DS 5220

Supervised Machine Learning and Learning Theory

Alina Oprea
Associate Professor, CCIS
Northeastern University

September 30 2019

Logistics

- HW 2 is due on Oct. 8
- Lab session on Wed, Oct 2., 5-6pm in ISEC 655
- Exams
  - Midterm: Monday, Oct. 28
  - Final exam: Wednesday, Dec. 4
- Project
  - Proposal due on Oct. 16; teams of 2-3
  - Project presentation on Dec. 9
  - Project report due on Dec. 10

Outline

- Regularization
  - Ridge and Lasso regression
- Classification
  - Linear classification
- K Nearest Neighbors (kNN)
  - Cross-validation for parameter selection
- Logistic regression
  - Classification based on probability
  - Maximum Likelihood Estimation
  - Cross-entropy loss

Gradient Descent vs Closed Form

Gradient Descent

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

simultaneous update
for j = 0 ... d

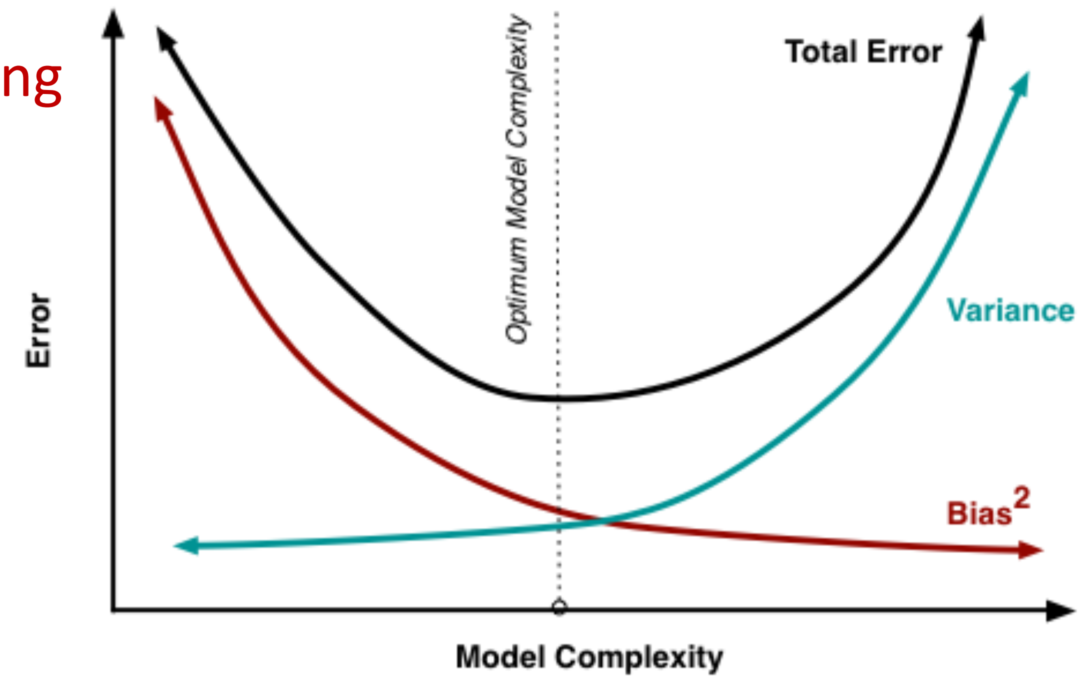
Closed form

$$\boldsymbol{\theta} = (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1} \boldsymbol{X}^\intercal \boldsymbol{y}$$

• Gradient Descent	• Closed Form
+ Linear increase in d and N	- Slow computation
+ Generally applicable	- Not generally applicable
- Choose α, stopping condition	+ No parameter tuning
- Might get stuck in local optima	+ Gives the global optimum

Bias-Variance Tradeoff

Under-fitting



Over-fitting

- Bias = Difference between estimated and true models
- Variance = Model difference on different training sets
- MSE is proportional to Bias^2 + Variance
- Regularization: general method to reduce model complexity

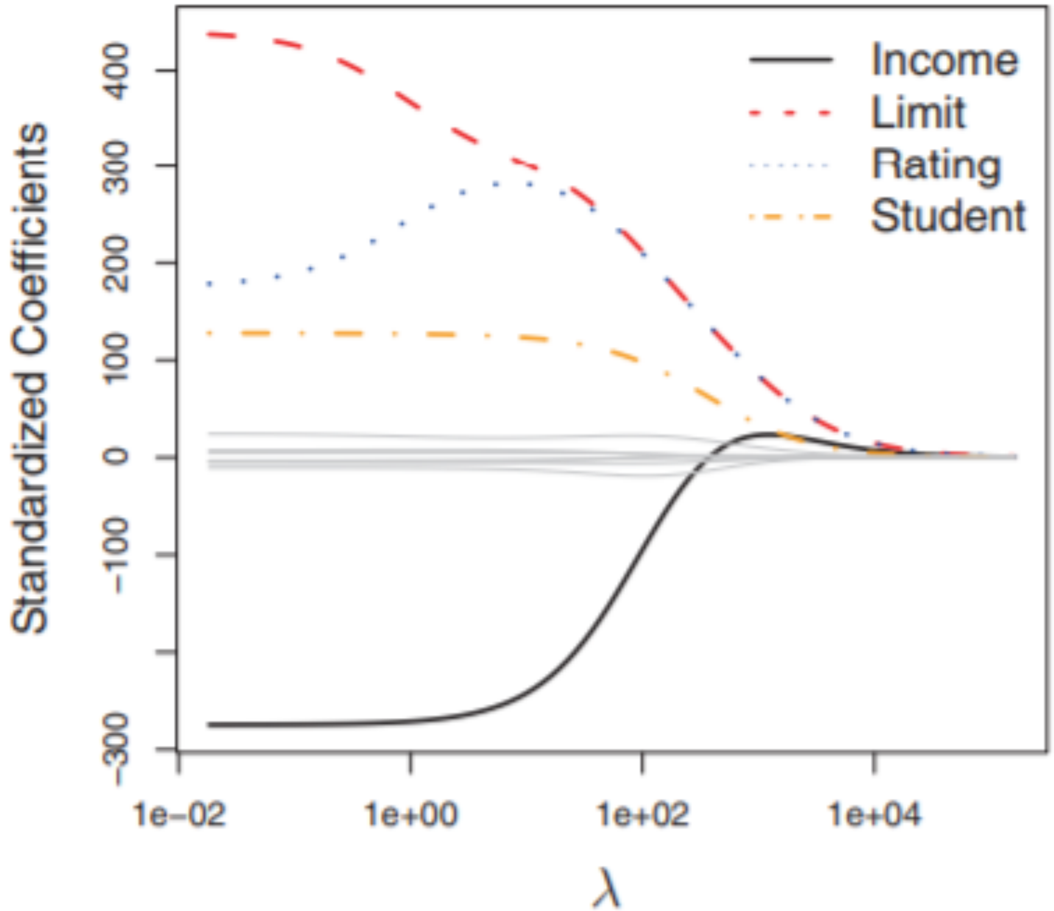
Ridge Regression

- Linear regression objective function

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

  - 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

Coefficient Shrinkage



- Example: Predict credit card balance
- Ridge is called “weight decay” in context of neural networks

