DS 4400

Machine Learning and Data Mining I

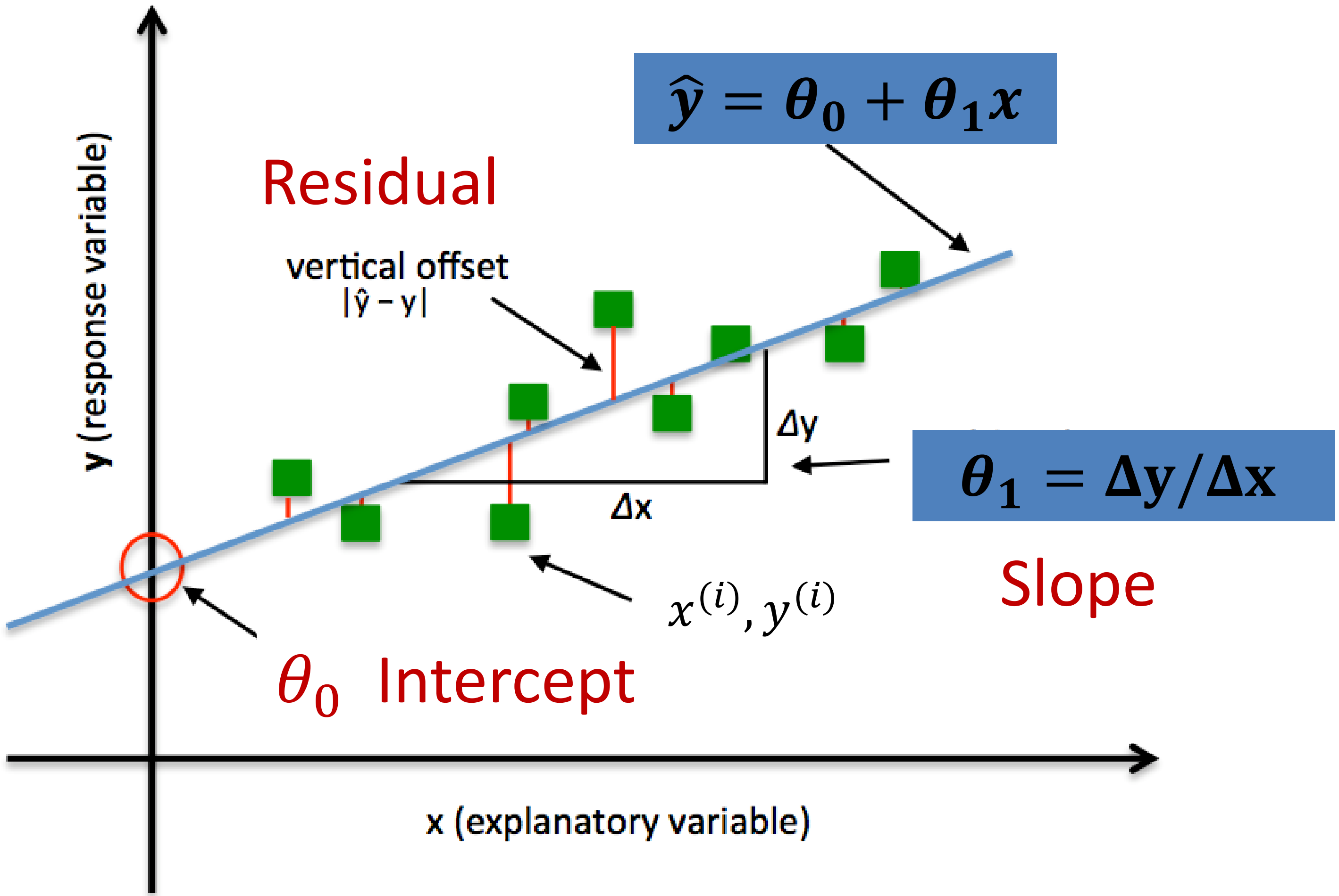
Alina Oprea
Associate Professor, CCIS
Northeastern University

September 25 2018

Review

- Solution for simple and multiple linear regression can be computed in closed form
  - Matrix inversion is computationally intense
- Gradient descent is an efficient algorithm for optimization and training LR
  - The most widely used algorithm in ML!
  - We derived GD update rule for simple and multiple LR
  - There are many practical issues with GD
- Regularization is general method to reduce model complexity and avoid overfitting
  - Add penalty to loss function
  - Ridge and Lasso regression

Simple LR



$$h_\theta(x) = \theta_0 + \theta_1 x$$

MSE $= \frac{1}{n}\sum_{i=1}^{n}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$

Simple LR – Closed Form

- Dataset $x^{(i)} \in R, y^{(i)} \in R, h_\theta(x) = \theta_0 + \theta_1 x$
- $J(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(\theta_0 + \theta_1 x^{(i)} - y^{(i)}\right)^2$ loss

$$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{2}{n}\sum_{i=1}^{n}\left(\theta_0 + \theta_1 x^{(i)} - y^{(i)}\right) = 0$$

$$\frac{\partial J(\theta)}{\partial \theta_1} = \frac{2}{n}\sum_{i=1}^{n} x^{(i)}\left(\theta_0 + \theta_1 x^{(i)} - y^{(i)}\right) = 0$$

- Closed-from solution of min loss

$$-\ \theta_0 = \bar{y} - \theta_1 \bar{x}$$

$$-\ \theta_1 = \frac{\sum (x^{(i)} - \bar{x})(y^{(i)} - \bar{y})}{\sum\left(x^{(i)} - \bar{x}\right)^2}$$

$$\bar{x} = \frac{\sum_{i=1}^{n} x^{(i)}}{n}$$

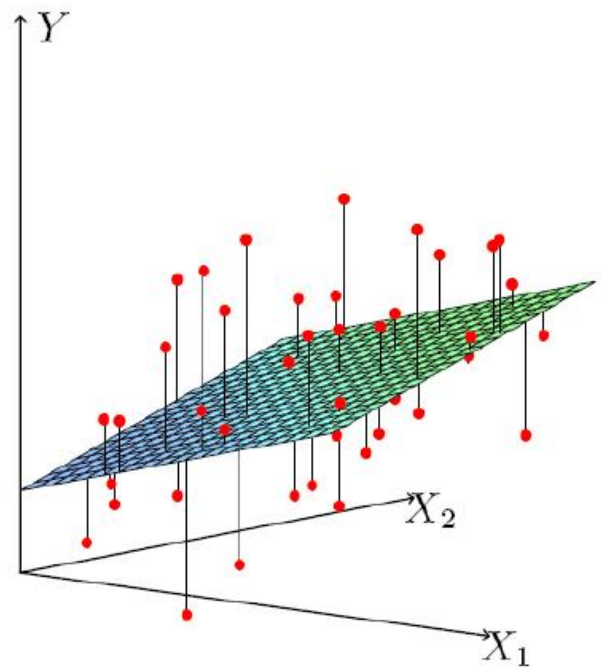
$$\bar{y} = \frac{\sum_{i=1}^{n} y^{(i)}}{n}$$

