DS 4400

Machine Learning and Data Mining I

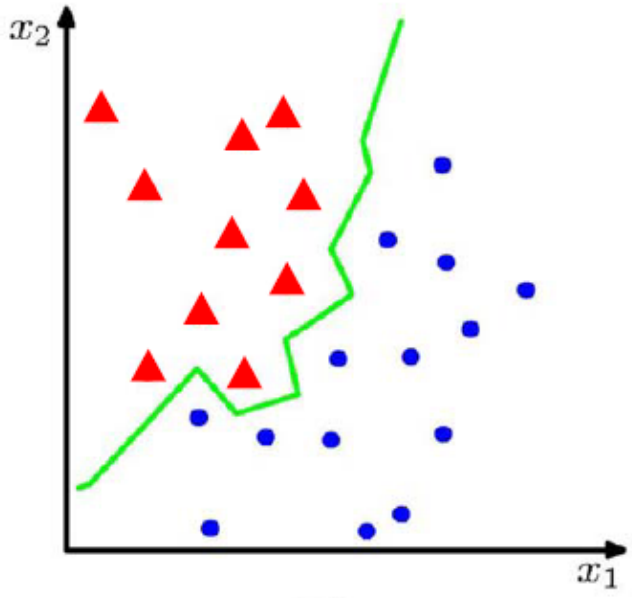
Alina Oprea
Associate Professor
Khoury College of Computer Science
Northeastern University

October 6 2020

Outline

- Classification
  - K Nearest Neighbors (kNN)
  - Cross-validation
- Linear classifiers
- Perceptron
  - Online and batch perceptron
- Logistic regression

Classification



Binary or discrete

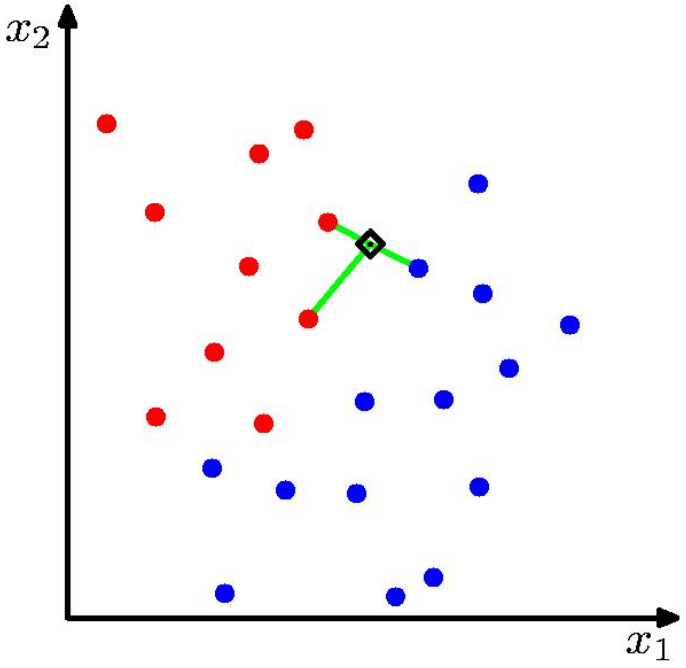
- Suppose we are given a training set of N observations

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

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

$$f(x_i) = y_i$$

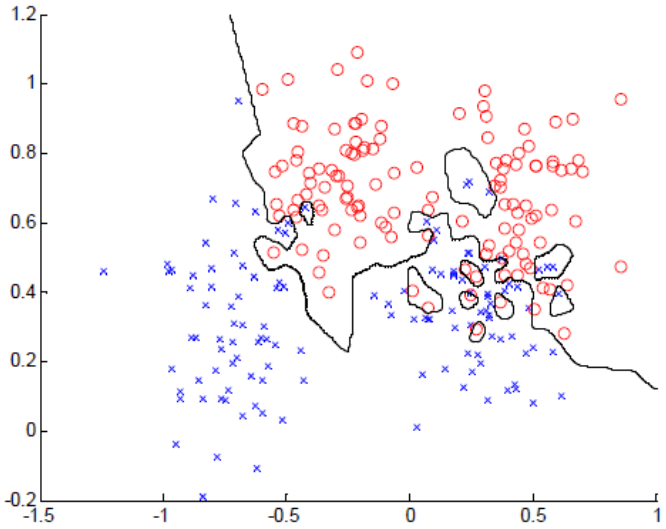
kNN



- 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)

