

- Regularization
  - A geometric understanding
- Classification
  - K Nearest Neighbors (kNN)
- Cross validation
  - K-fold cross validation
  - Leave one out cross validation
- Linear classifiers

Announcements

- Thank you for survey feedback!
  - Continue providing feedback if you haven't yet via Google Form posted on Piazza.
  - Clear request for more whiteboarding.
  - Request for examples, demos to understand the equations.
  - Will likely reduce supplemental lectures as a result.

- Homework 2 is posted

Ridge regression

- Linear regression objective function

$$J(\theta) = \underbrace{\frac{1}{2}\sum_{i=1}^{N}[h_\theta(x_i) - y_i]^2}_{\text{model fit to data}} + \underbrace{\frac{\lambda}{2}\sum_{j=1}^{d}\theta_j^2}_{\text{regularization}}$$

  – λ is the regularization parameter ($\lambda \geq 0$)
  – No regularization on θ_0!
    - If λ = 0, we train linear regression
    - If λ is large, the coefficients will shrink close to 0

Lasso Regression

$$J(\theta) = \underbrace{\sum_{i=1}^{N} (h_\theta(x_i) - y_i)^2}_{\text{Squared Residuals}} + \lambda \underbrace{\sum_{j=1}^{d} |\theta_j|}_{\text{Regularization}}$$

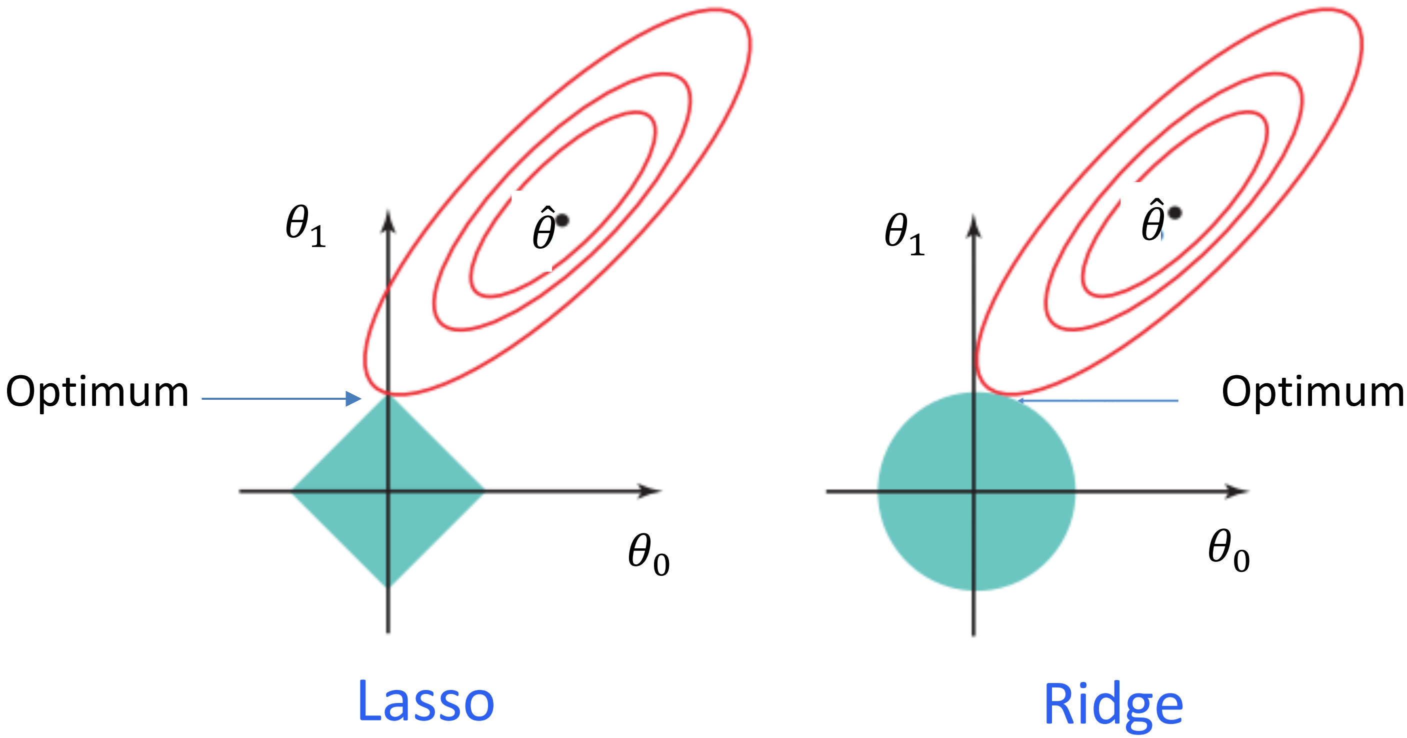
- L1 norm for regularization
- Results in sparse coefficients
- Issue: gradients cannot be computed around 0
- Method of sub-gradient optimization

Alternative Formulations

- Ridge
  - L2 Regularization
  - $\min_{\theta} \sum_{i=1}^{N} [h_{\theta}(x_i) - y_i]^2$ subject to $\sum_{j=1}^{d} |\theta_j|^2 \leq \epsilon$
- Lasso
  - L1 regularization
  - $\min_{\theta} \sum_{i=1}^{N} [h_{\theta}(x_i) - y_i]^2$ subject to $\sum_{j=1}^{d} |\theta_j| \leq \epsilon$

Lasso vs Ridge

- Ridge shrinks all coefficients
- Lasso sets some coefficients at 0 (sparse solution)
  - Perform feature selection



Ridge vs Lasso

- Both methods can be applied to any loss function (regression or classification)

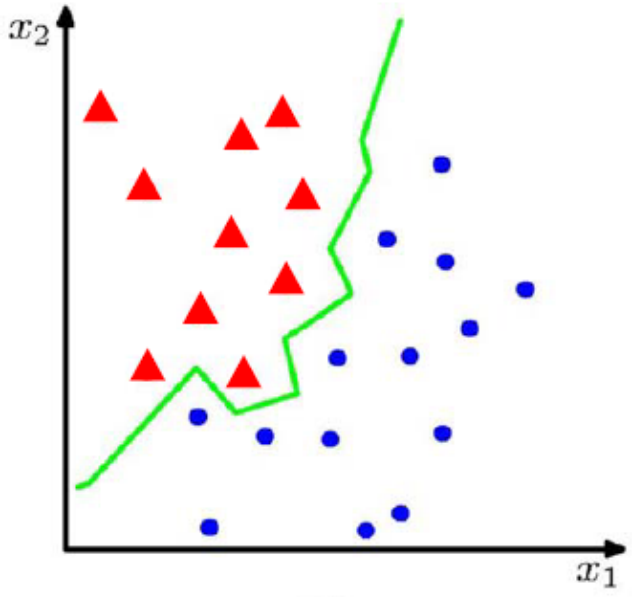
• Ridge	• Lasso

Ridge vs Lasso

- Both methods can be applied to any loss function (regression or classification)
- In both methods, value of regularization parameter λ needs to be adjusted
- Both reduce model complexity

Ridge	**Lasso**
+ Differentiable objective	- Gradient descent needs to be adapted
+ Gradient descent converges to global optimum	+ Results in sparse model
- Shrinks all coefficients	+ Can be used for feature selection in large dimensions