Multiple LR – Closed Form

- Dataset: $x^{(i)} \in R^d, y^{(i)} \in R$
- Hypothesis $h_\theta(x) = \theta^T x$
- MSE $= \frac{1}{n}\sum_{i=1}^{n}\left(\theta^T x^{(i)} - y^{(i)}\right)^2$ loss / cost

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

Closed-from solution



GD for Simple Linear Regression

- 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

- $J(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(\theta_0 + \theta_1 x^{(i)} - y^{(i)}\right)^2$
- $\frac{\partial J(\theta)}{\partial \theta_0} = \frac{2}{n}\sum_{i=1}^{n}(\theta_0 + \theta_1 x^{(i)} - y^{(i)})$
- $\frac{\partial J(\theta)}{\partial \theta_1} = \frac{2}{n}\sum_{i=1}^{n}(\theta_0 + \theta_1 x^{(i)} - y^{(i)})\, x^{(i)}$

Update rules for Gradient Descent

GD for Multiple Linear Regression

- 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

For Linear Regression:
$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \frac{\partial}{\partial \theta_j} \frac{1}{n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right)^2$$
$$= \frac{\partial}{\partial \theta_j} \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{k=0}^{d} \theta_k x_k^{(i)} - y^{(i)} \right)^2$$
$$= \frac{2}{n} \sum_{i=1}^{n} \left(\sum_{k=0}^{d} \theta_k x_k^{(i)} - y^{(i)} \right) \times \frac{\partial}{\partial \theta_j} \left(\sum_{k=0}^{d} \theta_k x_k^{(i)} - y^{(i)} \right)$$
$$= \frac{2}{n} \sum_{i=1}^{n} \left(\sum_{k=0}^{d} \theta_k x_k^{(i)} - y^{(i)} \right) x_j^{(i)}$$

Update rules for Gradient Descent

Gradient Descent vs Closed Form

Gradient Descent

- Initialize θ
- 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 Solution
• Requires multiple iterations • Need to choose α • Works well when n is large • Can support incremental learning	• Non-iterative • No need for α • Slow if n is large – Computing $(X^\intercal X)^{-1}$ is roughly $O(n^3)$

Outline

- Classification
  - Linear classifiers
  - Online perceptron and batch perceptron
- Instance learners
  - kNN
- Evaluation of classification algorithms
  - Metrics (accuracy, precision, recall)
- Cross validation

Supervised learning

Problem Setting

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

Input: Training examples of unknown target function f

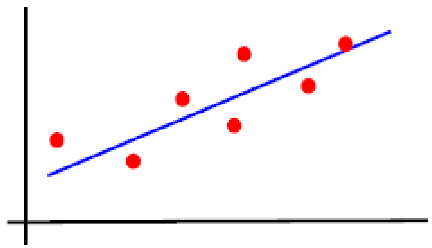
$$\{x^{(i)}, y^{(i)}\}, \text{for } i = 1, \ldots, n$$

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

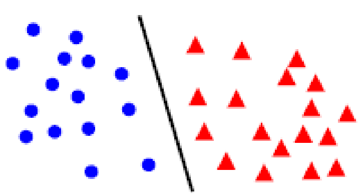
$$\hat{f}\left(x^{(i)}\right) \approx y^{(i)}$$

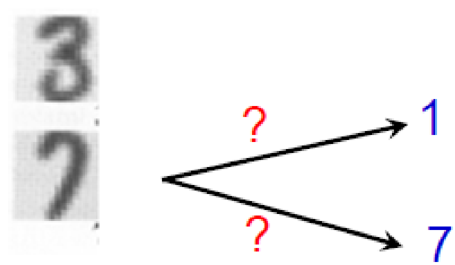
Three canonical learning problems

1. Regression - supervised
   - estimate parameters, e.g. of weight vs height

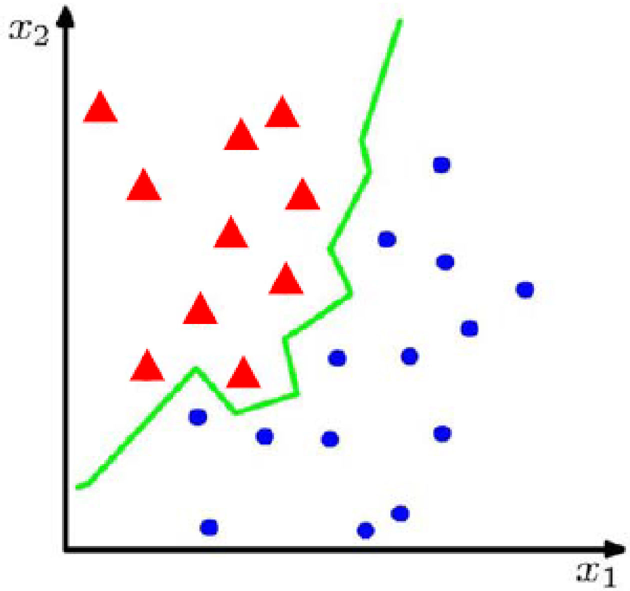


2. Classification - supervised
   - estimate class, e.g. handwritten digit classification





Classification



Binary or discrete

- Suppose we are given a training set of N observations

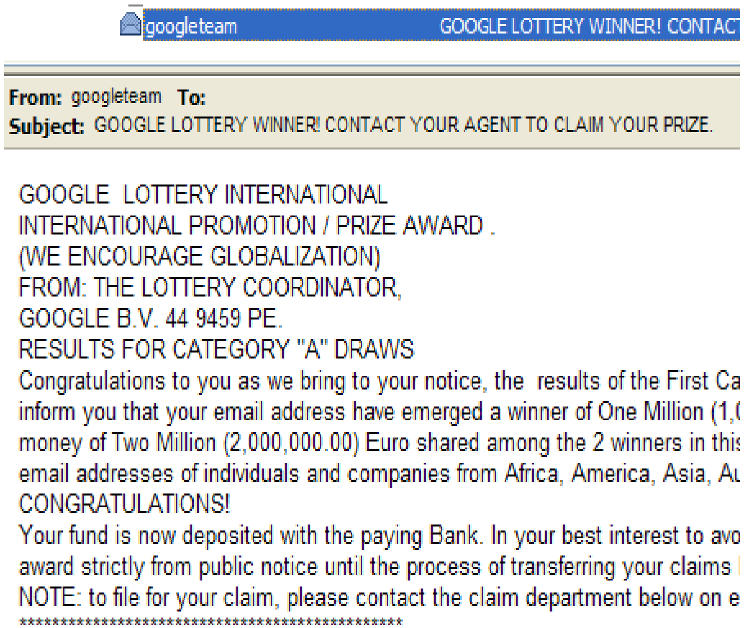
$$\{x^{(1)}, \dots, x^{(n)}\} \text{ and } \{y^{(1)}, \dots, y^{(n)}\}, x^{(i)} \in R^d, y^{(i)} \in \{-1, 1\}$$