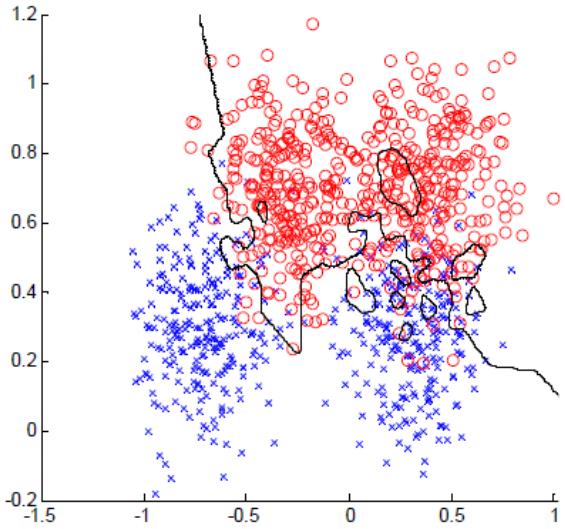
K = 1

Training data



error = 0.0

Testing data

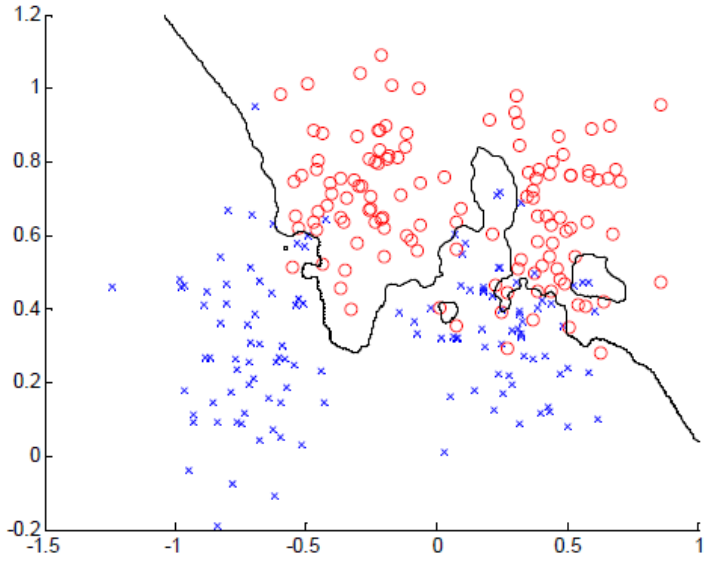


error = 0.15

How to choose k (hyper-parameter)?

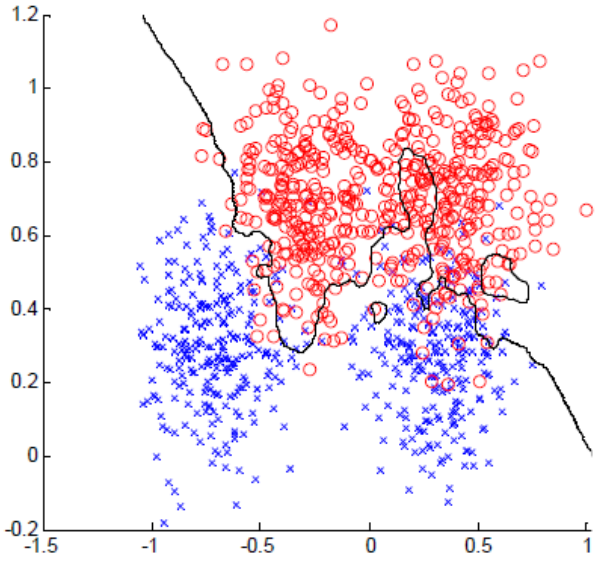
K = 3

Training data



error = 0.0760

Testing data

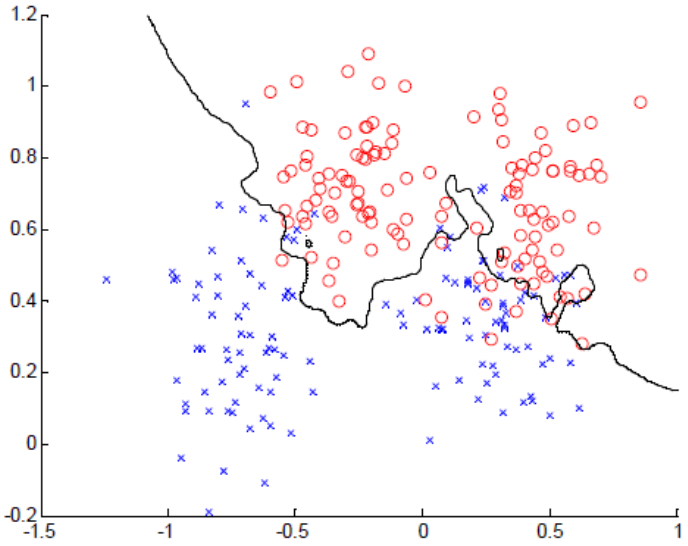


error = 0.1340

How to choose k (hyper-parameter)?