Classification



Binary or discrete

- Suppose we are given a training set of N observations

$$\{x_1, \dots, x_N\} \text{ and } \{y_1, \dots, y_N\}, x_i \in R^d, y_i \in \{0, 1\}$$

- Classification problem is to estimate f(x) from this data such that

$$f(x_i) = y_i$$

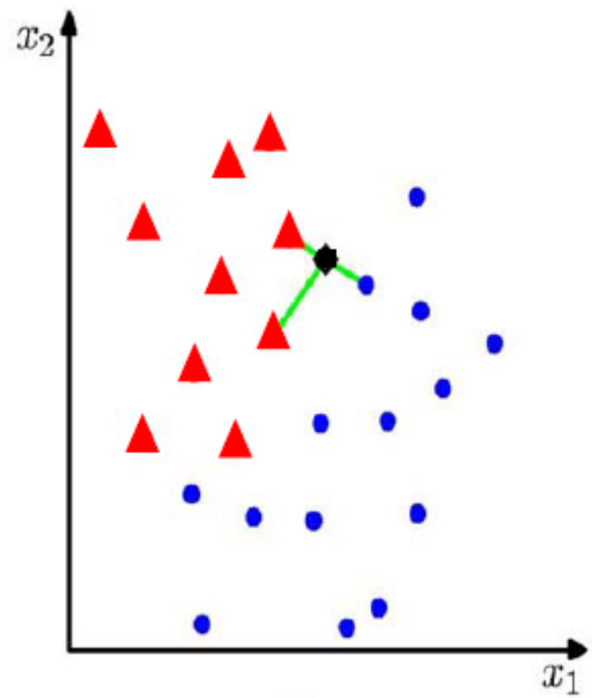
K Nearest Neighbour (K-NN) Classifier

Algorithm

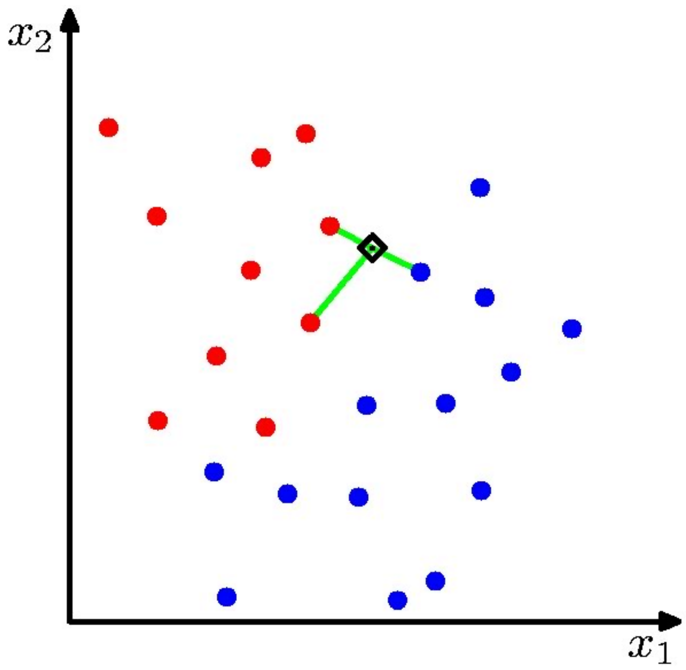
- For each test point, x, to be classified, find the K nearest samples in the training data
- Classify the point, x, according to the majority vote of their class labels

e.g. K = 3

- applicable to multi-class case

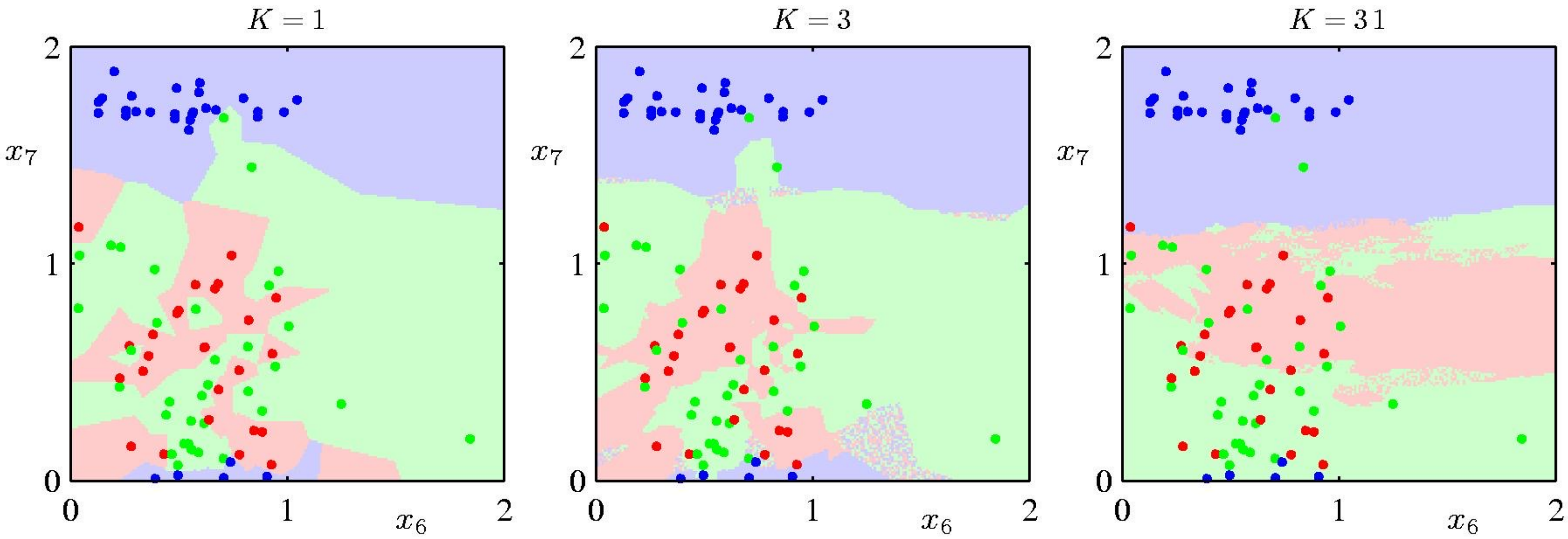


kNN



- Algorithm (to classify point x)
  - Find k nearest points to x (according to distance metric)
  - Perform majority voting to predict class of x
- Properties
  - Does not learn any model in training!
  - Instance learner (needs all data at testing time)

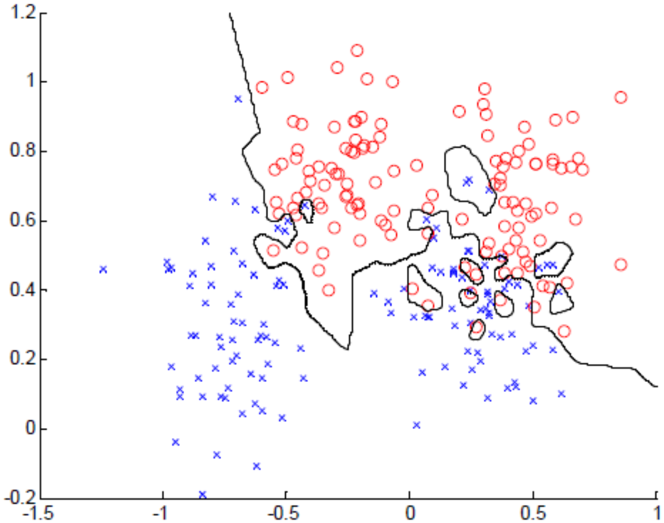
K-Nearest-Neighbours for Multi-class Classification