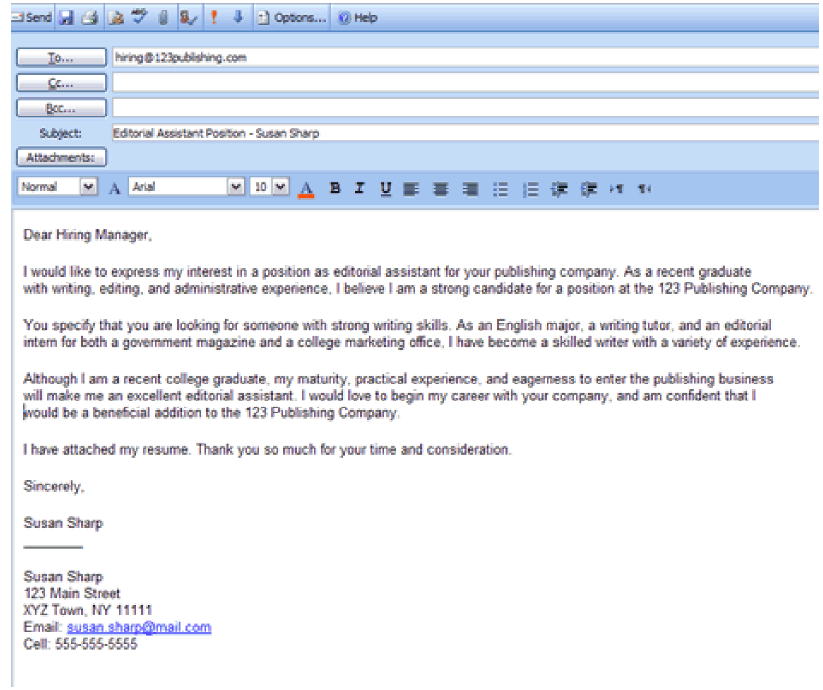
- Classification problem is to estimate f(x) from this data such that