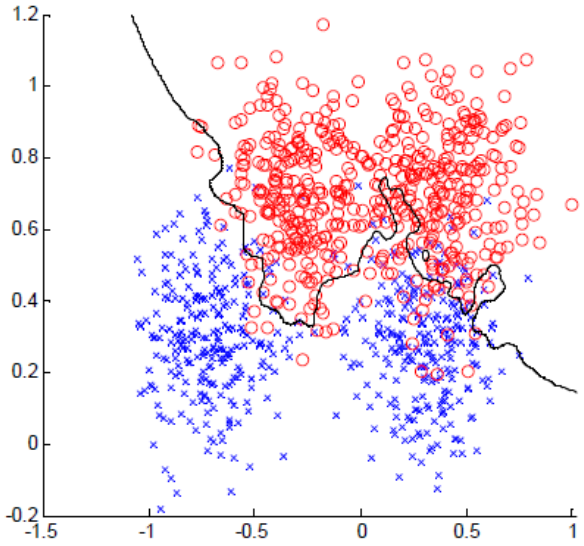
K = 7

Training data



error = 0.1320

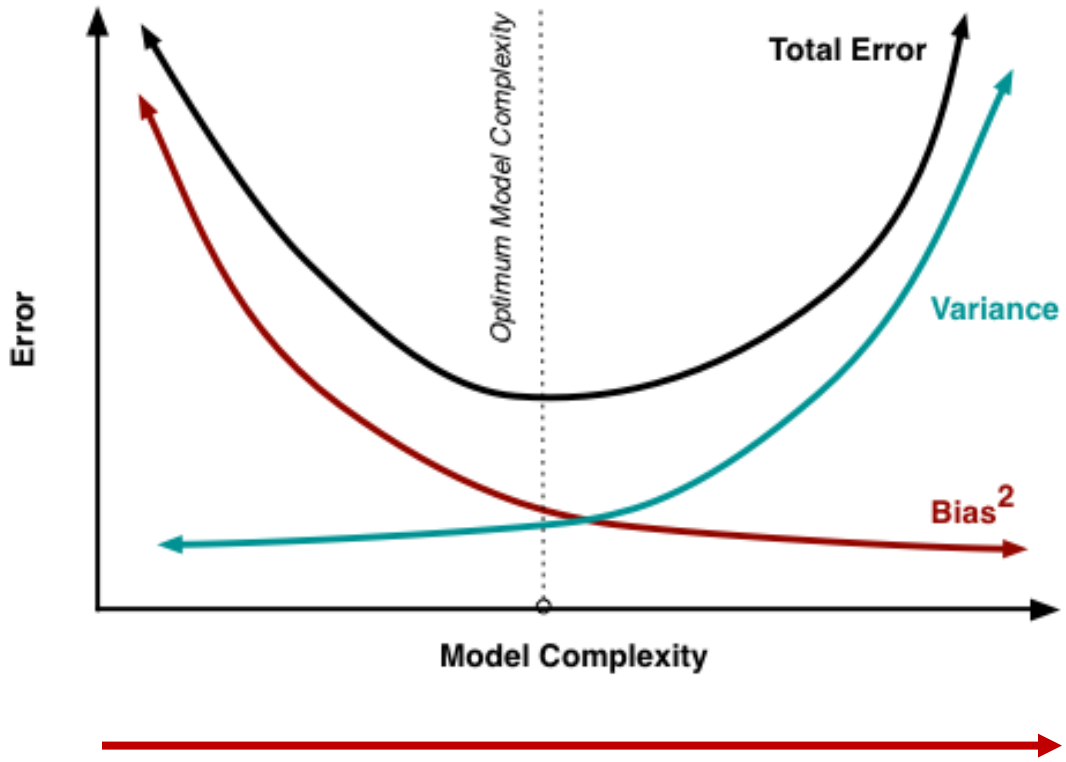