Gradient Descent for Ridge Regression

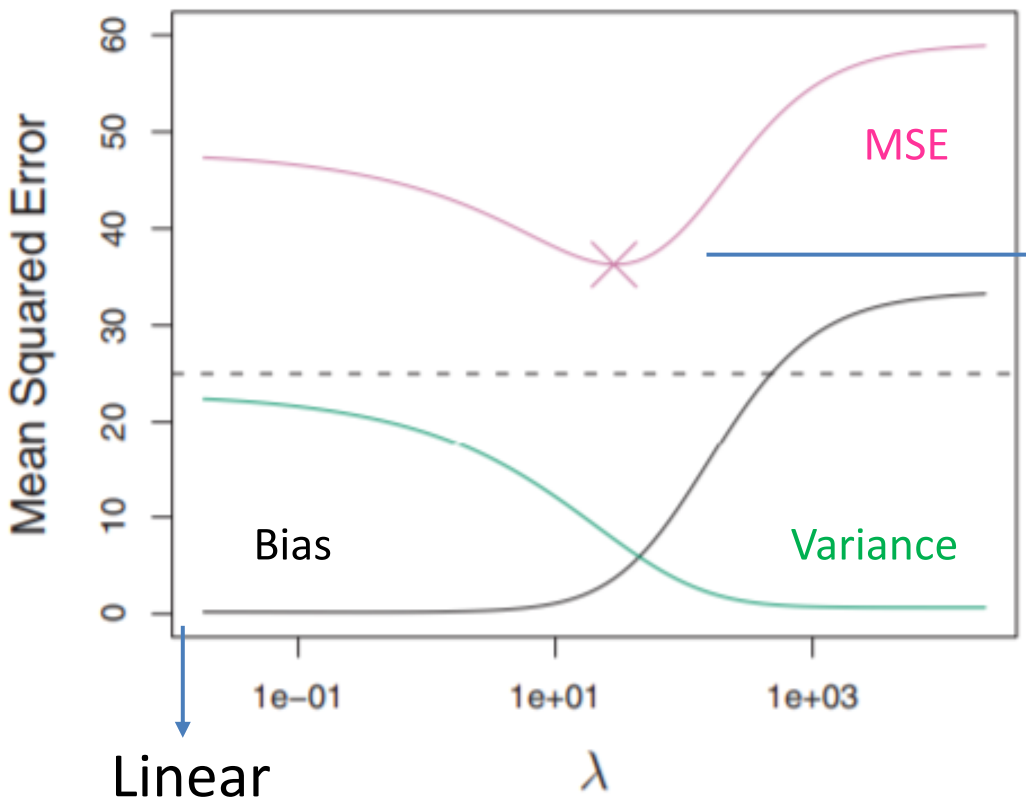
$$\boxed{J(\theta) = \frac{1}{2}\sum_{i=1}^{N} \left(h_\theta(x_i) - y_i \right)^2 + \frac{\lambda}{2}\sum_{j=1}^{d} \theta_i^2}$$

Gradient update: $\theta_0 \leftarrow \theta_0 - \alpha \sum_{i=1}^{N}(h_\theta(x_i) - y_i)$

$$\theta_j \leftarrow \theta_j - \alpha \sum_{i=1}^{N}(h_\theta(x_i) - y_i)x_{ij} - \underbrace{\alpha\lambda\theta_j}_{\text{Regularization}}$$

$$\boxed{\theta_j \leftarrow \theta_j(1 - \alpha\lambda) - \alpha \sum_{i=1}^{N}(h_\theta(x_i) - y_i)x_{ij}}$$

Bias-Variance Tradeoff



Optimal Ridge regression

Linear regression

Reduced model complexity

Lasso Regression

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

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

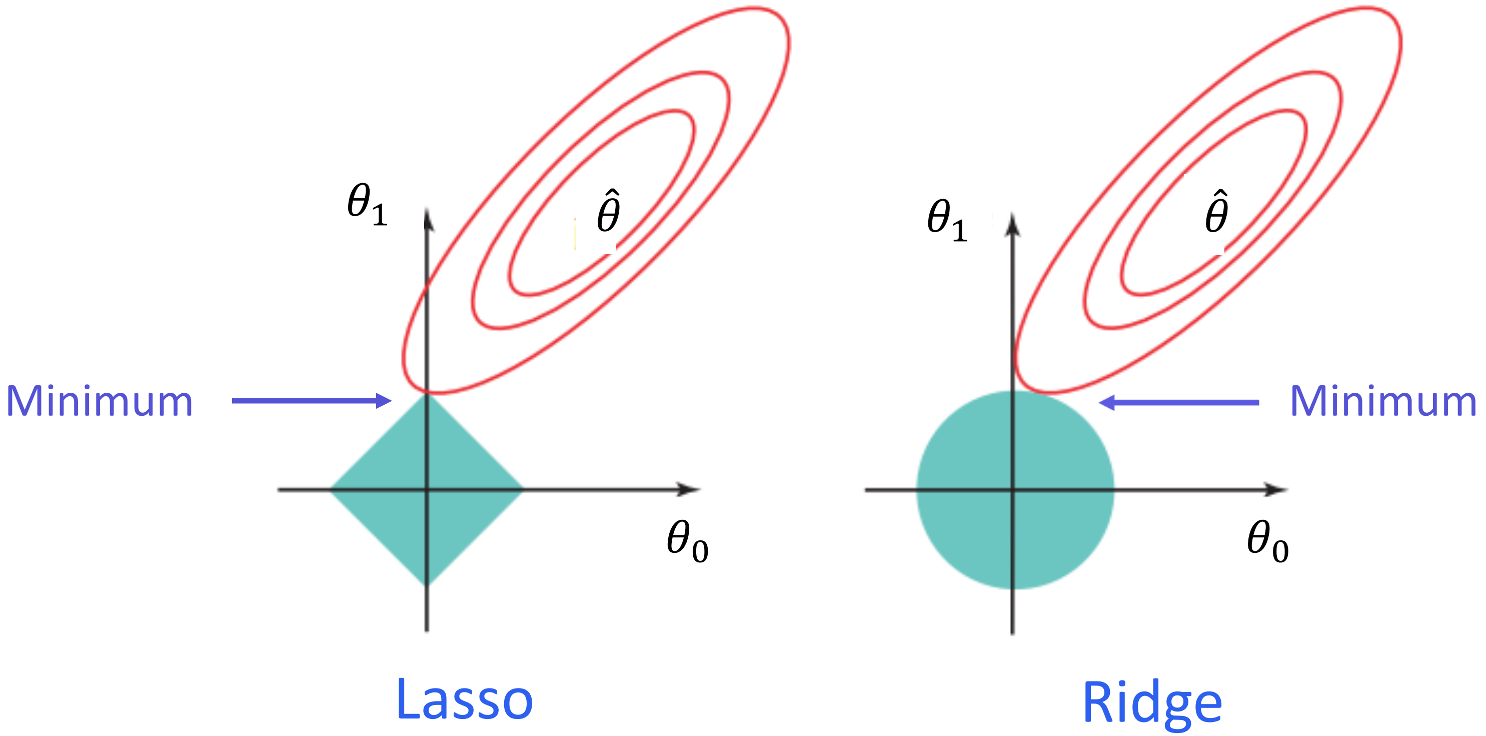
Alternative Formulations

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

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

Geometric Interpretation

- 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)
- 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

Supervised Learning

Problem Setting

- Set of possible instances $\mathcal{X}$
- Set of possible labels $\mathcal{Y}$
- Unknown target function $f : \mathcal{X} \to \mathcal{Y}$
- Set of function hypotheses $H = \{h \mid h : \mathcal{X} \to \mathcal{Y}\}$

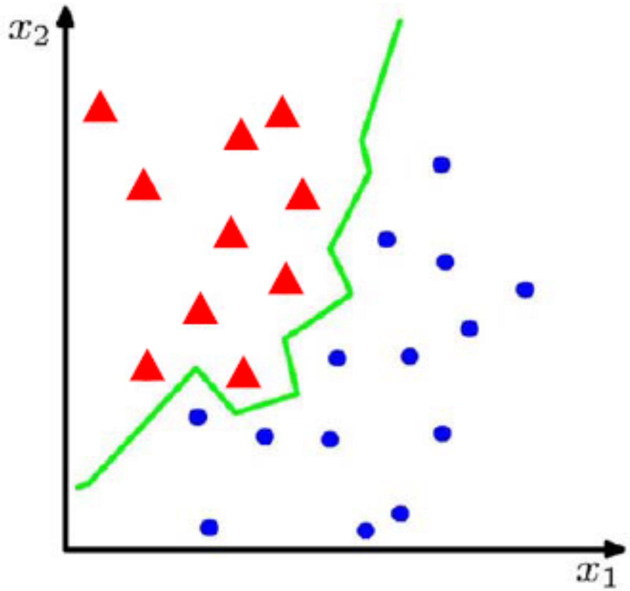
Input: Training examples of unknown target function f