Vote among multiple classes

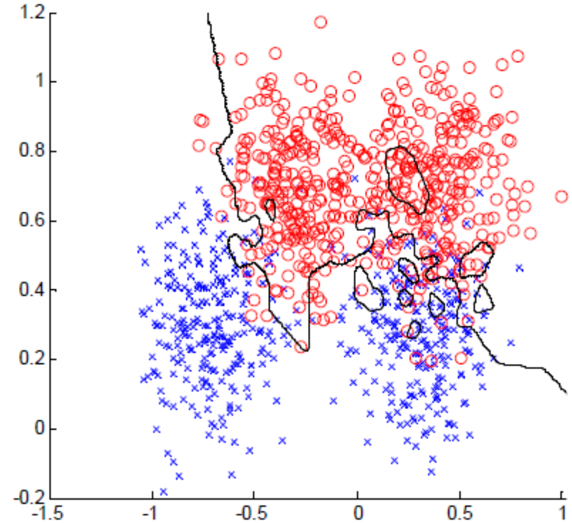
K = 1

Training data



error = 0.0

Testing data

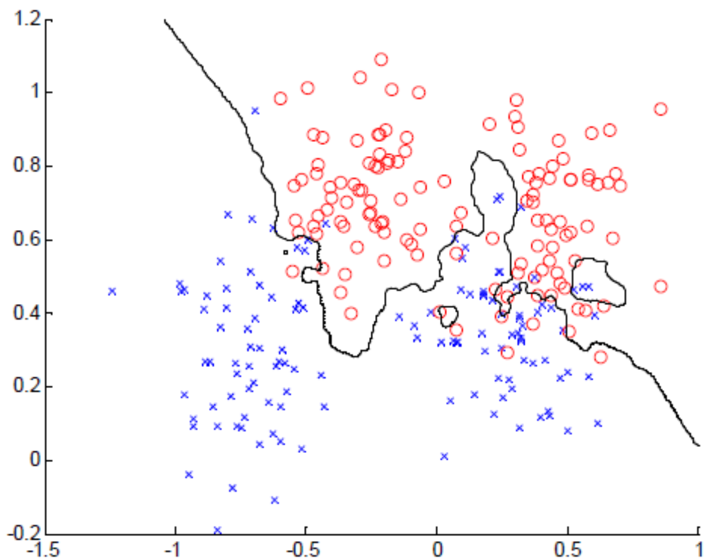


error = 0.15

How to choose k (hyper-parameter)?

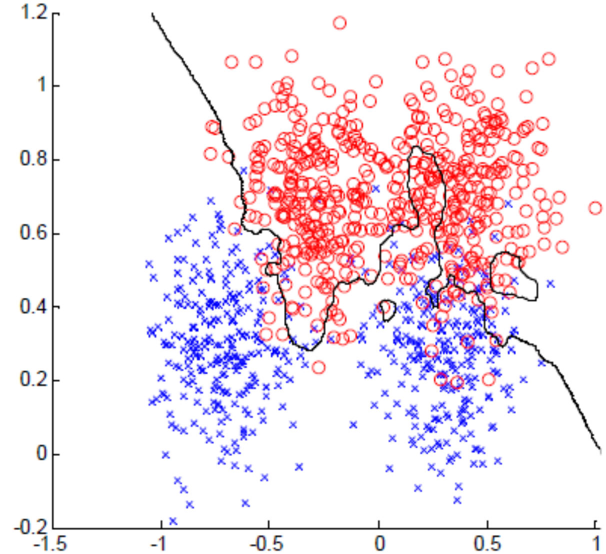
K = 3

Training data



error = 0.0760

Testing data

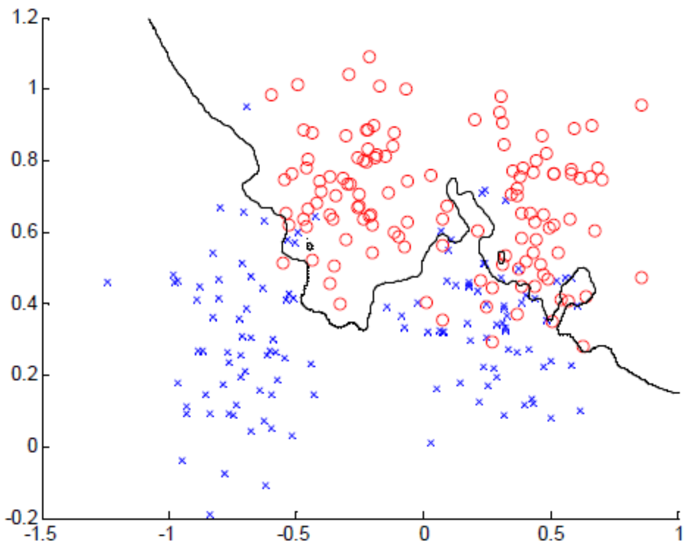


error = 0.1340

How to choose k (hyper-parameter)?