Testing data



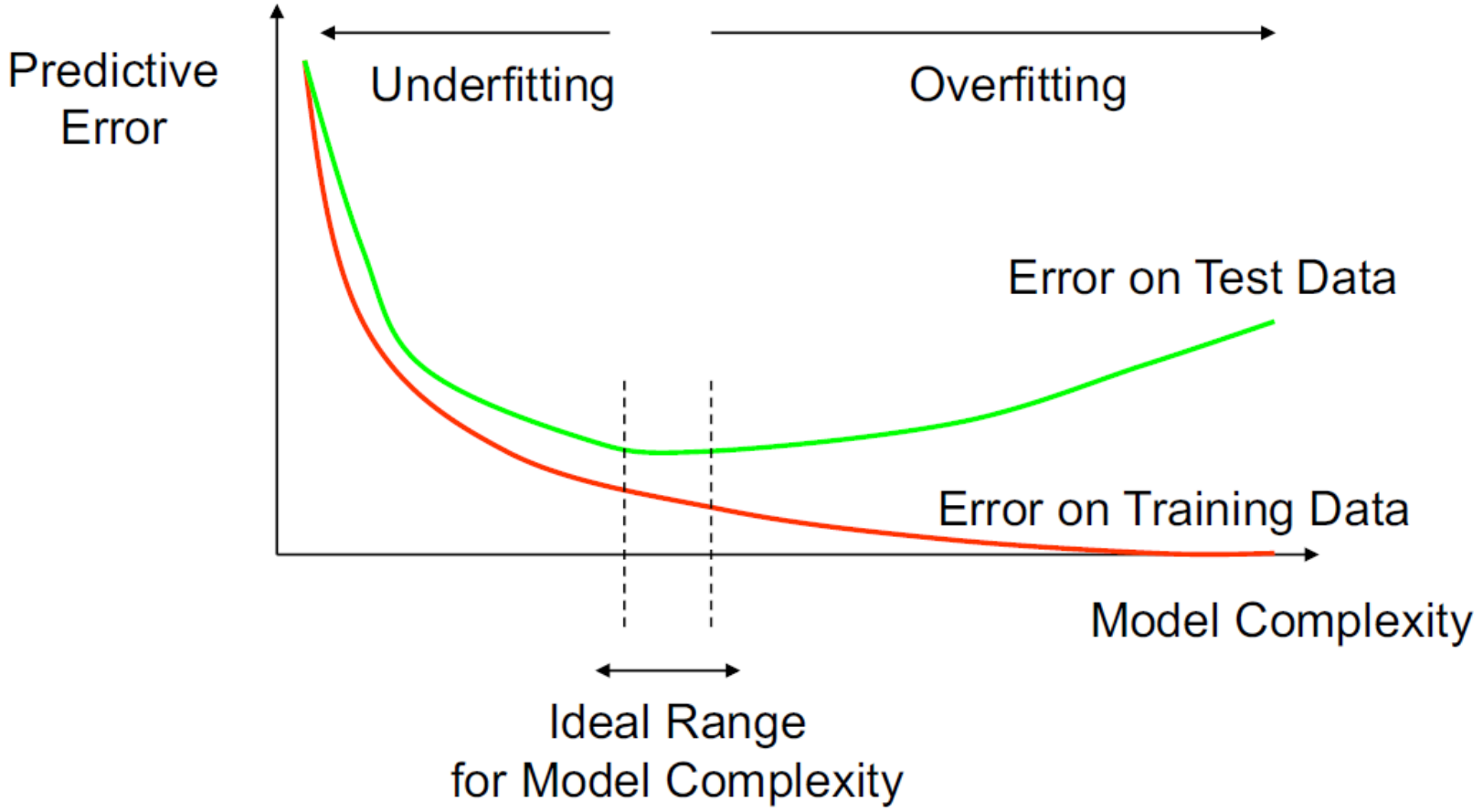
error = 0.1110

How to choose k (hyper-parameter)?