$$\{x_i, y_i\}, \text{for } i = 1, \ldots, N$$

Output: Hypothesis $\hat{f} \in H$ that best approximates f

$$\hat{f}(x_i) \approx y_i$$

Classification



Binary or discrete

- Suppose we are given a training set of N observations

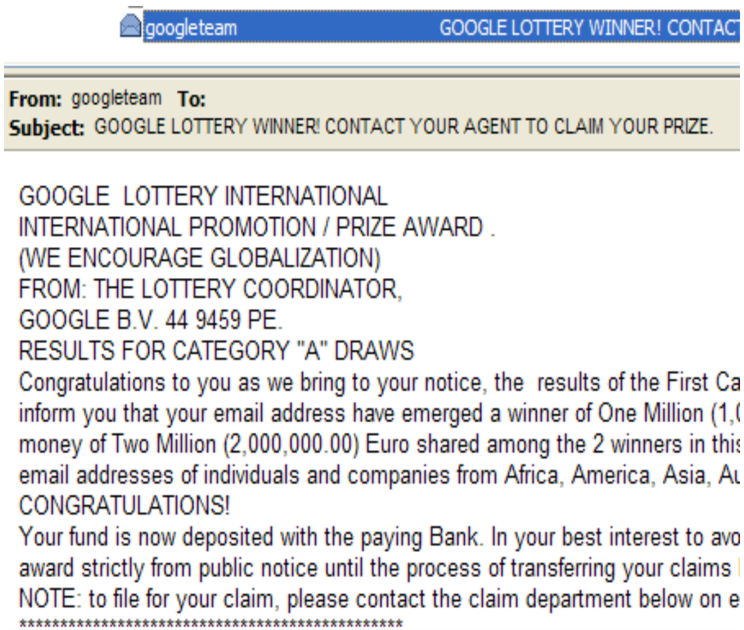
$$\{x_1, \dots, x_N\} \text{ and } \{y_1, \dots, y_N\}, x_i \in R^d, y_i \in \{0, 1\}$$

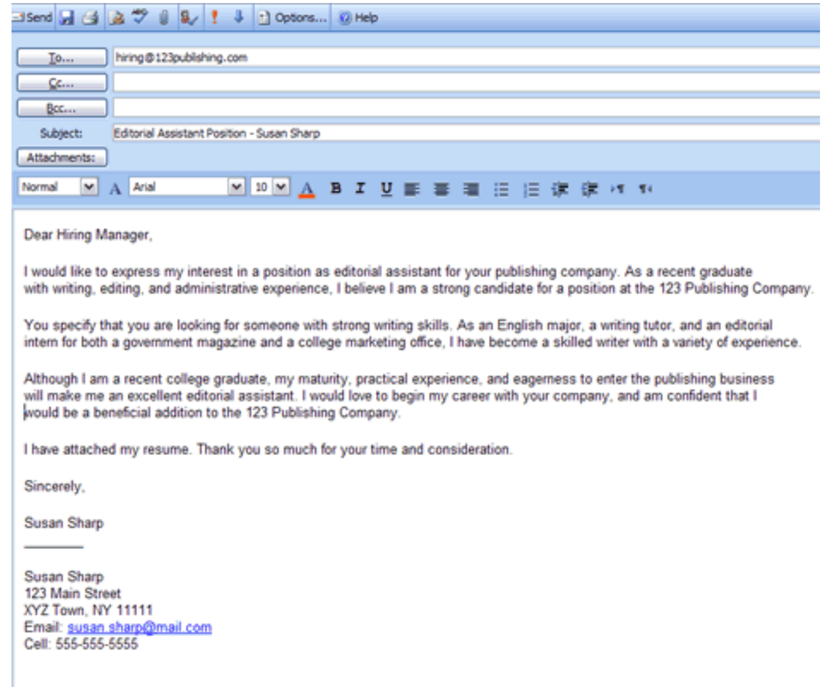
- Classification problem is to estimate f(x) from this data such that

$$f(x_i) = y_i$$

Example 1: Binary classification

Classifying spam email





Content-related features

- Use of certain words
- Word frequencies
- Language
- Sentence

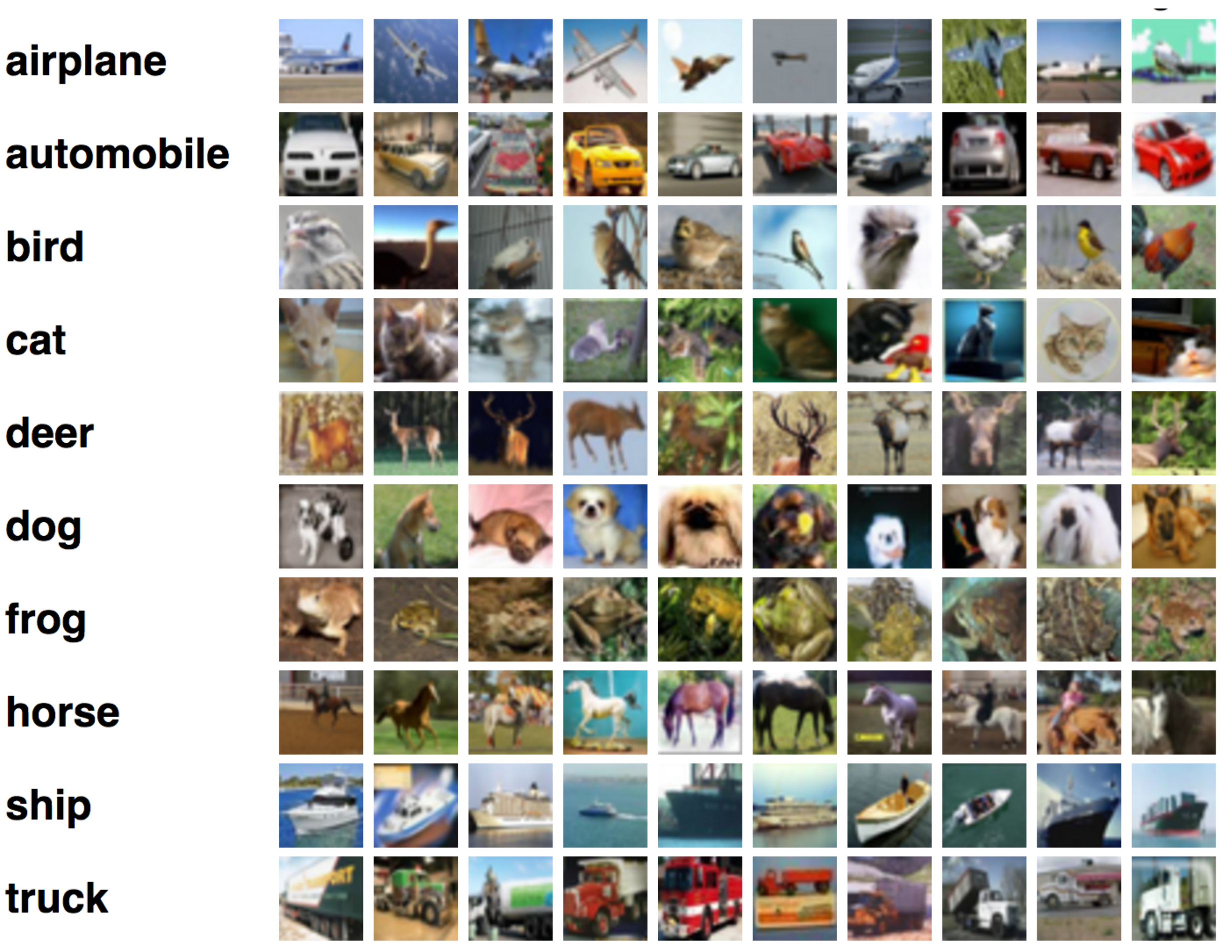
Structural features

- Sender IP address
- IP blacklist
- DNS information
- Email server
- URL links (non-matching)