$$f\left(x^{(i)}\right) = y^{(i)}$$

Example 1

Classifying spam email





Content-related features

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

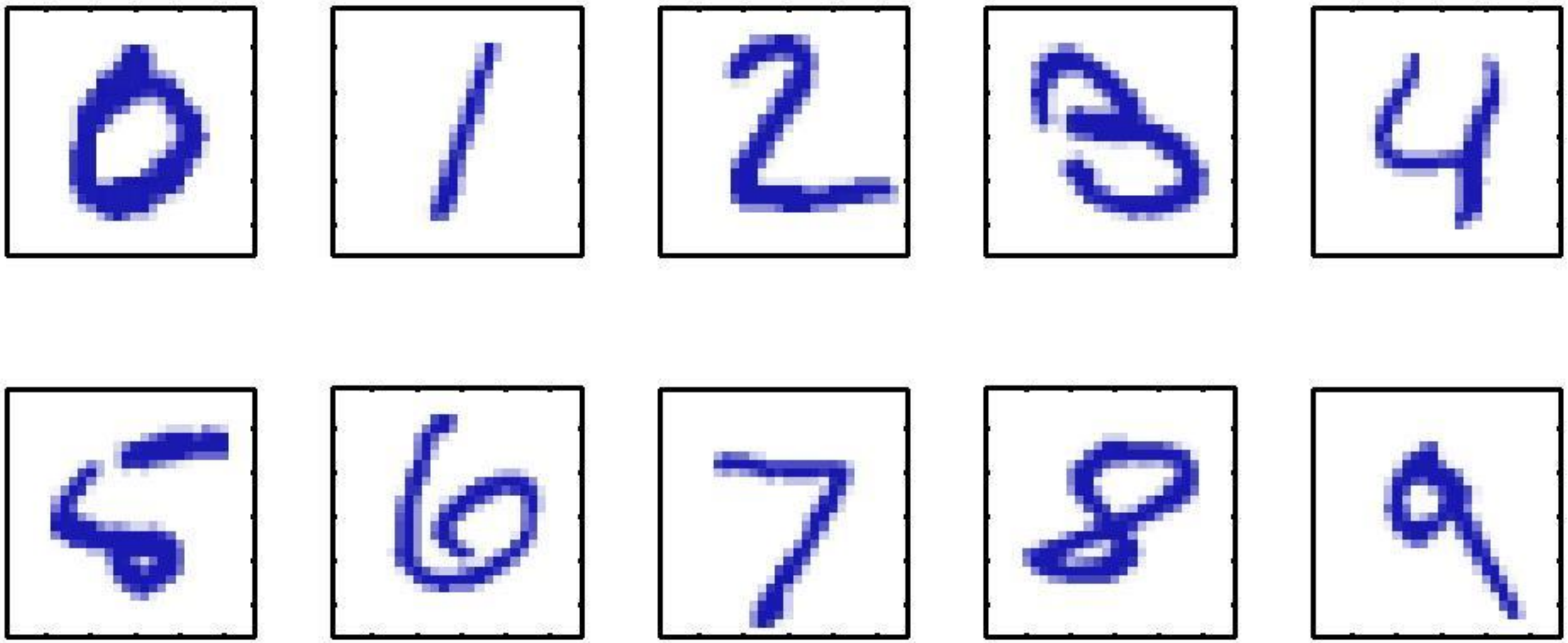
Structural features

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

Binary classification: SPAM or HAM

Example 2

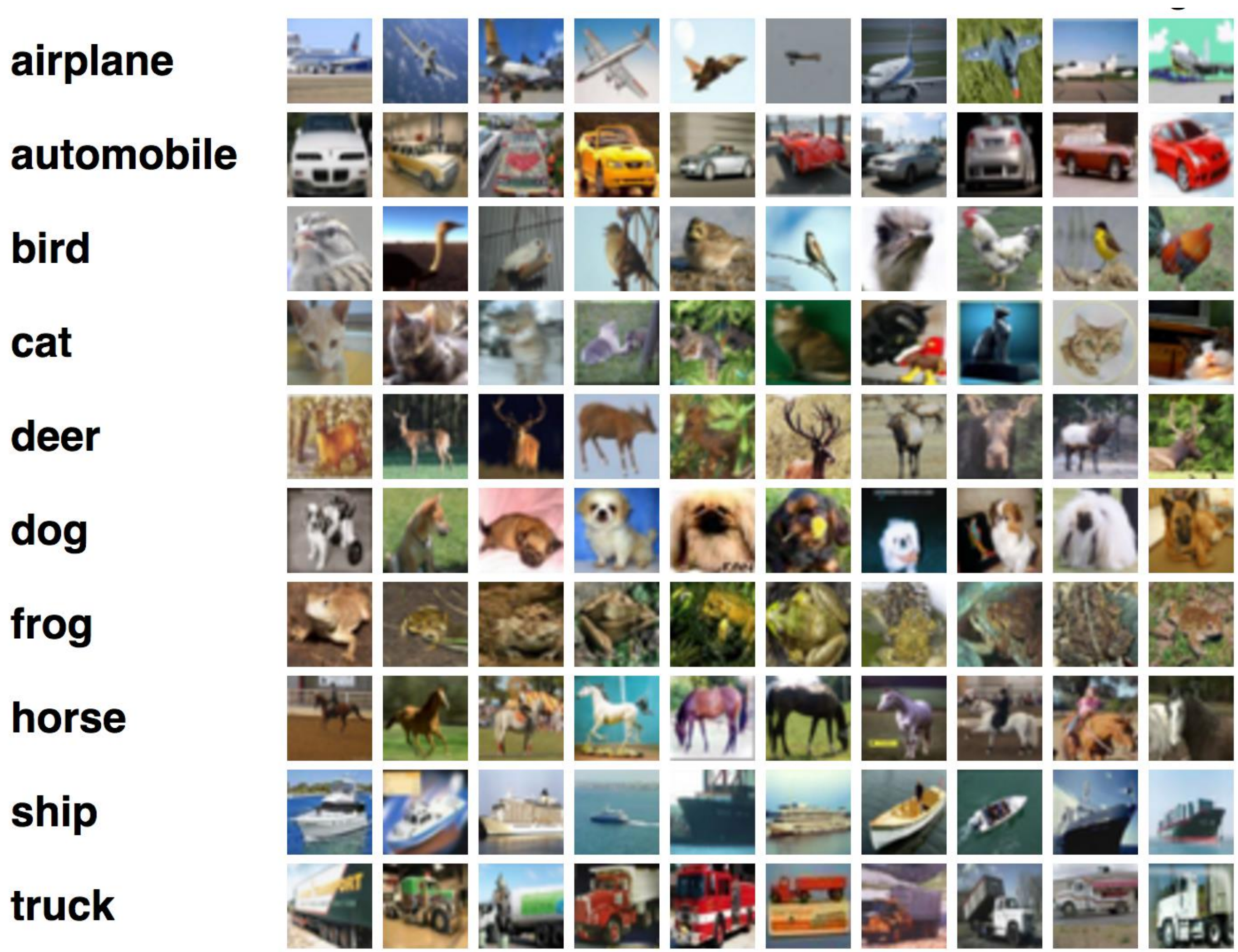
Handwritten Digit Recognition



Multi-class classification

Example 3

Image classification



Multi-class classification

Supervised Learning Process

Training

Data → Pre-processing → Feature extraction → Learning model

Labeled (Typically)

Normalization
Standardization

Feature
Selection

Classification
Regression

Testing

New data → Learning model → Predictions

Unlabeled

Malicious
Benign

Classification

Risk score

Regression

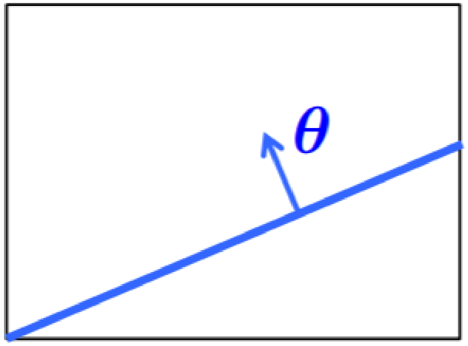
History of Perceptrons

- 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



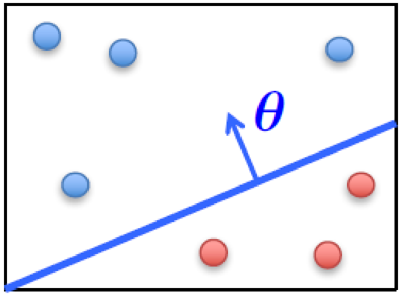
  - Assumed to pass through the origin
    - This is because we incorporated bias term θ_0 into it by $x_0 = 1$