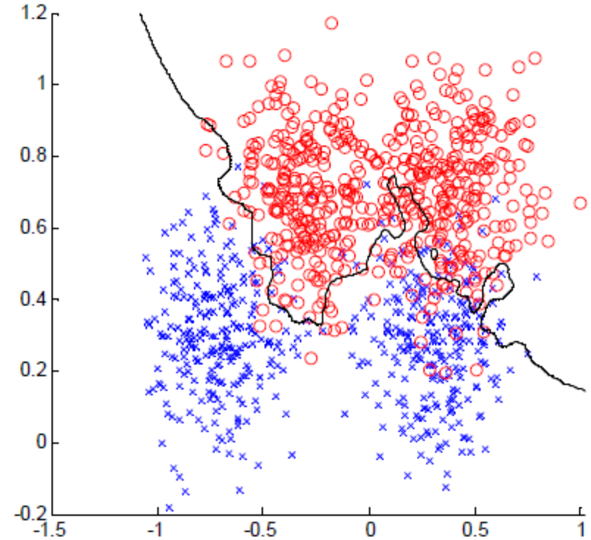
K = 7

Training data



error = 0.1320

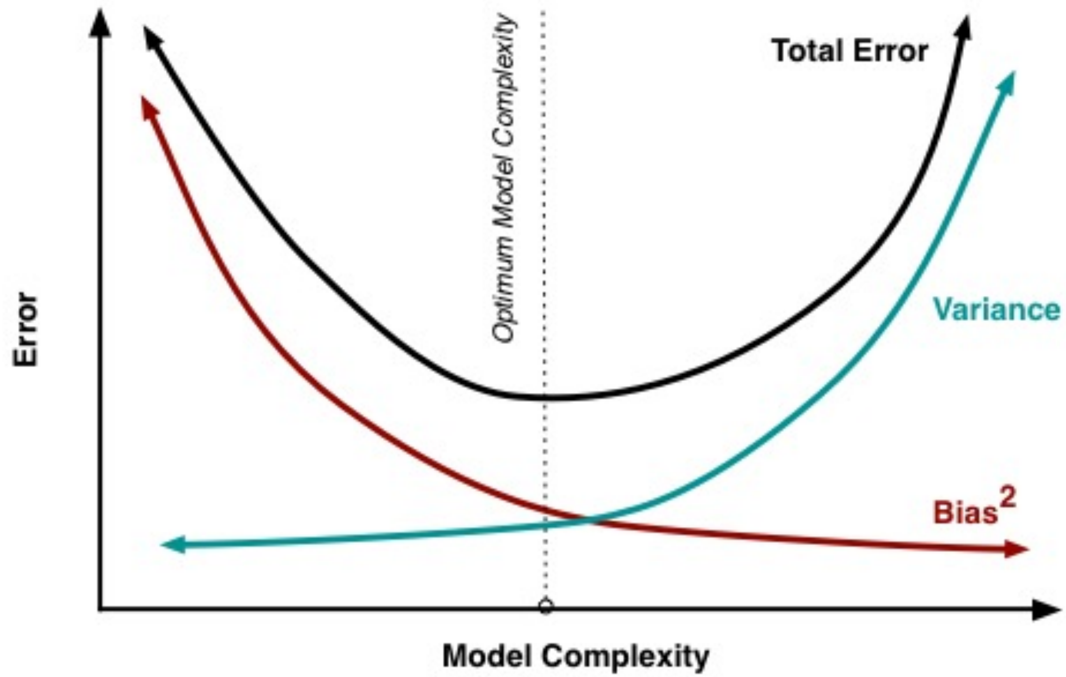
Testing data



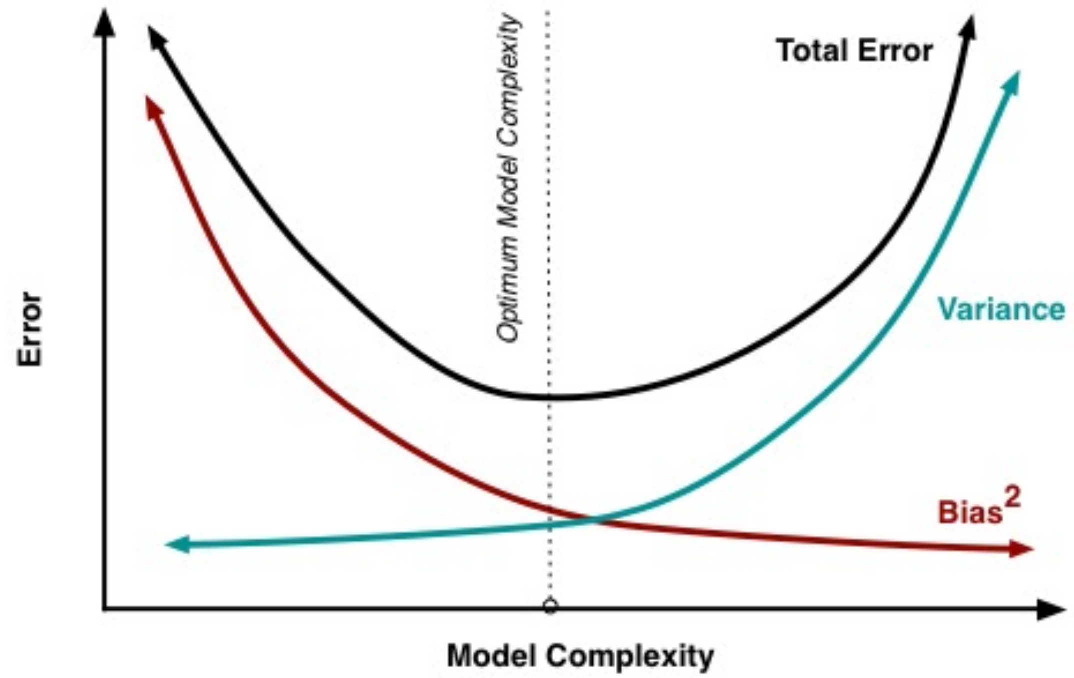
error = 0.1110

How to choose k (hyper-parameter)?

Bias-Variance Tradeoff for kNN

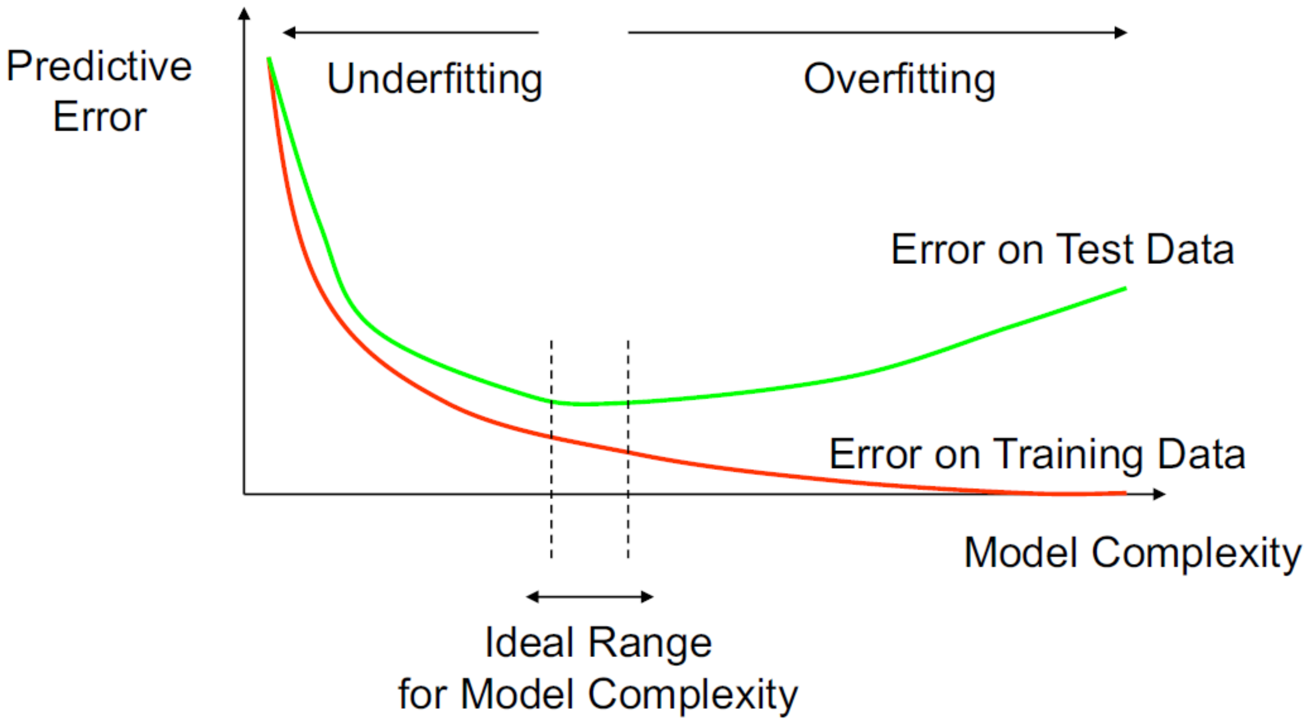


Bias-Variance Tradeoff for kNN



K decreases

How Overfitting Affects Prediction



- How to pick hyper-parameters without access to testing data?
- Goal: Reduce overfitting and variance

Important: Do not use testing data for hyper-parameter selection even if it is available

Cross Validation

As K increases:

- Classification boundary becomes smoother
- Training error can increase

Choose (learn) K by cross-validation

- Split training data into training and validation
- Hold out validation data and measure error on this

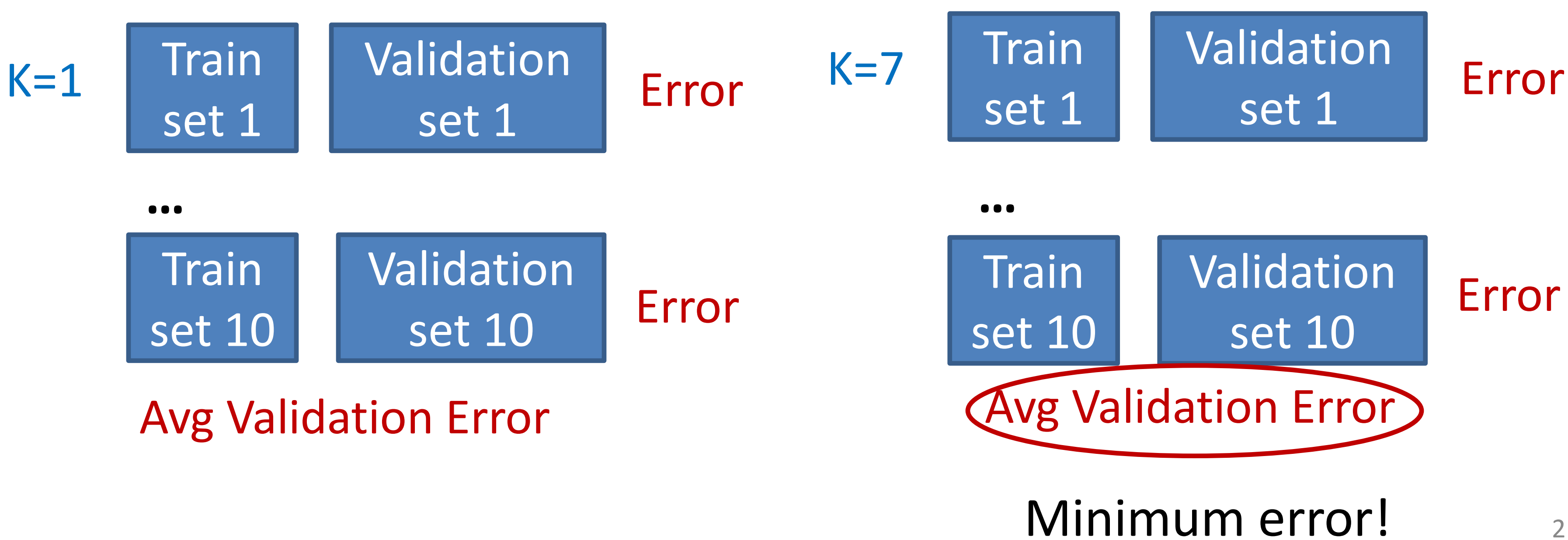
Cross Validation

As K increases:

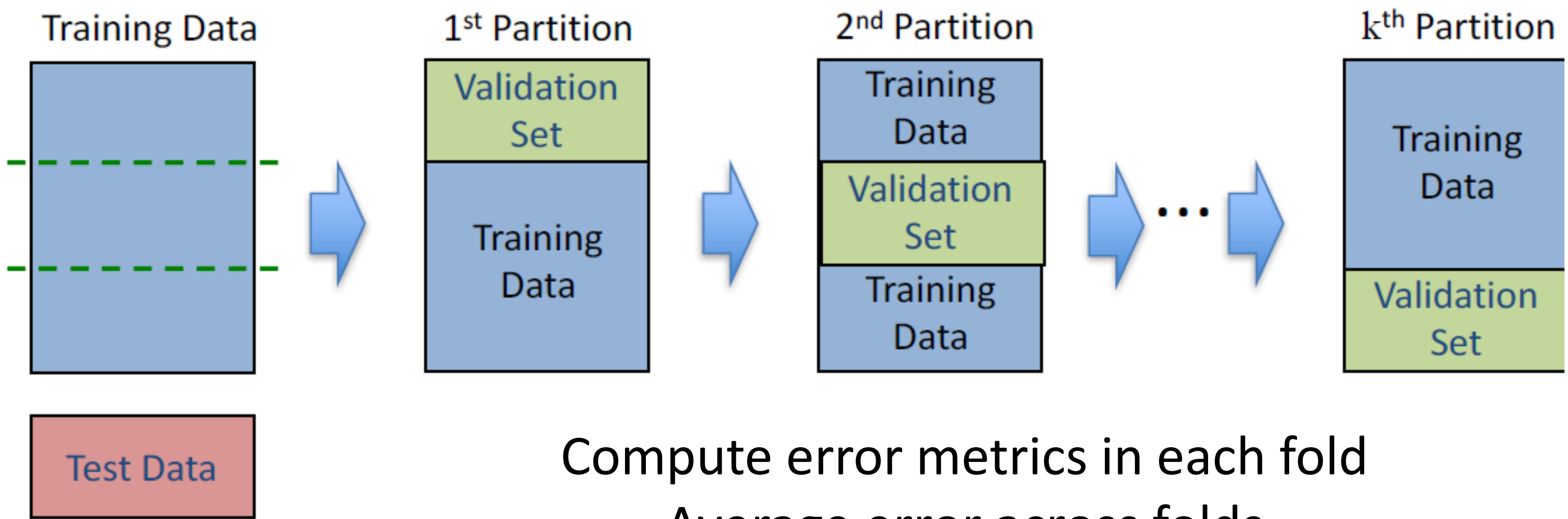
- Classification boundary becomes smoother
- Training error can increase

Choose (learn) K by cross-validation

- Split training data into training and validation
- Hold out validation data and measure error on this



Cross Validation

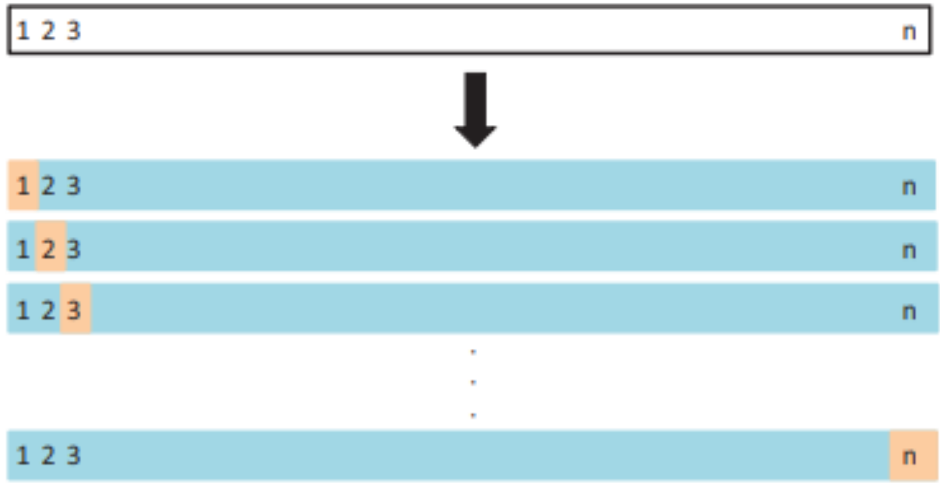


Compute error metrics in each fold
Average error across folds

1. k-fold CV

- Split training data into k partitions (folds) of equal size
- Pick the optimal value of hyper-parameter according to error metric averaged over all folds