Binary classification: SPAM or HAM

Example 2: Multi-class classification

Image classification



Multi-class classification

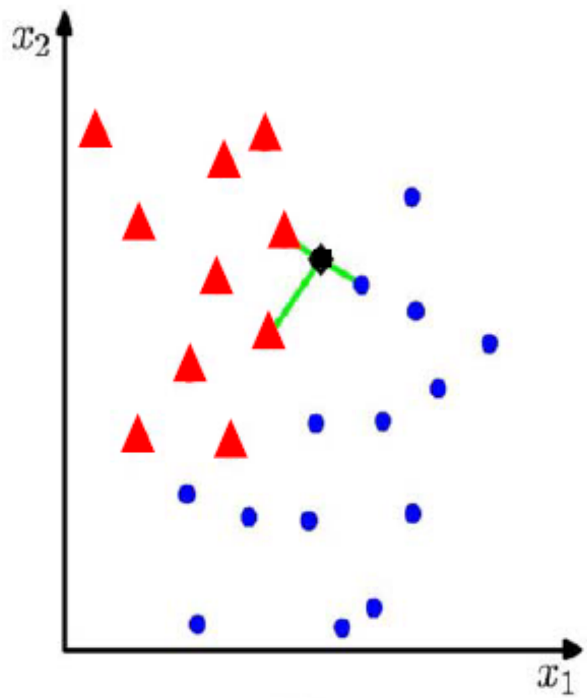
K Nearest Neighbour (K-NN) Classifier

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

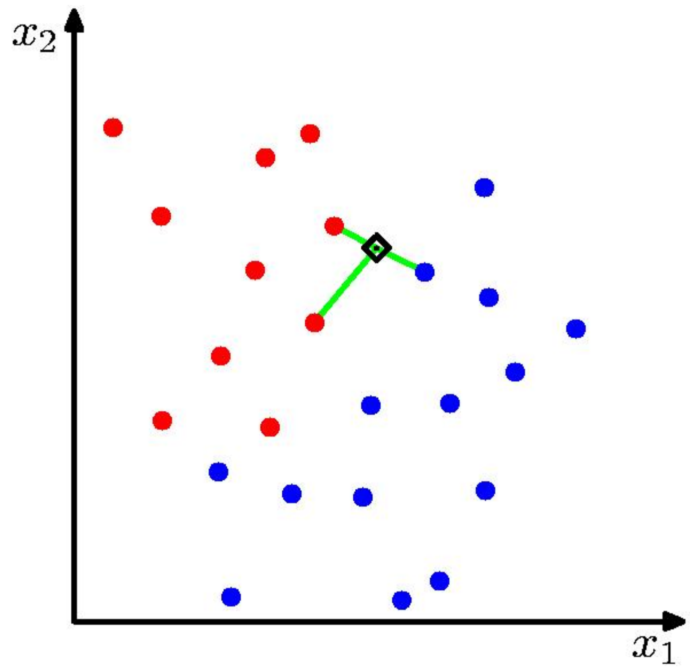
- applicable to multi-class case



Distance Metrics

- Euclidean Distance $$\sqrt{\left(\sum_{i=1}^{k}(x_i - y_i)^2\right)}$$
- Manhattan Distance $$\sum_{i=1}^{k}|x_i - y_i|$$
- Minkowski Distance $$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{\frac{1}{q}}$$

kNN

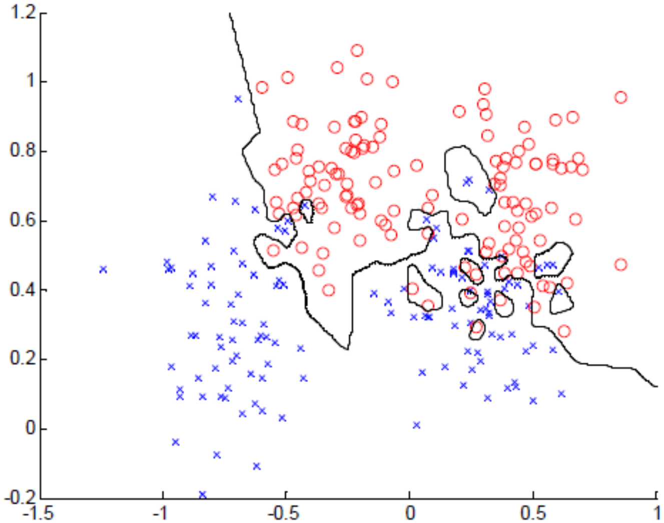


- Algorithm (to classify point x)
  - Find k nearest points to x (according to distance metric)
  - Perform majority voting to predict class of x
- Properties
  - Does not learn any model in training!
  - Instance learner (needs all data at testing time)

K = 1

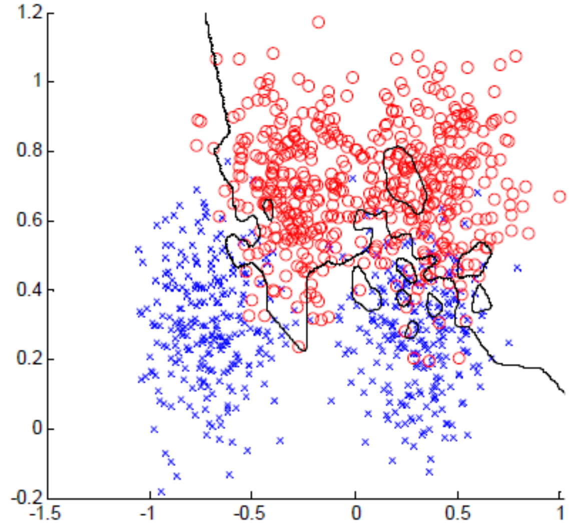
Overfitting!

Training data



error = 0.0

Testing data

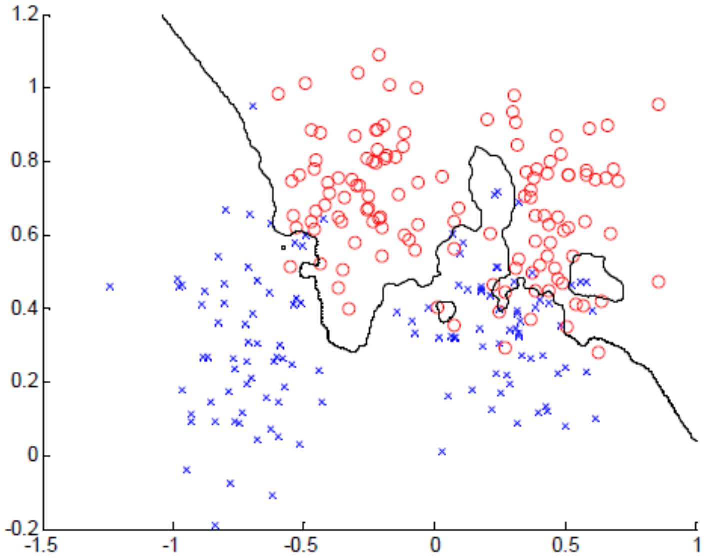


error = 0.15

How to choose k (hyper-parameter)?

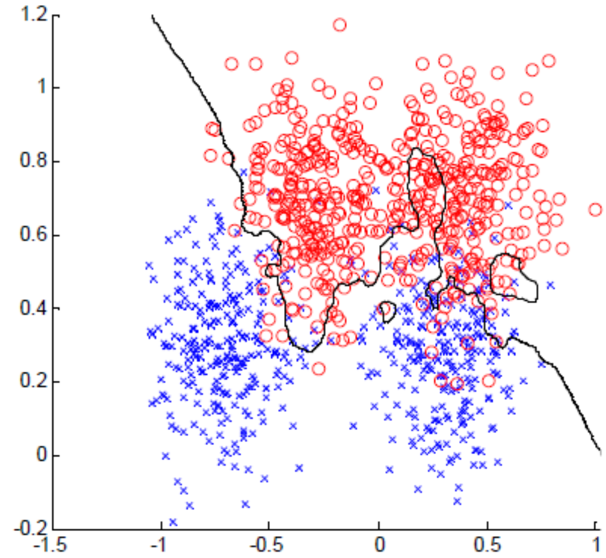
K = 3

Training data



error = 0.0760

Testing data

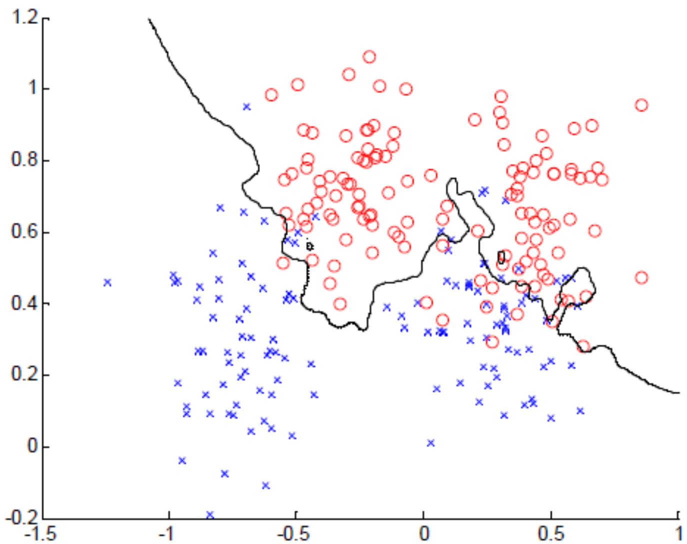


error = 0.1340

How to choose k (hyper-parameter)?