Bias-Variance Tradeoff for kNN



How Overfitting Affects Prediction



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

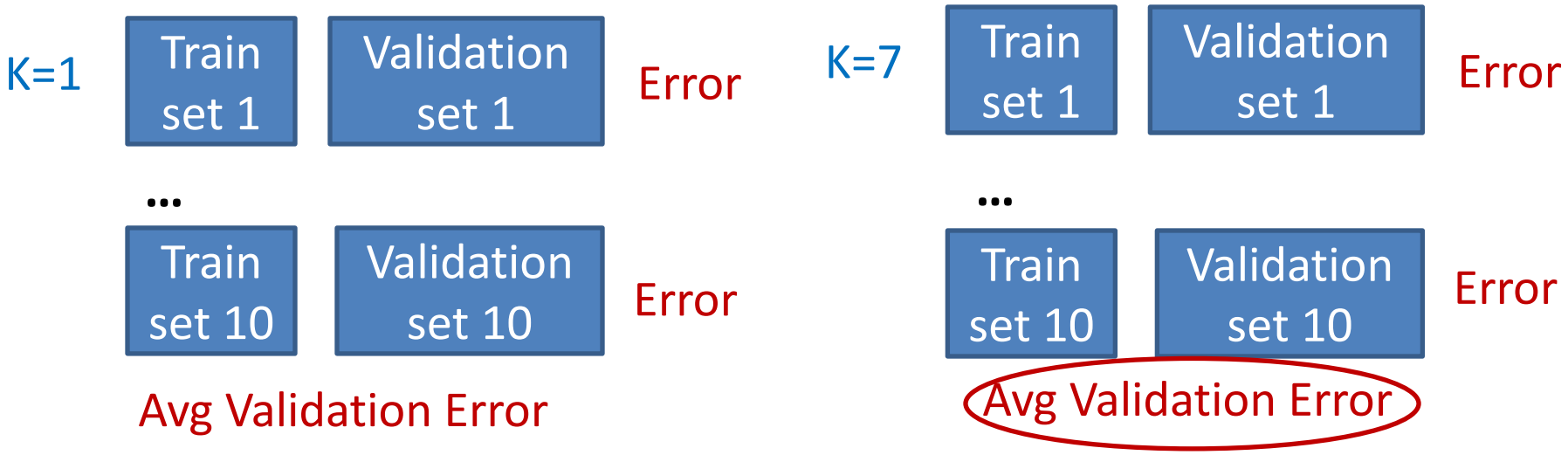
Cross Validation

As K increases:

- Classification boundary becomes smoother
- Training error can increase

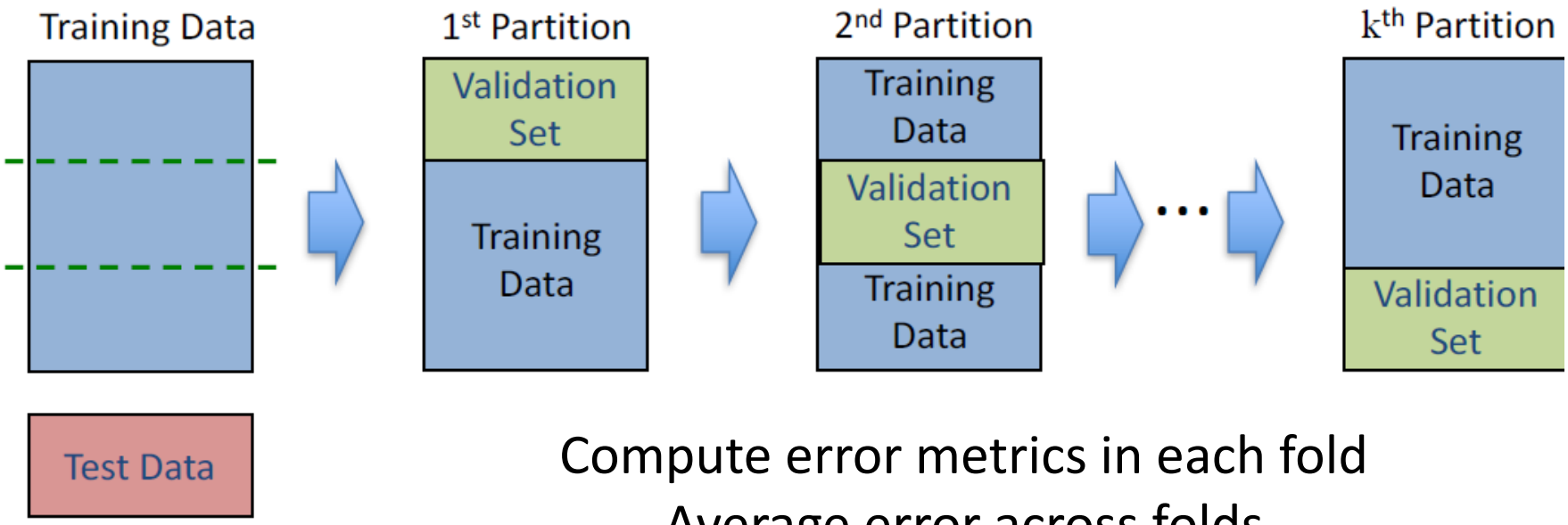
Choose (learn) K by cross-validation

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



Minimum error!