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

Linear classifiers

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

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



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

– 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
...
```

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

$\sum_i \ x_i \theta_i > 0$ ➔ 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 $(\boldsymbol{x}^{(i)}, y^{(i)})$

$$\theta_j \leftarrow \theta_j - \frac{1}{2}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right) x_j^{(i)}$$

The Perceptron

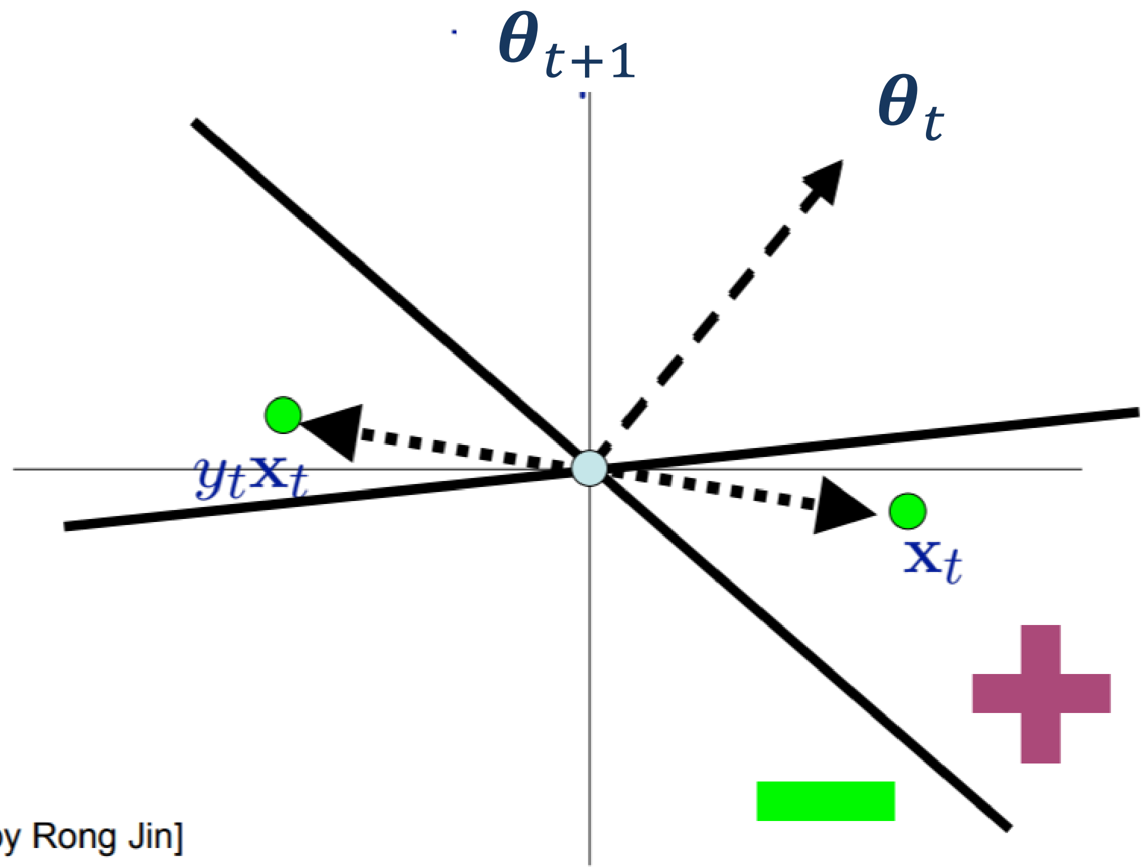
- The perceptron uses the following update rule each time it receives a new training instance $(\boldsymbol{x}^{(i)}, y^{(i)})$

$$\theta_j \leftarrow \theta_j - \frac{1}{2} \underbrace{\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)}_{\text{either 2 or -2}} x_j^{(i)}$$

- Re-write as $\theta_j \leftarrow \theta_j + \quad y^{(i)} x_j^{(i)}$ (only upon misclassification)

Perceptron Rule: If $\boldsymbol{x}^{(i)}$ is misclassified, do $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + y^{(i)} \boldsymbol{x}^{(i)}$

Geometric interpretation



[Slide by Rong Jin]

Online Perceptron

Let $\boldsymbol{\theta} \leftarrow [0, 0, \ldots, 0]$
Repeat:
  Receive training example $(\boldsymbol{x}^{(i)}, y^{(i)})$
  if $y^{(i)} \theta^T x^{(i)} \leq 0$ // prediction is incorrect