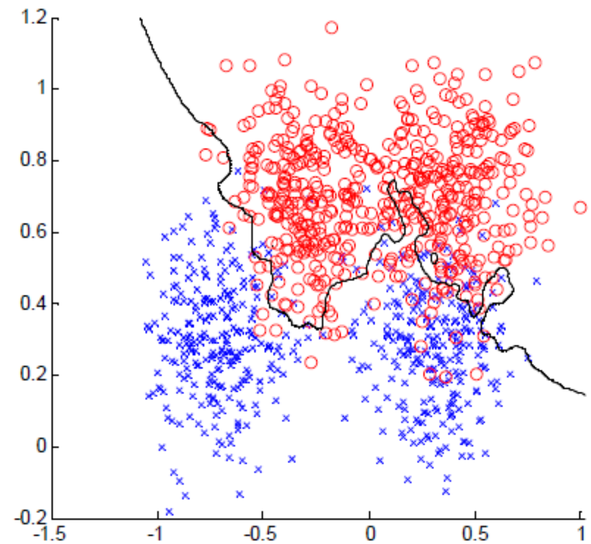
K = 7

Training data



error = 0.1320

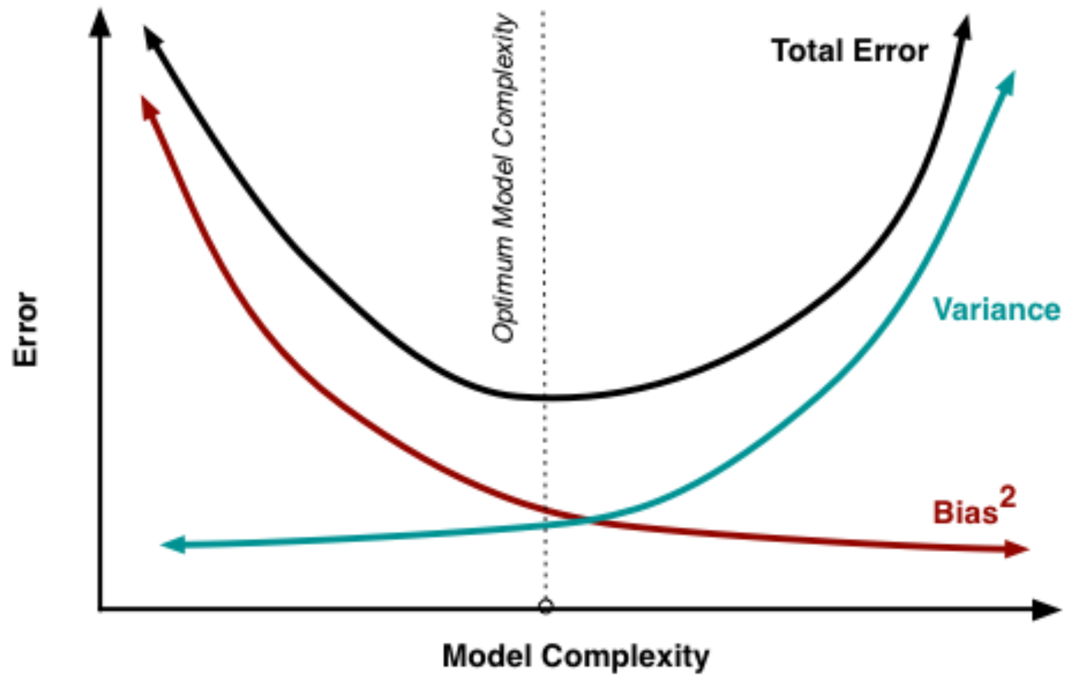
Testing data



error = 0.1110

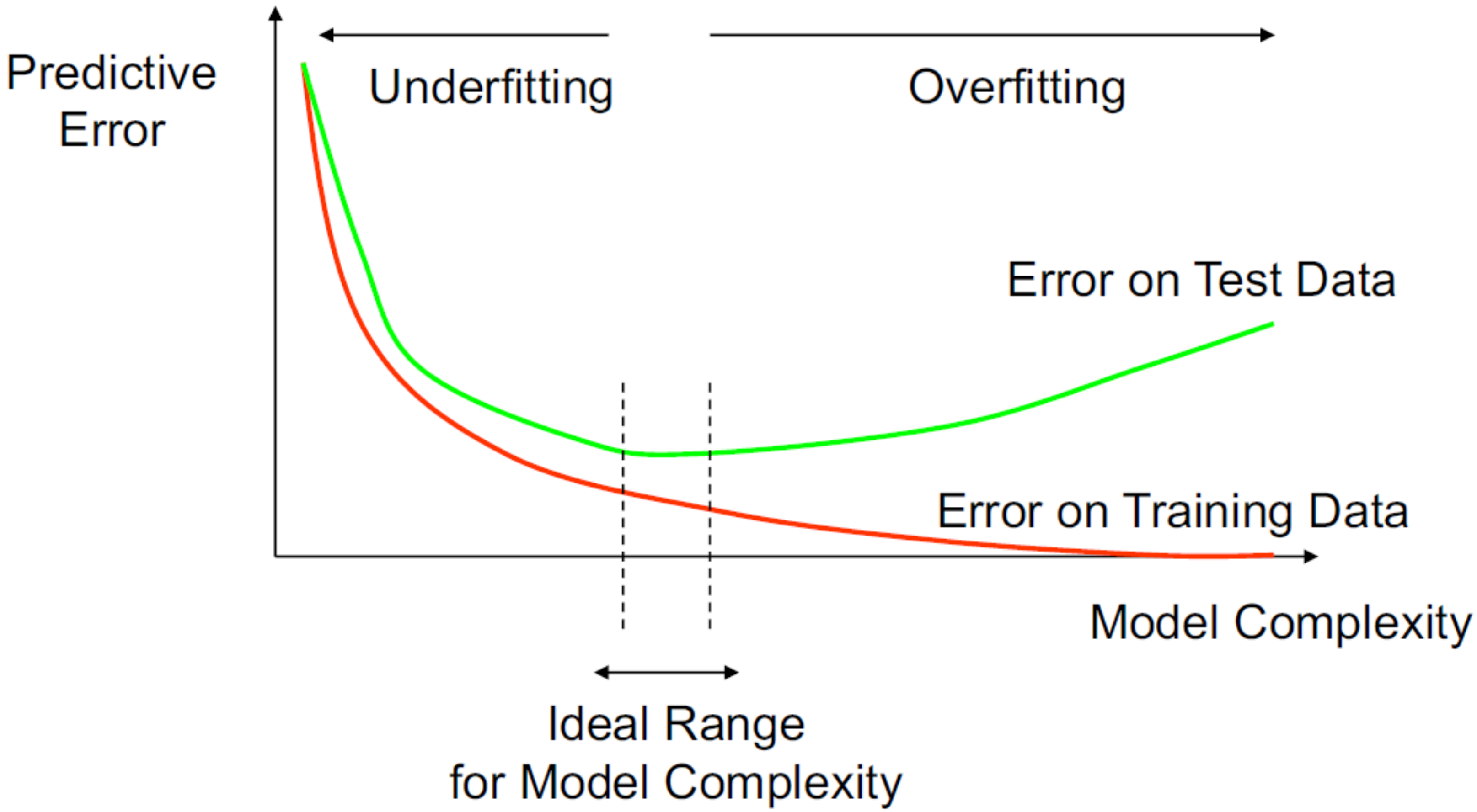
How to choose k (hyper-parameter)?