Cross Validation



2. Leave-one-out CV (LOOCV)

- k=n (validation set only one point)

- Pros: Less bias
- Cons: More expensive to implement, higher variance
- Used for small training sets

Recommendation: perform k-fold CV with k=5 or k=10

Cross-Validation Takeaways

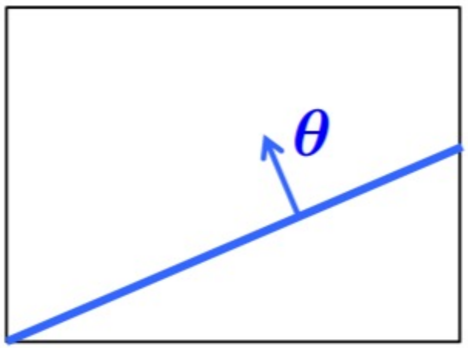
- General method to estimate performance of ML model at testing and select hyper-parameters
  - Improves model generalization
  - Avoids overfitting to training data
- Techniques for CV: k-fold CV and LOOCV
- Compare to regularization

Cross-Validation Takeaways

- General method to estimate performance of ML model at testing and select hyper-parameters
  - Improves model generalization
  - Avoids overfitting to training data
- Techniques for CV: k-fold CV and LOOCV
- Compare to regularization
  - Regularization works when training with GD
  - Cross-validation can be used for hyper-parameter selection
  - The two methods can be combined

Linear classifiers

- A **hyperplane** partitions space into 2 half-spaces
  - Defined by the normal vector $\boldsymbol{\theta} \in \mathbb{R}^{d+1}$
    - $\boldsymbol{\theta}$ is orthogonal to any vector lying on the hyperplane

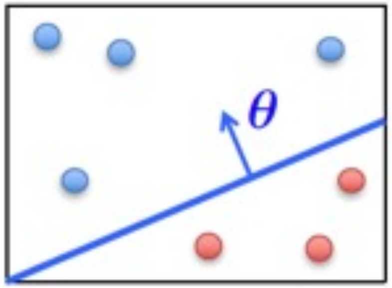


- Consider classification with +1, -1 labels ...

Linear Classifiers

- **Linear classifiers**: represent decision boundary by hyperplane

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \quad \boldsymbol{x}^\intercal = \begin{bmatrix} 1 & x_1 & \ldots & x_d \end{bmatrix}$$

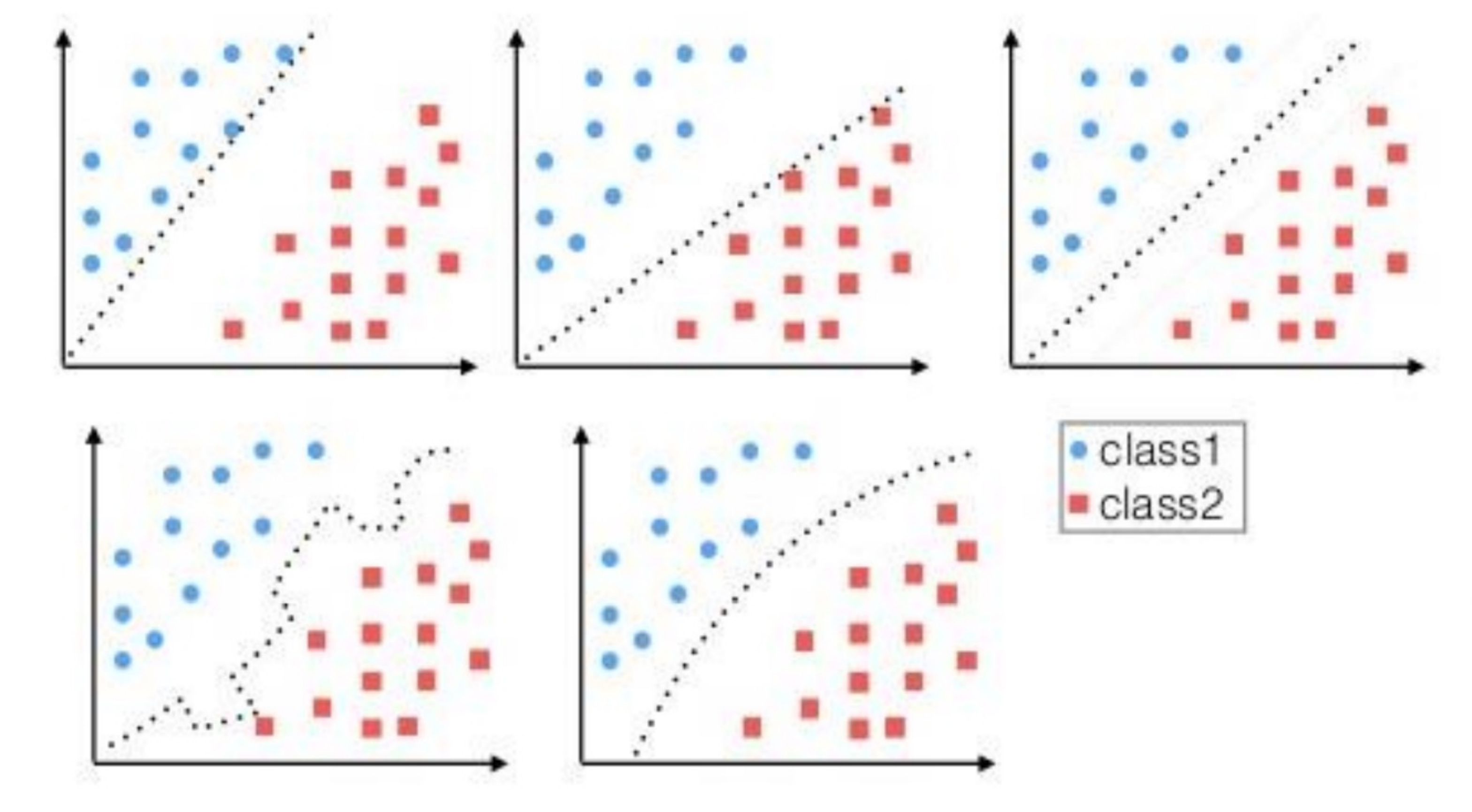


$h_\theta(x) = f(\theta^T x)$ linear classifier

- If $\theta^T x > 0$ classify "Class 1"
- If $\theta^T x < 0$ classify "Class 0"

All the points x on the hyperplane satisfy: $\theta^T x = 0$

Linear vs Non-Linear Classifiers



Classification Based on Probability

- Instead of just predicting the class, give the *probability of the instance being in that class*

- Consider binary classifier with classes 0 and 1

- Advantages: interpretability and confidence of output

Classification Based on Probability

- Instead of just predicting the class, give the *probability of the instance being in that class*
  - Learn $P(Y|X)$
- Consider binary classifier with classes 0 and 1
  - $P(Y = 1|X) + P(Y = 0|X) = 1$
  - Sufficient to learn $P(Y = 1|X)$
- Advantages: interpretability and confidence of output