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + y^{(i)} \boldsymbol{x}^{(i)}$

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

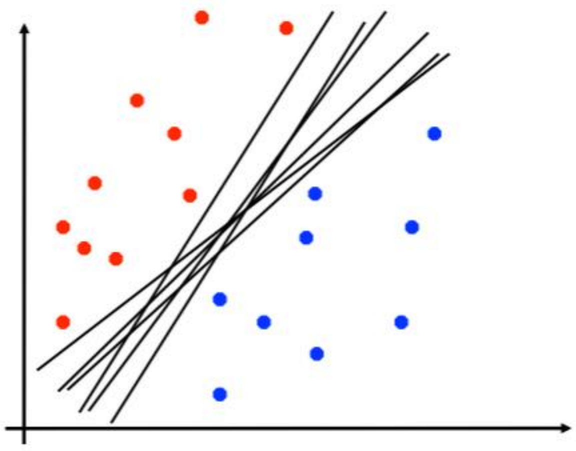
Batch Perceptron

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

Guaranteed to find separating hyperplane if data is linearly separable

Perceptron Limitations

- Is dependent on starting point
- It could take many steps for convergence
- Perceptron can overfit
  - Move the decision boundary for every example



Which of this is optimal?

Improving the Perceptron

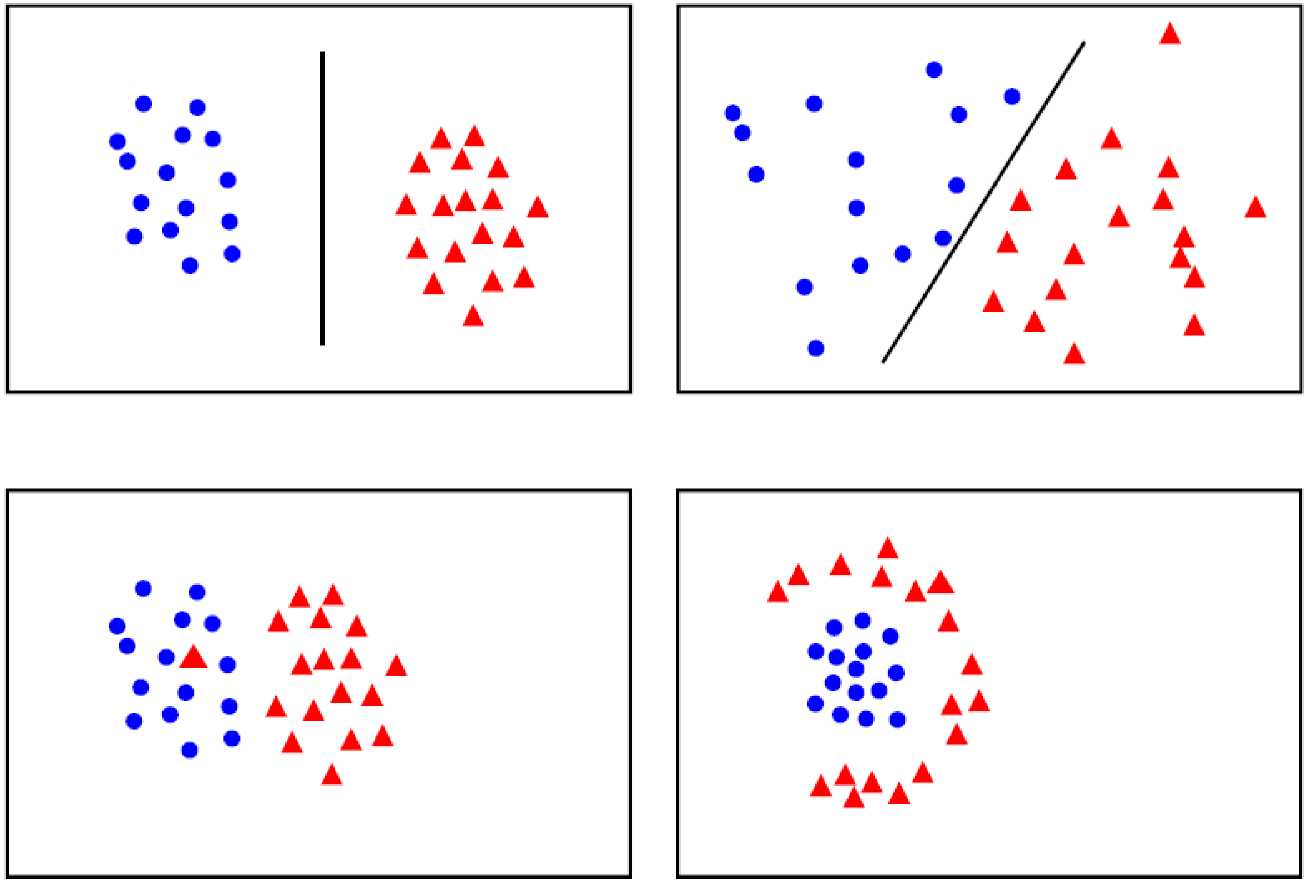
- The Perceptron produces many θ's during training
- The standard Perceptron simply uses the final θ at test time
  - This may sometimes not be a good idea!
  - Some other θ may be correct on 1,000 consecutive examples, but one mistake ruins it!

- **Idea:** Use a combination of multiple perceptrons
  - (i.e., neural networks!)
- **Idea:** Use the intermediate θ's
  - **Voted Perceptron**: vote on predictions of the intermediate θ's
  - **Averaged Perceptron**: average the intermediate θ's

Linear separability

linearly separable

not linearly separable

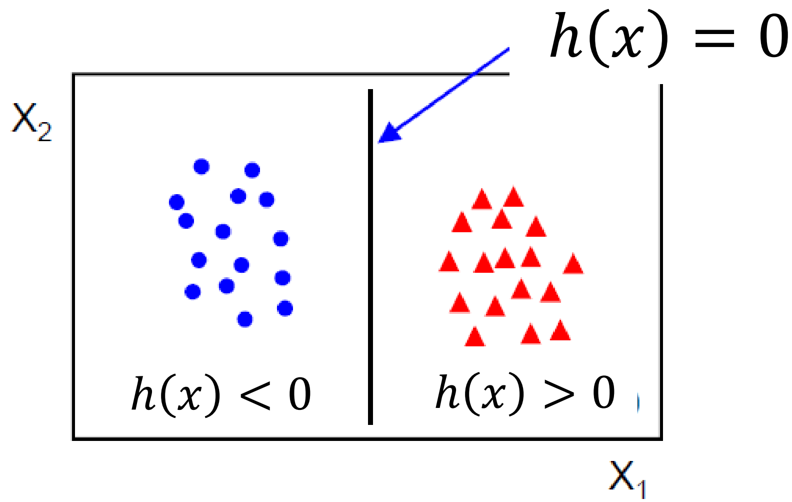


- For linearly separable data, can prove bounds on perceptron error (depends on how well separated the data is)

Linear classifiers

A linear classifier has the form

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



- Properties