Bias-Variance Tradeoff for kNN



K decreases

How Overfitting Affects Prediction



How can we avoid over-fitting without having access to testing data?

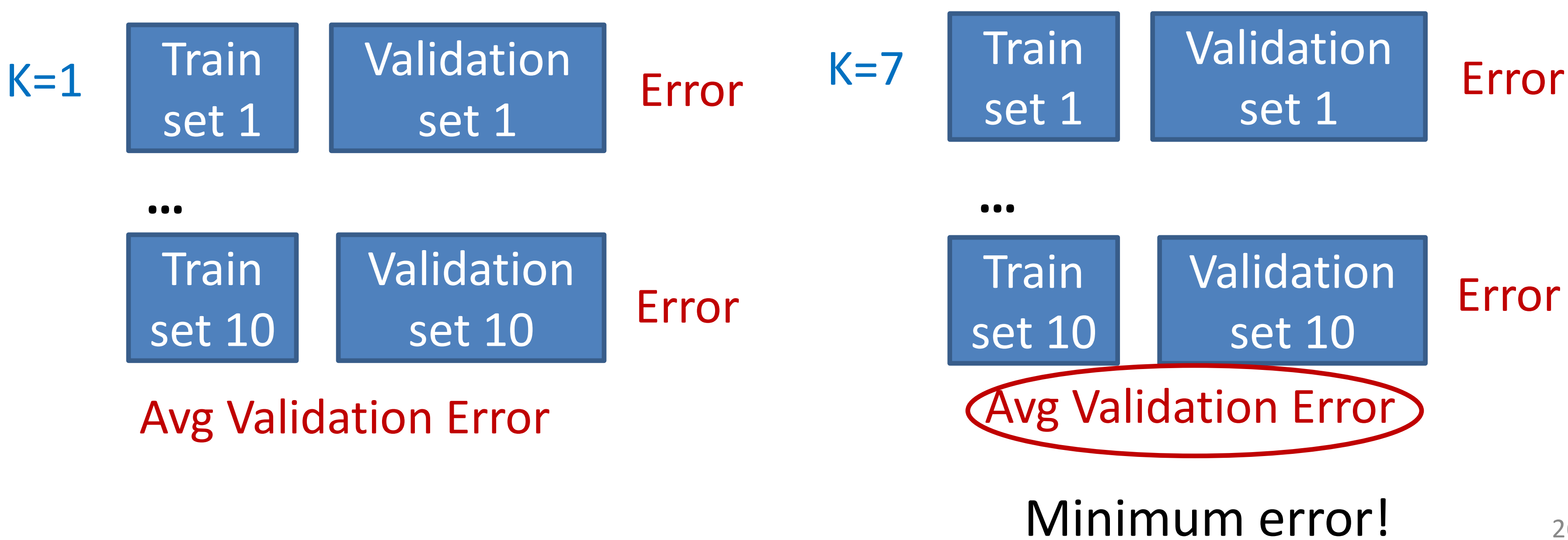
Cross Validation

As K increases:

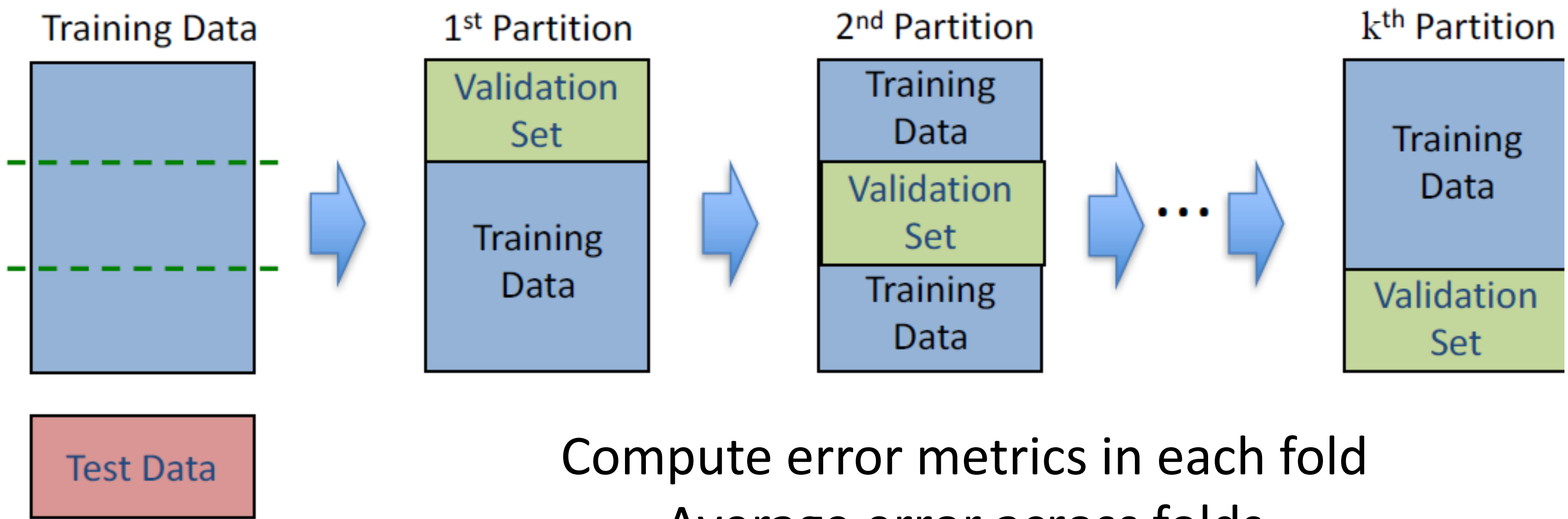
- Classification boundary becomes smoother
- Training error can increase