Logistic Regression

- Setup
  - Training data: $\{x_i, y_i\}$, for $i = 1, \dots, N$
  - Labels: $y_i \in \{0,1\}$
- Goals
  - Learn $P(Y = 1|X = x)$
- Highlights
  - Probabilistic output
  - At the basis of more complex models (e.g., neural networks)
  - Supports regularization (Ridge, Lasso)
  - Can be trained with Gradient Descent

Interpretation of Model Output

$h_{\boldsymbol{\theta}}(\boldsymbol{x})$ = estimated $P(Y = 1|X;\theta)$

Example: Cancer diagnosis from tumor size

$$\boldsymbol{x} = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumorSize} \end{bmatrix}$$

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = 0.7$$

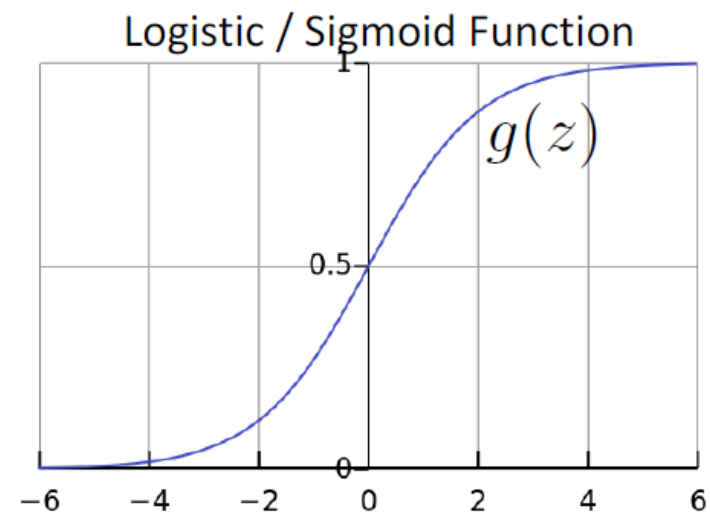
→ Tell patient that 70% chance of tumor being malignant

Note that: $P(Y = 0|X;\theta) + P(Y = 1|X;\theta) = 1$

Therefore, $P(Y = 0|X;\theta) = 1 - P(Y = 1|X;\theta)$

Logistic Regression

- Takes a probabilistic approach to learning discriminative functions (i.e., a classifier)
- $h_{\boldsymbol{\theta}}(\boldsymbol{x})$ should give $P(Y = 1|X; \theta)$
  – Want $0 \leq h_{\boldsymbol{\theta}}(\boldsymbol{x}) \leq 1$



Logistic Regression

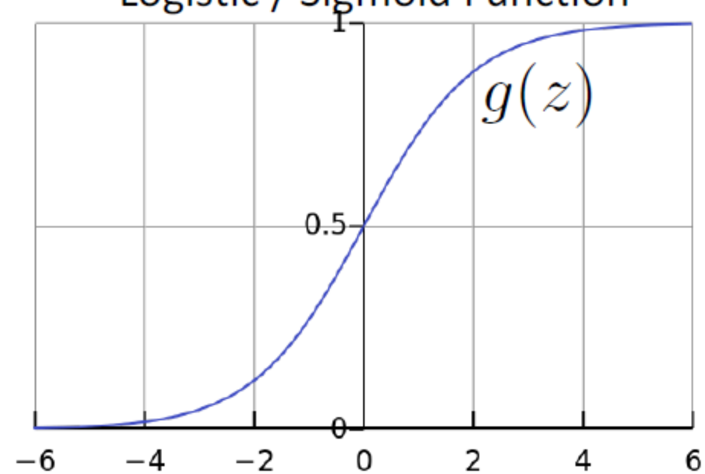
- Takes a probabilistic approach to learning discriminative functions (i.e., a classifier)
- $h_{\boldsymbol{\theta}}(\boldsymbol{x})$ should give $P(Y = 1|X;\theta)$
  - Want $0 \leq h_{\boldsymbol{\theta}}(\boldsymbol{x}) \leq 1$
- Logistic regression model:

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g\left(\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}\right)$$

$$g(z) = \frac{1}{1+e^{-z}}$$

$$\boxed{h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{1+e^{-\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}}}}$$

Logistic / Sigmoid Function



Logistic Regression Predictions

- Predict $Y = 1$ if:

LR is a Linear Classifier!

- Predict $Y = 1$ if:
  - $\mathrm{P}[Y = 1|X = x; \theta] > \mathrm{P}[Y = 0|X = x; \theta]$
  - $\mathrm{P}[Y = 1|X = x; \theta] > \frac{1}{2}$

$$\frac{1}{1 + e^{-\theta^T x}} > \frac{1}{2}$$

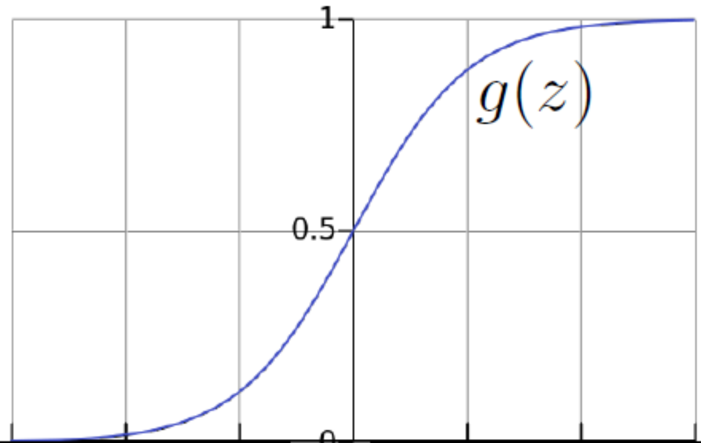
- Equivalent to:
  - $e^{\theta_0 + \sum_{j=1}^{d} \theta_j x_j} > 1$
  - $\theta_0 + \sum_{j=1}^{d} \theta_j x_j > 0$

Logistic Regression is a linear classifier!

Logistic Regression

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g\left(\boldsymbol{\theta}^\intercal \boldsymbol{x}\right)$$

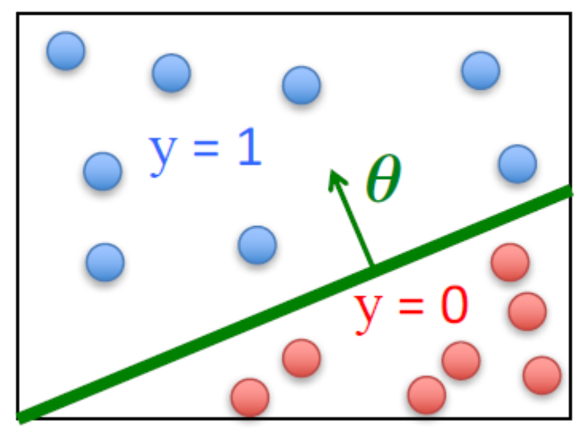
$$g(z) = \frac{1}{1 + e^{-z}}$$



$\boldsymbol{\theta}^\intercal \boldsymbol{x}$ should be large negative values for negative instances

$\boldsymbol{\theta}^\intercal \boldsymbol{x}$ should be large positive values for positive instances

- Assume a threshold and...
  - Predict $Y = 1$ if $h_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq 0.5$
  - Predict $Y = 0$ if $h_{\boldsymbol{\theta}}(\boldsymbol{x}) < 0.5$



Logistic Regression is a linear classifier!

How to Pick Loss Function?