  - $(\theta_0, \theta_1, \ldots, \theta_d)$ = model parameters
  - Perceptron is a special case with $f = sign$
  - Linear regression can be used as classifier $f(x) = x$
    - If $h(x) > 0.5$, output 1; otherwise output -1
- Pros
  - Very compact model (size d)
  - Perceptron is fast
- Cons
  - Does not work for data that is not linearly separable

Outline

- Classification
  - Linear classifiers
  - Online perceptron and batch perceptron
- Instance learners
  - kNN
- Evaluation of classification algorithms
  - Metrics (accuracy, precision, recall)
- Cross validation

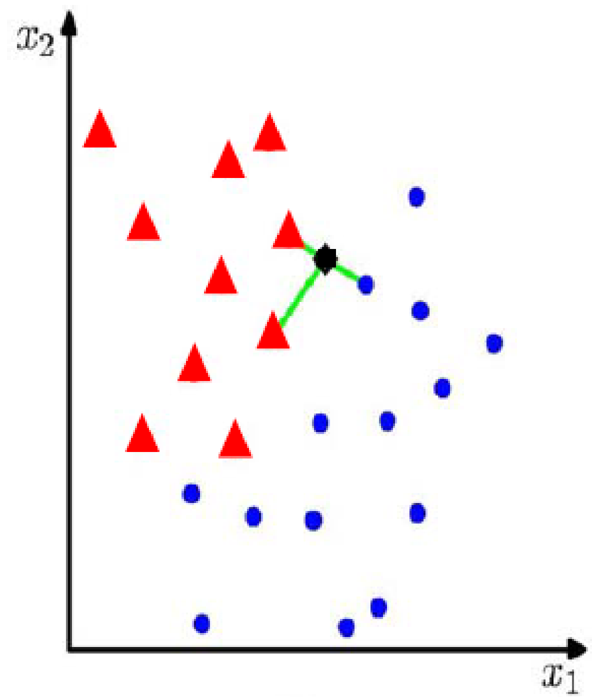
K Nearest Neighbour (K-NN) Classifier

Algorithm

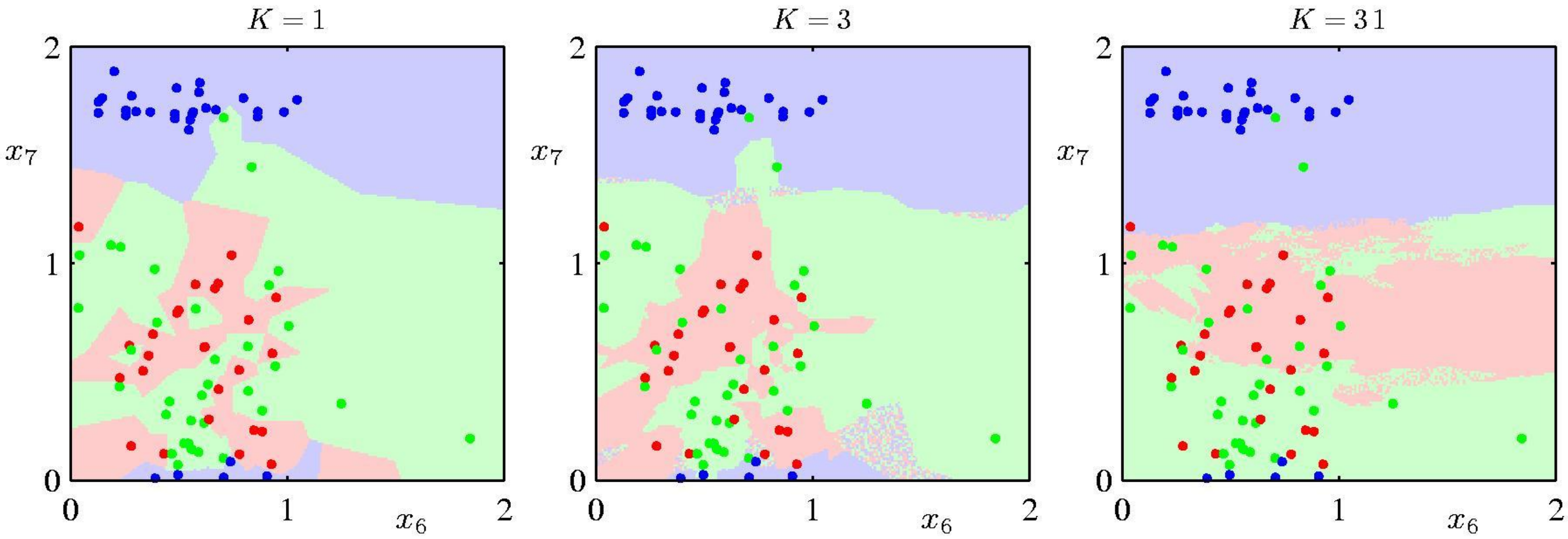
- For each test point, x, to be classified, find the K nearest samples in the training data
- Classify the point, x, according to the majority vote of their class labels

e.g. K = 3

• applicable to multi-class case



K-Nearest-Neighbours for multi-class classification



Vote among multiple classes

Vector distances

Vector norms: A norm of a vector ||x|| is informally a measure of the "length" of the vector.

$$||x||_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

– Common norms: L_1, L_2 (Euclidean)

$$||x||_1 = \sum_{i=1}^{n} |x_i| \qquad ||x||_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

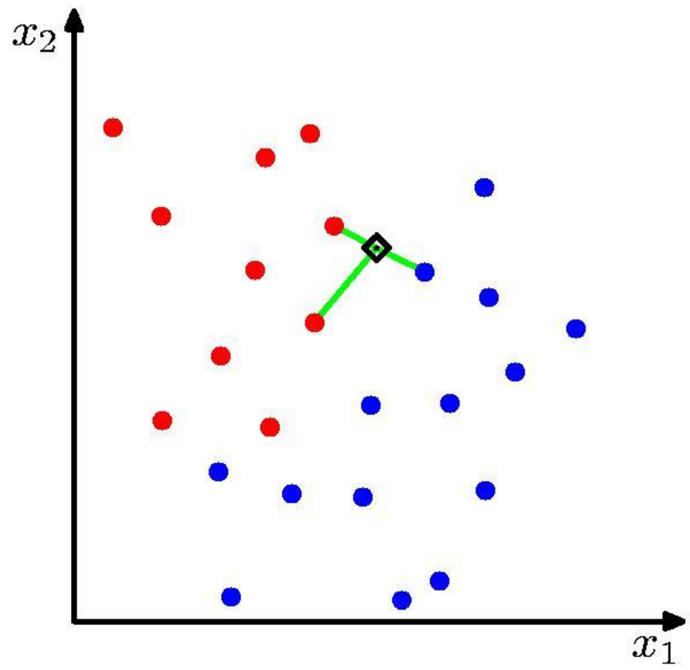
Norm can be used as distance between vectors x and y

- $||x - y||_p$

Distance norms