Cross Validation

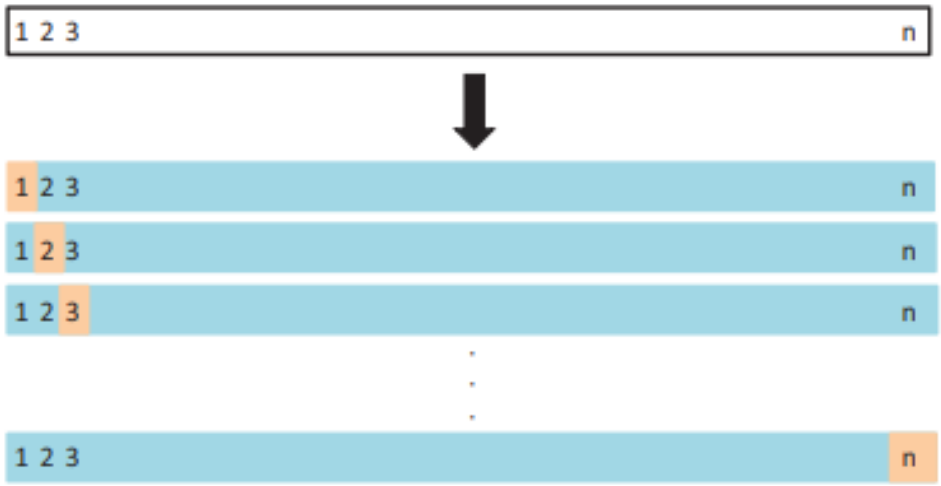


Compute error metrics in each fold
Average error across folds

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

Outline

- Classification
  - K Nearest Neighbors (kNN)
  - Cross-validation
- Linear classifiers
- Perceptron
  - Online and batch perceptron
- Logistic regression

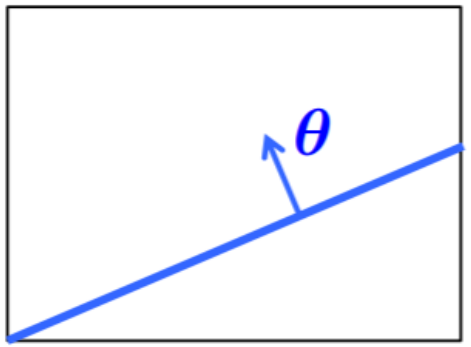
History of Perceptrons

- They were popularised by Frank Rosenblatt in the early 1960's.
  - They appeared to have a very powerful learning algorithm.
  - Lots of grand claims were made for what they could learn to do.
- In 1969, Minsky and Papert published a book called "Perceptrons" that analysed what they could do and showed their limitations.
  - Many people thought these limitations applied to all neural network models.
- The perceptron learning procedure is still widely used today for tasks with enormous feature vectors that contain many millions of features.

They are the basic building blocks for
Deep Neural Networks

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



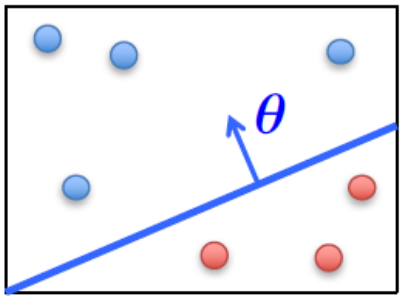
  - Assumed to pass through the origin
    - This is because we incorporated bias term θ_0 into it by $x_0 = 1$

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

Linear classifiers

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

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



$$h(\boldsymbol{x}) = \text{sign}(\boldsymbol{\theta}^\intercal \boldsymbol{x}) \text{ where } \quad \text{sign}(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases}$$

– Note that: $\boldsymbol{\theta}^\intercal \boldsymbol{x} > 0 \implies y = +1$

$\boldsymbol{\theta}^\intercal \boldsymbol{x} < 0 \implies y = -1$

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

Example: Spam

- Imagine 3 features (spam is "positive" class):
  1. free (number of occurrences of "free")
  2. money (occurrences of "money")
  3. BIAS (intercept, always has value 1)

$$\sum_{i=0}^{d} x_i \theta_i$$

"free money"

x	
BIAS	1
free	1
money	1
...	

θ	
BIAS	-3
free	4
money	2
...	

$$(1)(-3) + (1)(4) + (1)(2) + \ldots = 3$$

$\sum_i x_i \theta_i > 0$ ➔ SPAM!!!

The Perceptron

$$h(\boldsymbol{x}) = \text{sign}(\boldsymbol{\theta}^\intercal \boldsymbol{x}) \quad \text{where} \quad \text{sign}(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases}$$

- The perceptron uses the following update rule each time it receives a new training instance (x_i, y_i)

$$\theta_j \leftarrow \theta_j - \frac{1}{2}(h_\theta(x_i) - y_i)x_{ij}$$

either 2 or -2

  - If the prediction matches the label, make no change
  - Otherwise, adjust $\boldsymbol{\theta}$