Maximum Likelihood Estimation (MLE)

Given training data $X = \{x_1, \dots, x_N\}$ with labels $\mathrm{Y} = \{y_1, \dots, y_N\}$

What is the likelihood of training data for parameter θ?

Define likelihood function

Maximum Likelihood Estimation (MLE)

Given training data $X = \{x_1, \dots, x_N\}$ with labels $Y = \{y_1, \dots, y_N\}$

What is the likelihood of training data for parameter θ?

Define likelihood function

$$Max_\theta \, L(\theta) = P[Y|X;\theta]$$

Assumption: training points are independent

$$L(\theta) = \prod_{i=1}^{N} P[Y = y_i | X = x_i; \theta]$$

General probabilistic method for classifier training

Log Likelihood

- Max likelihood is equivalent to maximizing log of likelihood

Log Likelihood

- Max likelihood is equivalent to maximizing log of likelihood

$$L(\theta) = \prod_{i=1}^{N} P[Y = y_i | X = x_i; \theta]$$

$$\log L(\theta) = \sum_{i=1}^{N} \log P[Y = y_i | X = x_i; \theta]$$

- They both have the same maximum θ_{MLE}

Maximum Likelihood Estimation (MLE)

Given training data $X = \{x_1, \dots, x_N\}$ with labels $\mathrm{Y} = \{y_1, \dots, y_N\}$

What is the likelihood of training data for parameter θ?

Define likelihood function

$$Max_\theta \, L(\theta) = P[Y|X;\theta]$$

Assumption: training labels are conditionally independent

$$L(\theta) = \prod_{i=1}^{N} P[Y = y_i | X = x_i; \theta]$$

General probabilistic method for parameter estimation

MLE for Logistic Regression

$$\mathrm{P}(Y = y_i | X = x_i; \theta) = h_\theta(x_i)^{y_i}\left(1 - h_\theta(x_i)\right)^{1-y_i}$$

How to Train Logistic Regression

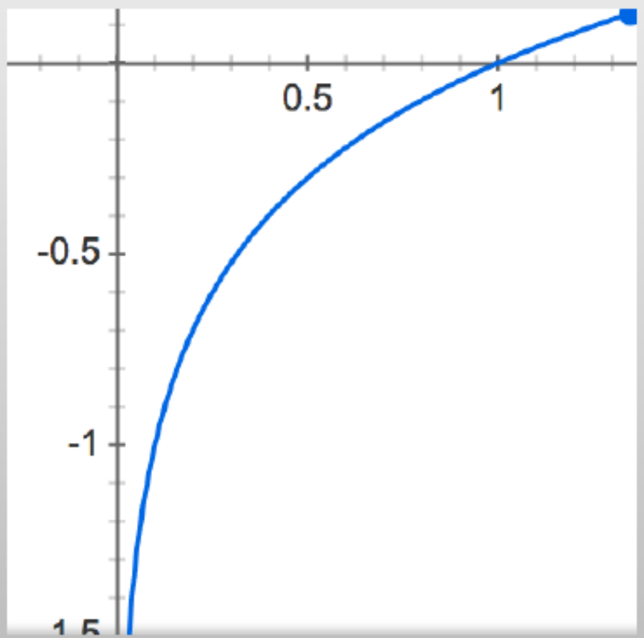
Cross-Entropy Loss Objective

$$J(\theta) = -\sum_{i=1}^{N} [y_{\mathrm{i}} \log h_\theta(x_i) + (1 - y_i) \log \left(1 - h_\theta(x_i)\right)]$$

Intuition

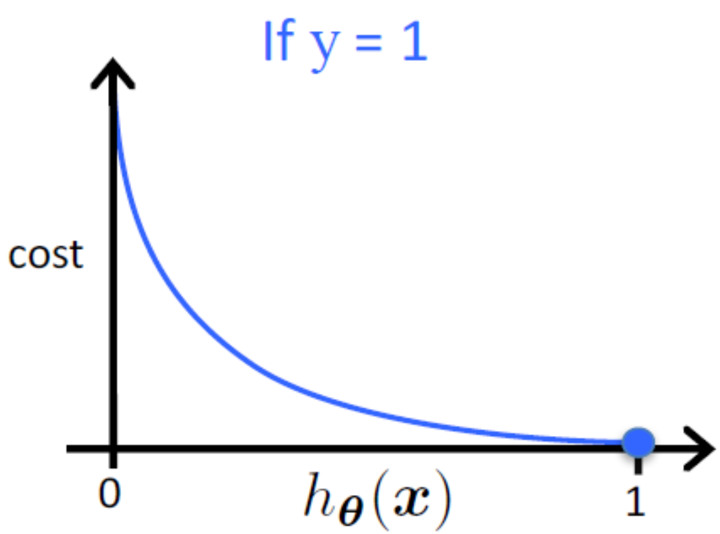
$$\text{cost}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y\right) = \begin{cases} -\log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 1 \\ -\log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 0 \end{cases}$$

Aside: Recall the plot of log(z)



Intuition

$$\text{cost}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y\right) = \begin{cases} \boxed{-\log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) \quad \text{if } y = 1} \\ -\log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) \quad \text{if } y = 0 \end{cases}$$

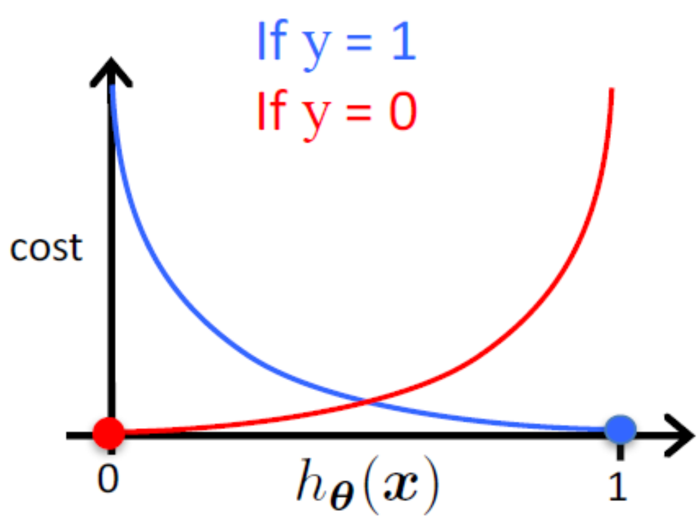


If y = 1

- Cost = 0 if prediction is correct
- As $h_{\boldsymbol{\theta}}(\boldsymbol{x}) \rightarrow 0, \text{cost} \rightarrow \infty$
- Captures intuition that larger mistakes should get larger penalties
  - e.g., predict $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = 0$, but y = 1

Intuition

$$\text{cost}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y\right) = \begin{cases} \boxed{-\log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) \quad \text{if } y = 1} \\ \boxed{-\log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) \quad \text{if } y = 0} \end{cases}$$



If y = 0

- Cost = 0 if prediction is correct
- As $(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) \to 0, \text{cost} \to \infty$
- Captures intuition that larger mistakes should get larger penalties

Logistic Regression
Lab Example

Review

- K nearest neighbors is the first example of classifier
  - Instance learner
- Cross-validation should be performed to
  - Improve generalization and avoid over-fitting
  - Choose hyper parameters (k in kNN)
- Logistic regression is a linear classifier that predicts class probability
  - Cross-entropy objective derived with MLE
  - MLE: probabilistic method of maximum likelihood