Choose (learn) K by cross-validation

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



Cross Validation

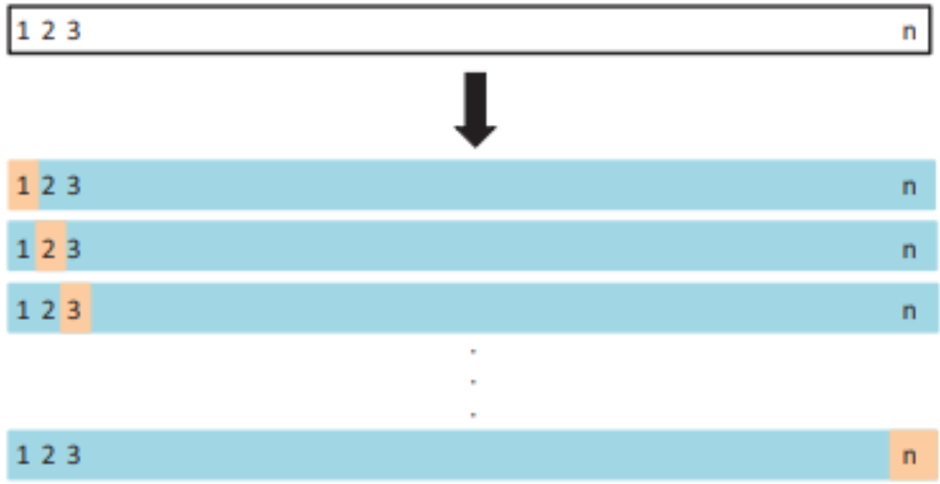


Compute error metrics in each fold
Average error across folds

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



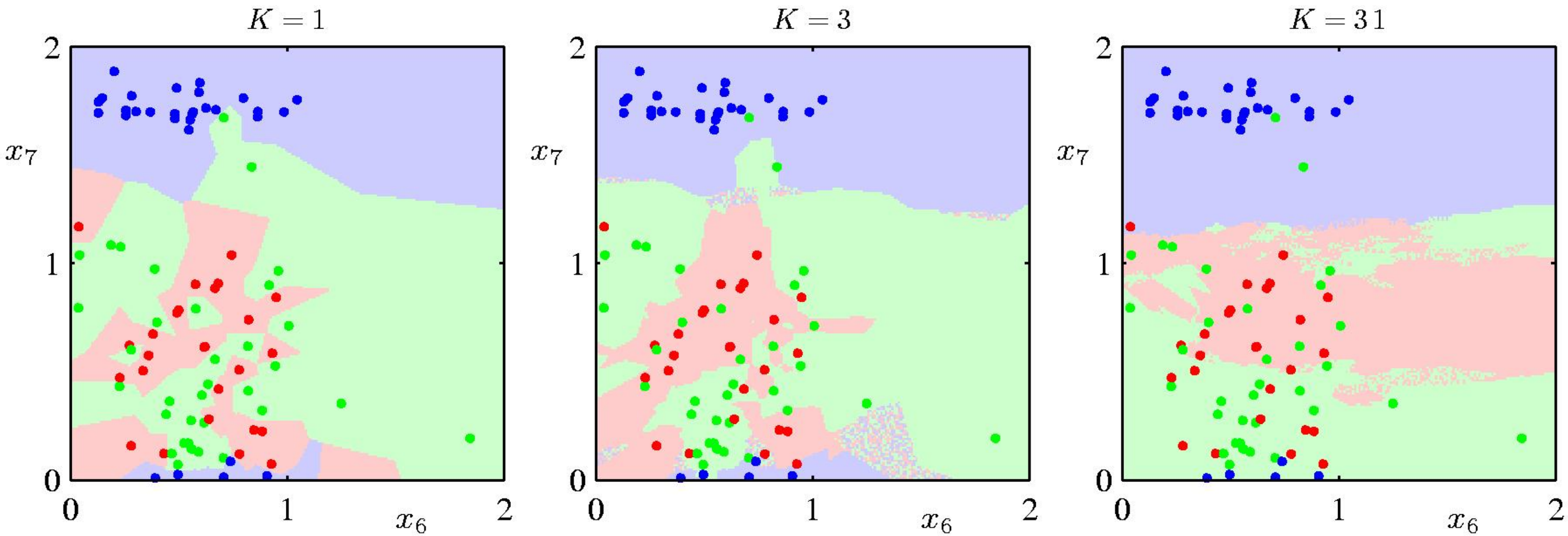
2. Leave-one-out CV (LOOCV)
   – k=n (validation set only one point)

- Pros: Less bias
- Cons: More expensive to implement, higher variance
- 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
  - Regularization works when training with GD
  - Cross-validation can be used for hyper-parameter selection
  - The two methods can be combined

K-Nearest-Neighbours for Multi-class Classification

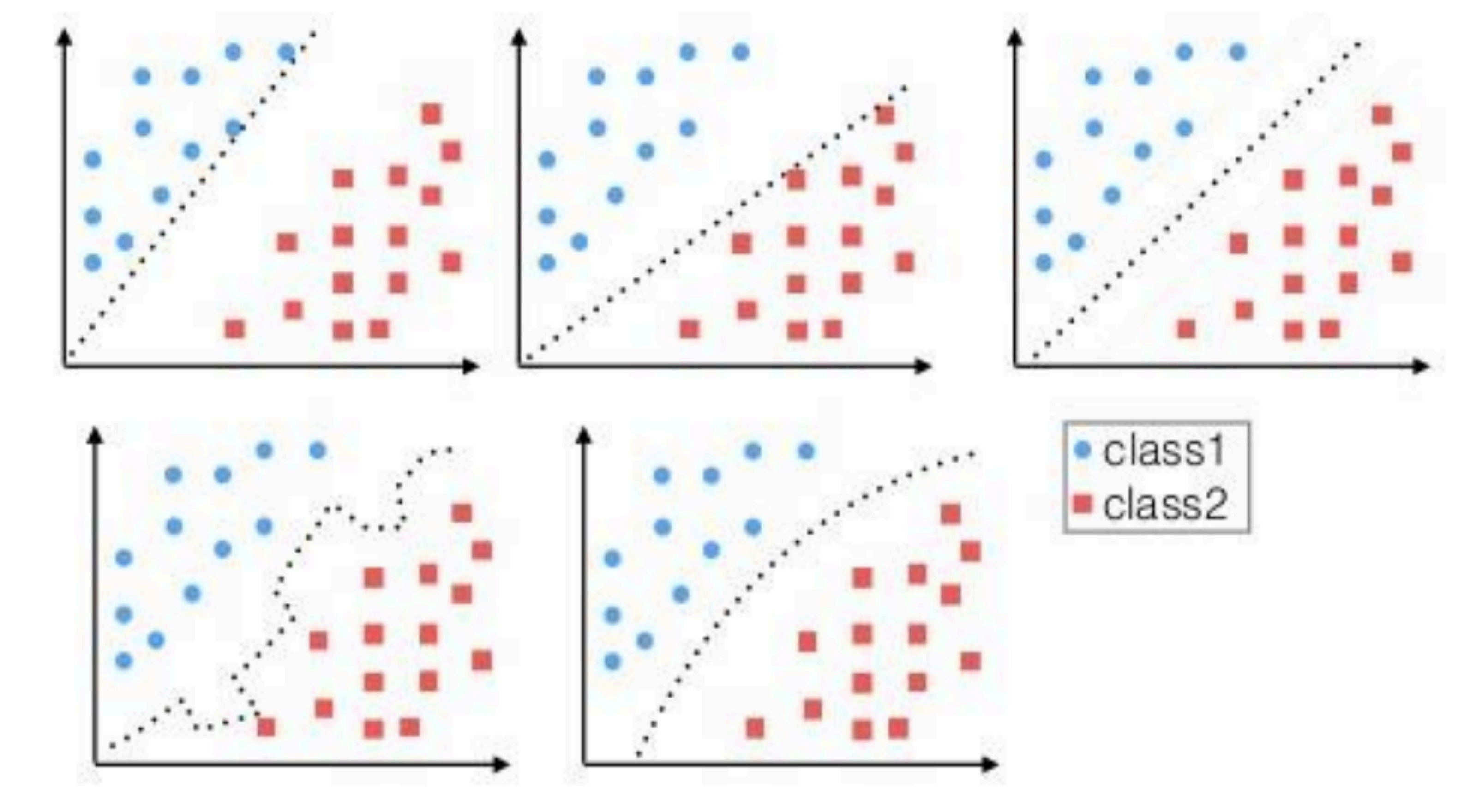


Vote among multiple classes

Outline

- Regularization
  - Ridge and Lasso regression
- Classification
  - Linear classification
- K Nearest Neighbors (kNN)
  - Cross-validation for parameter selection
- Logistic regression
  - Classification based on probability
  - Maximum Likelihood Estimation
  - Cross-entropy loss

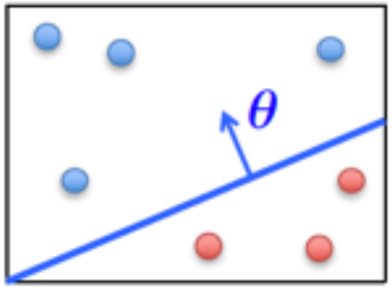
Linear vs Non-Linear Classifiers



Linear Classifiers

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

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



$h_\theta(x) = f(\theta^T x)$ linear function
- If $\theta^T x > 0$ classify 1
- If $\theta^T x < 0$ classify 0

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

Linear Classifiers

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

- Examples: perceptron, LDA
- Pros
  
  - Very compact model (size *d*)
- Cons of linear models studied so far
  - Perceptron depend on the order of training data and it could take many steps for convergence
  
  - LDA assumes normal distribution of features

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

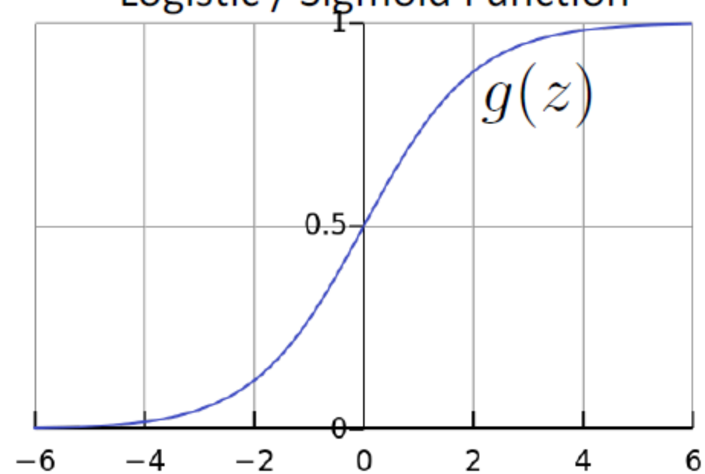
Logistic Regression

- 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:

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

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

Logistic / Sigmoid Function



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

LR is a Linear Classifier!