- Euclidean Distance $$\sqrt{\left(\sum_{i=1}^{k}(x_i - y_i)^2\right)}$$
- Mahattan 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

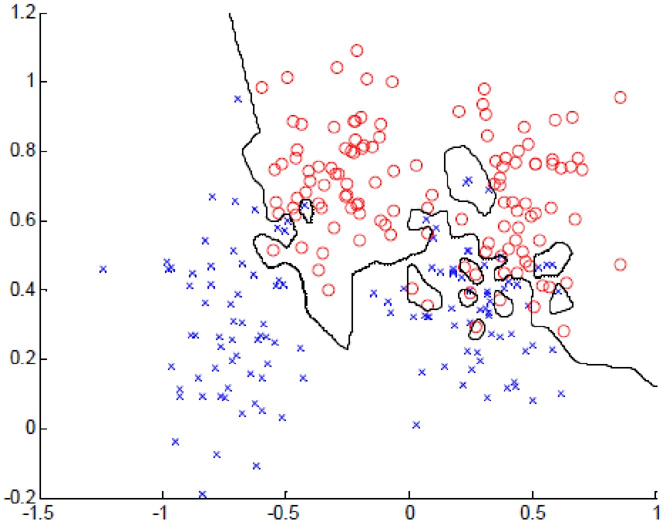


- Algorithm (to classify point x)
  - Find k nearest points to x (according to distance metric)
  - Perform majority voting to predict class of x
- Properties
  - Does not learn any model in training!
  - Instance learner (needs all data at testing time)

K = 1

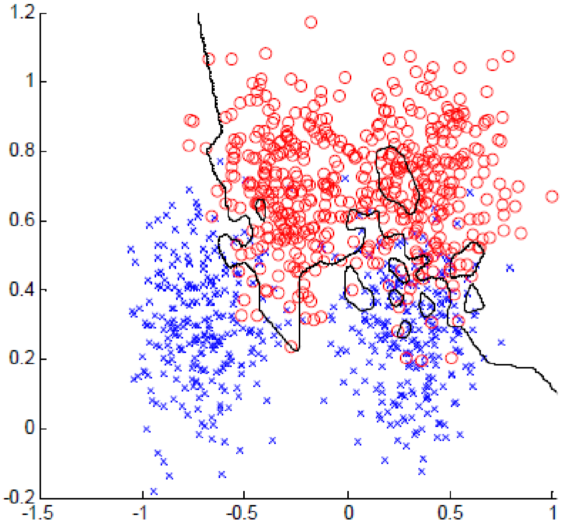
Overfitting!

Training data



error = 0.0

Testing data

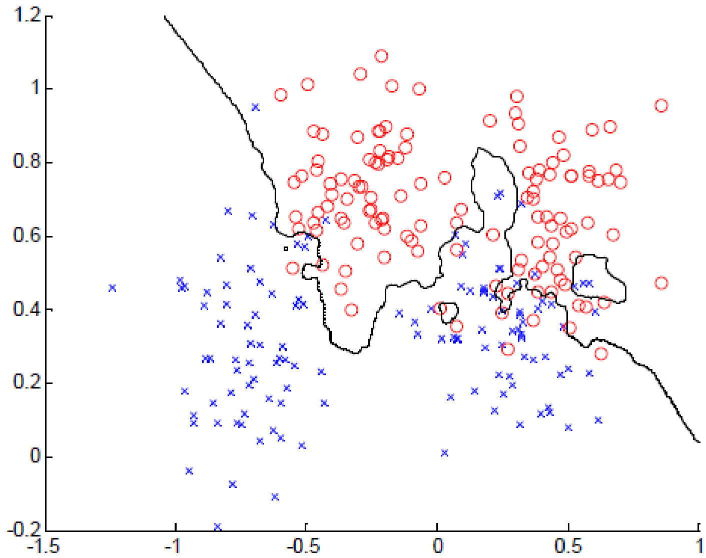


error = 0.15

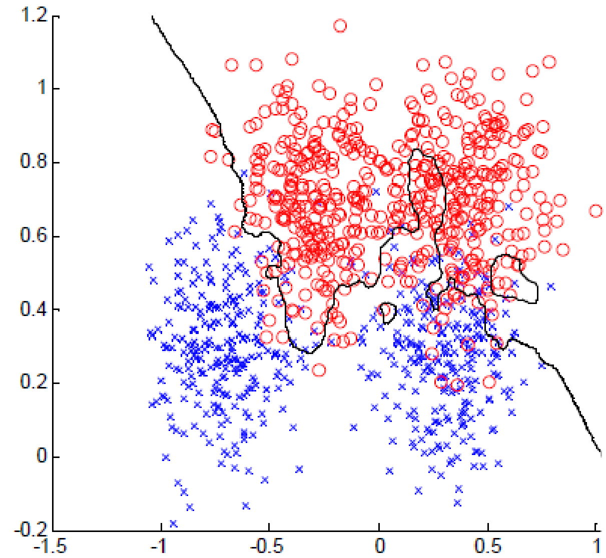
How to choose k (hyper-parameter)?

K = 3

Training data



Testing data



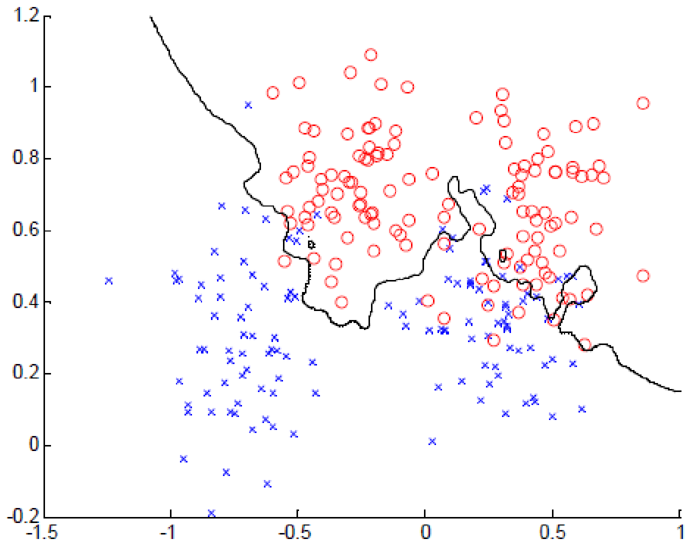
error = 0.0760

error = 0.1340

How to choose k (hyper-parameter)?

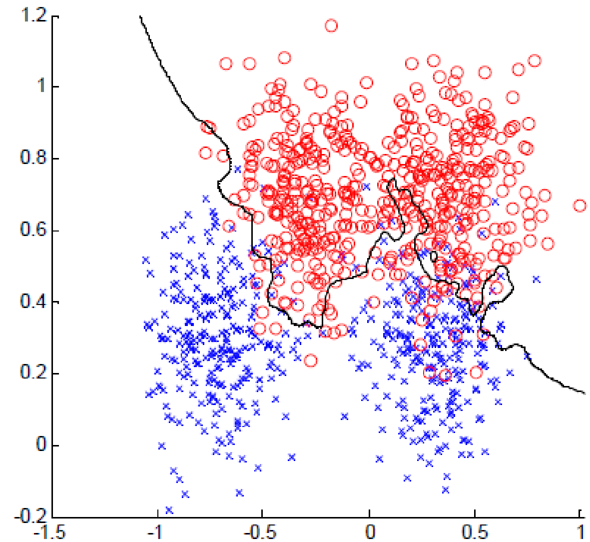
K = 7

Training data



error = 0.1320

Testing data



error = 0.1110

How to choose k (hyper-parameter)?

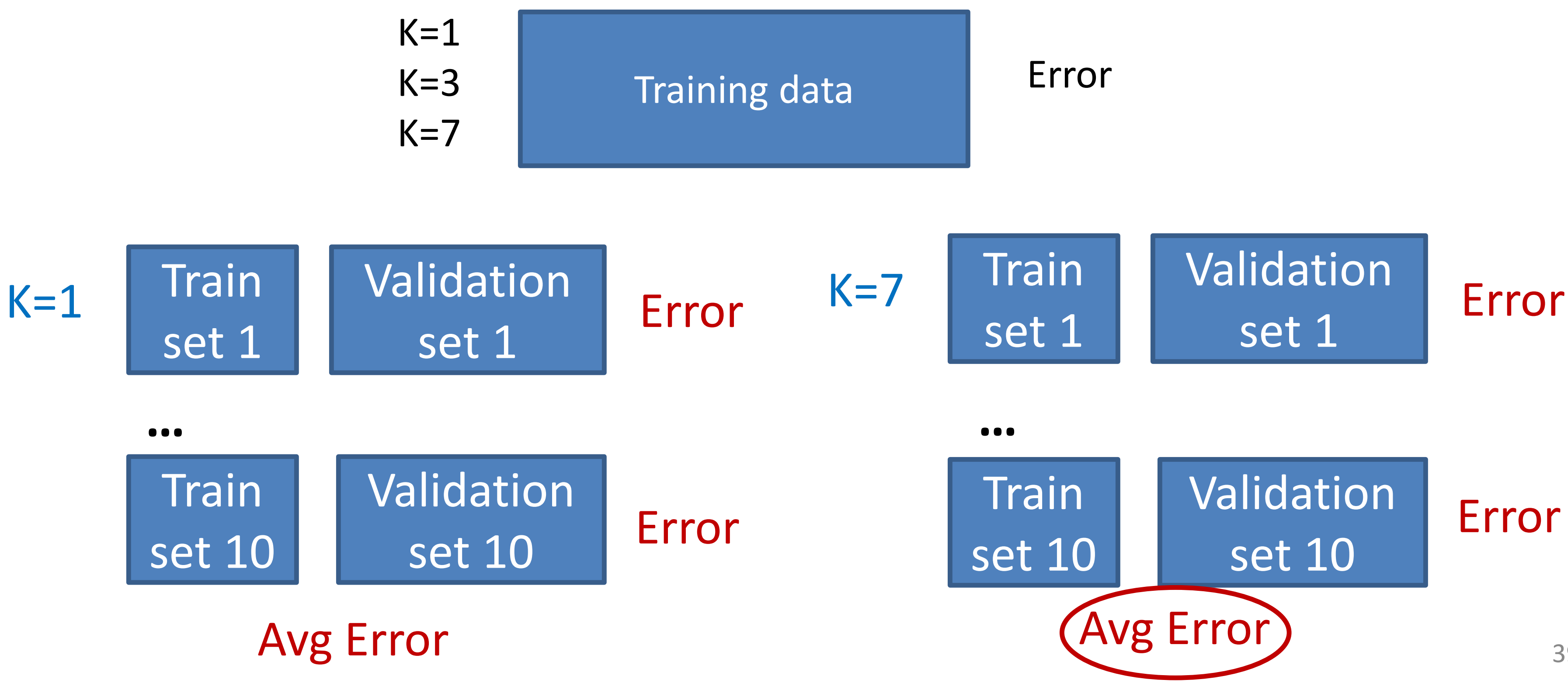
Cross-validation

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



Review

- Classification is a supervised learning problem
  - Prediction is binary or multi-class
- Examples
  - Linear classifiers (perceptron)
  - Instance learners (kNN)
- ML methodology includes cross-validation for parameter selection and estimation of model error
  - K-fold CV or LOOCV

Acknowledgements

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