The Perceptron

- The perceptron uses the following update rule each time it receives a new training instance (x_i, y_i)

$$\theta_j \leftarrow \theta_j - \frac{1}{2}\,(h_\theta(x_i) - y_i)x_{ij}$$

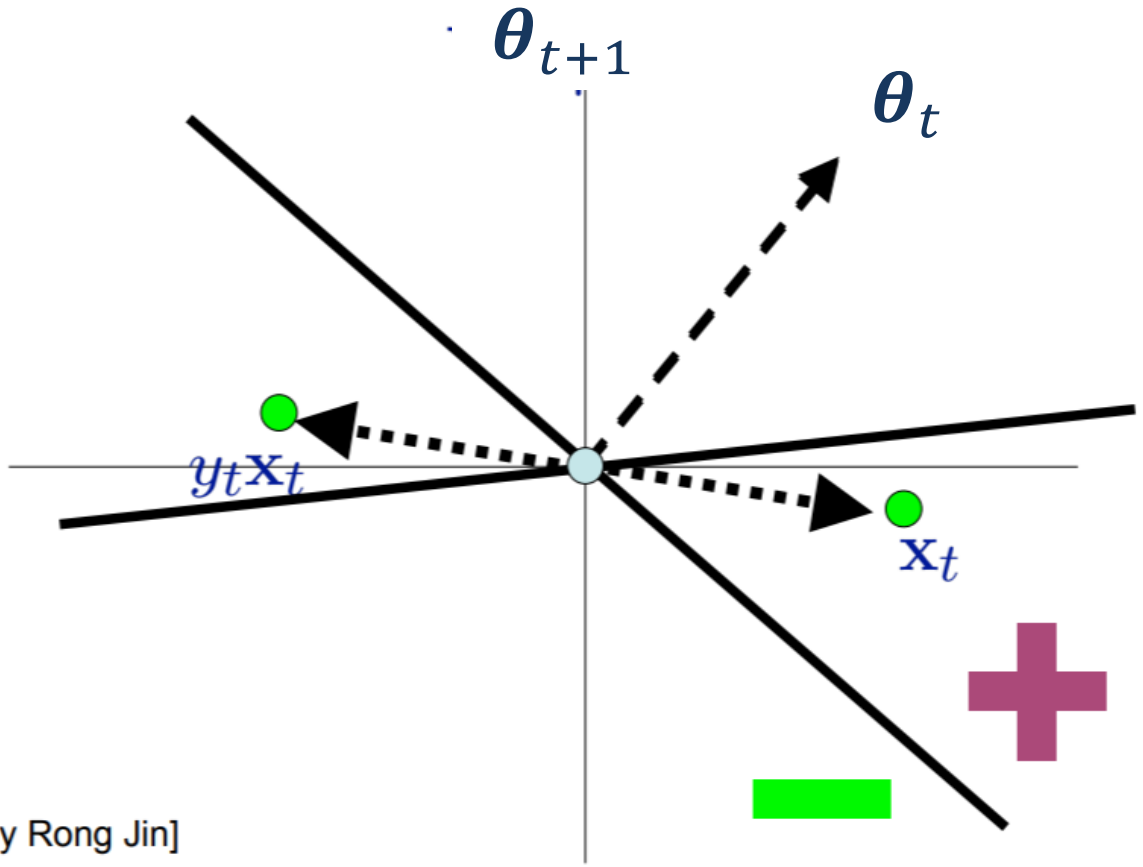
either 2 or -2

- Re-write as $\theta_j \leftarrow \theta_j + y_i x_{ij}$ (only upon misclassification)

Perceptron Rule: If x_i is misclassified, do

$$\theta \leftarrow \theta + y_i\, x_i$$

Geometric interpretation



[Slide by Rong Jin]

Online Perceptron

Let $\theta \leftarrow$[0,0,…,0]
Repeat:
    Receive training example (x_i, y_i)
    If $y_i \theta^T x_i \leq 0$ // prediction is incorrect
        $\theta \leftarrow \theta + y_i \, x_i$

Online learning – the learning mode where the model update is performed each time a single observation is received