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

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

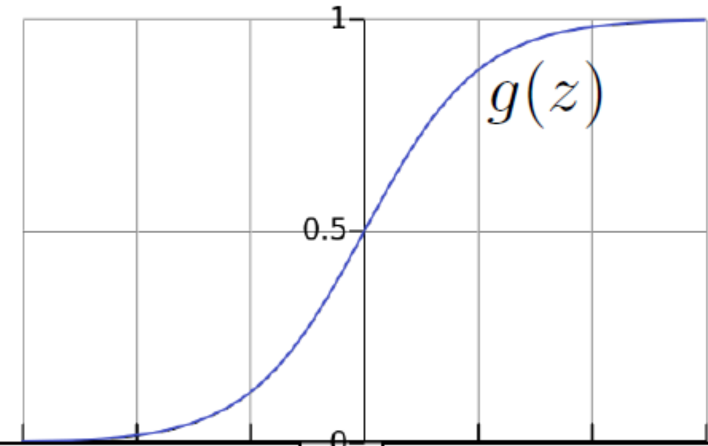
- Equivalent to:
  - $e^{\theta_0 + \sum_{j=1}^{d} \theta_j x_j} > 1$
  - $\theta_0 + \sum_{j=1}^{d} \theta_j x_j > 0$

Logistic Regression is a linear classifier!

Logistic Regression

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

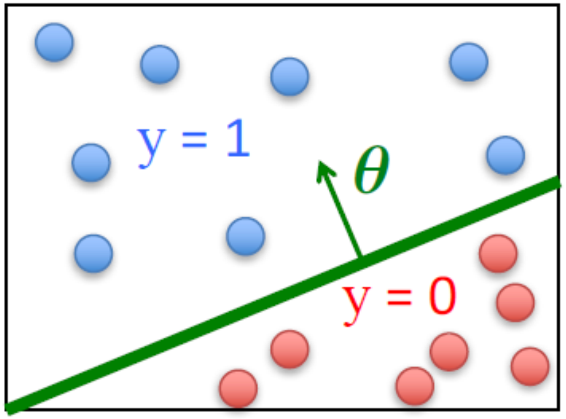
$$g(z) = \frac{1}{1+e^{-z}}$$



$\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}$ should be large <u>negative</u> values for negative instances

$\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}$ should be large <u>positive</u> 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!

Logistic Regression Objective

- Can't just use squared loss as in linear regression:

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N}(h_\theta(x_i) - y_i)^2$$

  - Using the logistic regression model

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

  results in a non-convex optimization

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 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

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

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

- They both have the same maximum θ_{MLE}

MLE for Logistic Regression

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

$$\theta_{MLE} = \mathrm{argmax}_\theta \sum_{i=1}^{N} \log P[Y = y_i | X = x_i; \theta]$$
$$= \mathrm{argmax}_\theta \sum_{i=1}^{N} y_\mathrm{i} \log h_\theta(x_i) + (1 - y_i) \log\left(1 - h_\theta(x_i)\right)$$

Logistic regression objective

$$\min_\theta J(\theta)$$

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

Cross-Entropy Objective

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

- Cost of a single instance:

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

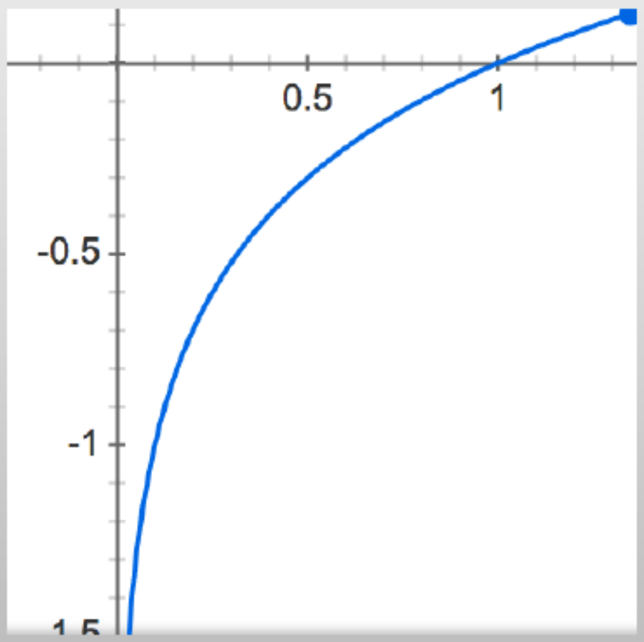
- Can re-write objective function as

$$J(\boldsymbol{\theta}) = \sum_{i=1}^{n} \underbrace{\text{cost}\left(h_\theta(x_i), y_i \right)}_{\text{Cross-entropy loss}}$$

Intuition

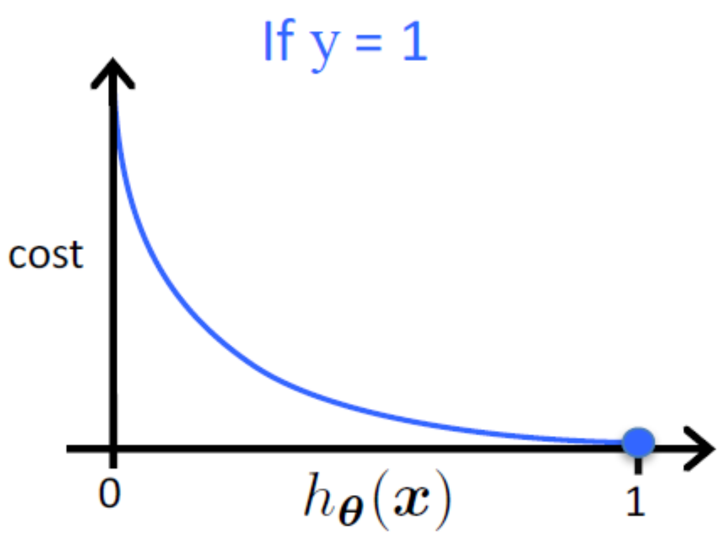
$$\text{cost}\,(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y) = \begin{cases} -\log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 1 \\ -\log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 0 \end{cases}$$

Aside: Recall the plot of log(z)



Intuition

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

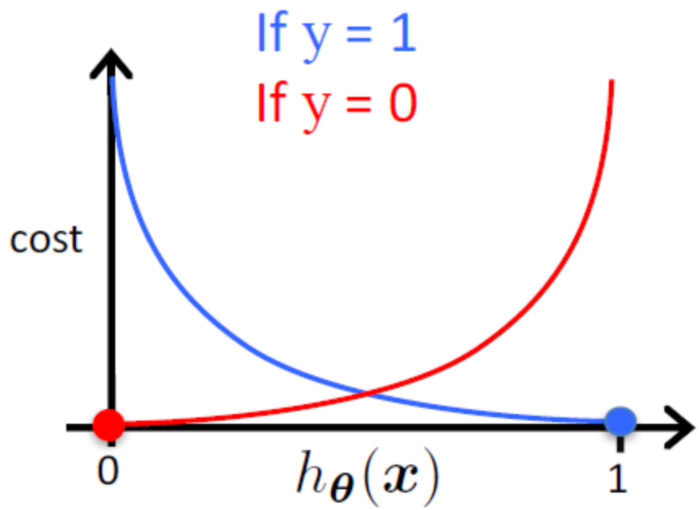


If y = 1

- 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

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



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

Review

- Regularization is a general method to avoid over-fitting
- Cross-validation should be performed to
  – Improve model generalization
  – Avoid over-fitting
  – Choose hyper parameters (k in kNN)
- Logistic regression is a linear classifier that predicts class probability
  – MLE objective: Cross-entropy loss
  – Can be trained with Gradient Descent: next time

Acknowledgements

- Slides made using resources from:
  - Andrew Ng
  - Eric Eaton
  - David Sontag
- Thanks!