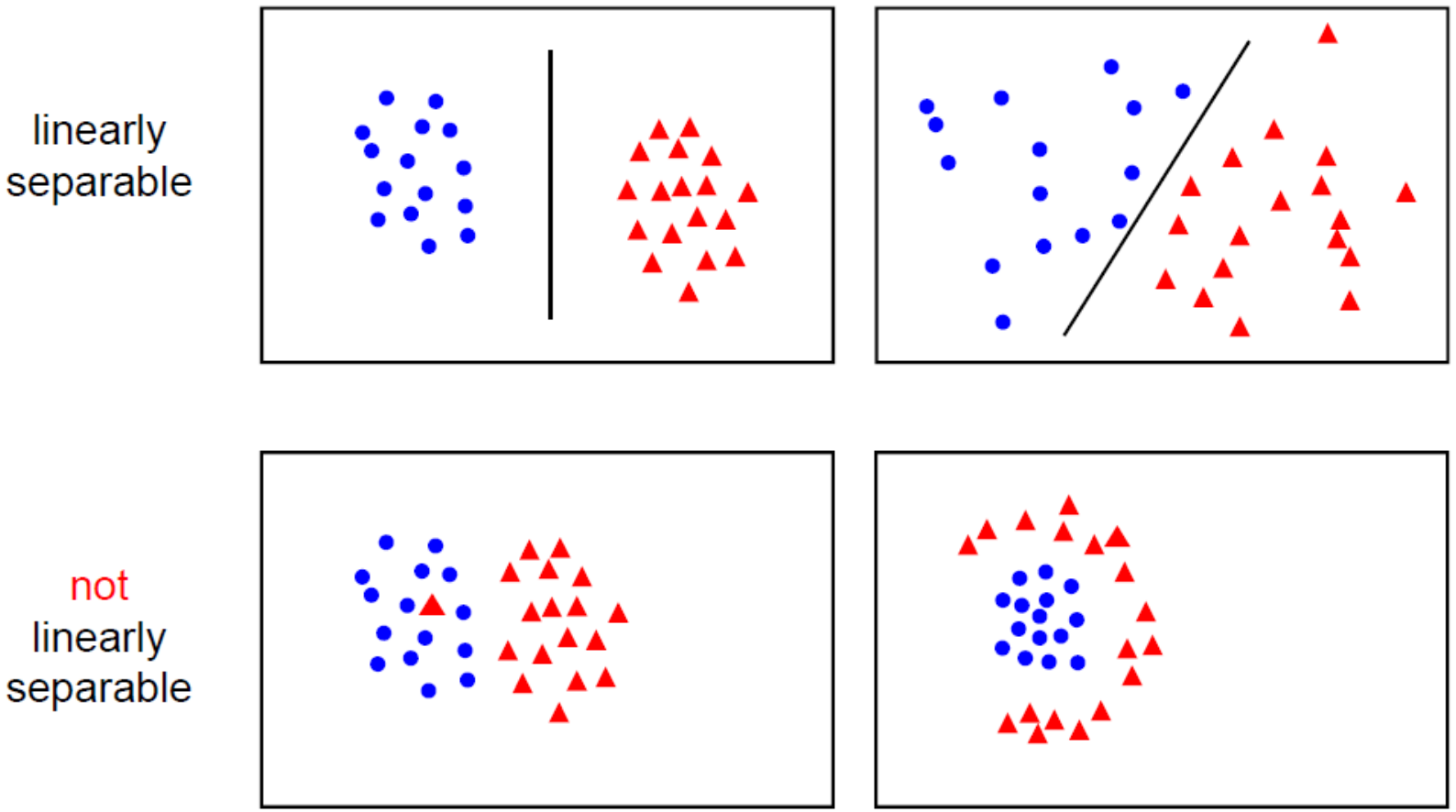
Batch learning – the learning mode where the model update is performed after observing the entire training set

Batch Perceptron

Given training data $\{(\quad x_i, y_i \quad)\}_{i=1}^{n}$
Let $\boldsymbol{\theta} \leftarrow [0, 0, \ldots, 0]$
Repeat:
    Let $\boldsymbol{\Delta} \leftarrow [0, 0, \ldots, 0]$
    for $i = 1 \ldots n$, do
        if $\quad y_i \theta^T x_i \quad \leq 0$ // prediction for ith instance is incorrect
            $\boldsymbol{\Delta} \leftarrow \boldsymbol{\Delta} + \quad y_i \, x_i$
    $\boldsymbol{\Delta} \leftarrow \boldsymbol{\Delta}/n$ // compute average update
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \quad \boldsymbol{\Delta}$
Until $\|\boldsymbol{\Delta}\|_2 < \epsilon$

Guaranteed to find separating hyperplane if data is linearly separable

Linear separability



- For linearly separable data, can prove bounds on perceptron error (depends on how well separated the data is)

The perceptron

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

- Linear classifier
- f is the sign function
- Pros
  - Very compact model (size d)

- Cons of the perceptron
  - Perceptron depends on the order of training data and it could take many steps for convergence
  - Only classifies well data that is linearly separable

Review

- Cross-validation should be performed to
  - Improve generalization and avoid over-fitting
  - Choose hyper parameters (k in kNN)
- Linear classifiers learn a hyperplane decision boundary
  - Compact representation (d+1 parameters)
- Perceptron is the first example of linear classifier
  - At the basis of neural networks
  - Has several limitations