Classification Methods

Chapter 26

Supervised learning techniques

- **Regression models (**linear**, logistic)**
- **Classification**
- Similarity learning
- Support vector machines
- Linear discriminate analysis
- Decision tree
- Neural networks

How do you know what is in a class?

Types of learning models

- **One-class** – training data is all one class
  - Build a model that predicts class membership
  - Anomaly detection
- **Binary** or **Two-class** – Training samples from exactly two classes
  - Find a boundary between the two classes
- **Multi-class learning** – More that two

Goals

- Provide a reasonable good fit for available data
- Have a reasonable chance of making good predictions about unseen data

- Not too specific
- Not too general
- Fits 'just right'

Supervised learning basics

- Data has feature vectors and labels
- Divide into **training** and **test** data
- Create a model that minimizes **training error**

Confusion matrix (error types)

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

$$accuracy = \frac{true\ positive + true\ negative}{true\ positive + true\ negative + false\ positive + false\ negative}$$

Is accuracy a good measure

- Not so much when there is a large **class imbalance**
  - Disease detection

$$sensitivity = \frac{true\ positive}{true\ positive + false\ negative}$$

$$specificity = \frac{true\ negative}{true\ negative + false\ positive}$$

$$positive\ predictive\ value = \frac{true\ positive}{true\ positive + false\ positive}$$

$$negative\ predictive\ value = \frac{true\ negative}{true\ negative + false\ negative}$$

sensitivity = recall
specificity = precision

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

K-nearest neighbor

- Simplest **classification** method

  - Create training and test data sets
  - For each example in the test set
    - Find the **k** nearest neighbors in the training set
    - Align with the majority of the neighbors

- $Complexity \ = O\big(len(training) \times len(test)\big)$
- For large data sets it might be useful to down sample

Boston Marathon data

```
"Gebremariam Gebregziabher",M,27,14,ETH,142.93
"Matebo Levy",M,22,2,KEN,133.10
"Cherop Sharon",F,28,1,KEN,151.83
"Chebet Wilson",M,26,5,KEN,134.93
"Dado Firehiwot",F,28,4,ETH,154.93
"Korir Laban",M,26,6,KEN,135.48
"Jeptoo Rita",F,31,6,KEN,155.88
"Korir Wesley",M,29,1,KEN,132.67
"Kipyego Bernard",M,25,3,KEN,133.22
"Barmasai David",M,23,8,KEN,137.27
"Rono Georgina",F,31,3,KEN,153.15
"Getaneh Genet",F,26,9,ETH,162.18
"Kisorio Mathew",M,22,10,KEN,138.25
"Sigei Diana",F,24,5,KEN,155.67
"Sumgong Jemima Jelagat",F,27,2,KEN,151.87
"Hartmann Jason",M,31,4,USA,134.52
"Leonteva Nadezdha",F,27,8,RUS,160.67
"Butter Michel",M,26,7,NED,136.63
"Fujita Mayumi",F,28,7,JPN,159.18
```

80/20 split

- Randomly selection 20% for training data
- The other 80% will be our test data

```
def split80_20(examples):
    num_examples = len(examples)
    sampleIndicies = random.sample(range(num_examples),
                                   num_examples//5)
    training_set, test_set = [], []
    for i in range(num_examples):
        if i in sampleIndicies:
            test_set.append(examples[i])
        else:
            training_set.append(examples[i])
    return (training_set, test_set)
```

How accurate is accurate?

- Baseline using a **prevalence** classifier
  - What is the frequency of **label** in the training set
    - (e.g. how prevalent is label in the training set)
  - We should see the same prevalence in the test data
    - Can replace k-nearest neighbor with a randomizer

Comparing results

	KNN	Prevalence	Reduced KNN
Accuracy	0.651	0.581	0.676
Sensitivity	0.699	0.580	0.727
Specificity	0.582	0.583	0.606
Positive Predict	0.703	0.653	0.714
Negative Predict	0.578	0.506	0.621

What is the right k

- We want an *odd* number for tie-breakers
- **N-fold cross validation**
  - For each possible (odd) value of k
    - For n folds
      - Create new test and training data from original training data
      - Perform KNN on new subsets
      - Calculate accuracy
    - Calculate mean accuracy of using k
  - Choose a “reasonable” value for k

Regression analysis

- Statistical method that uses one or more **independent variables** to determine the value of a **dependent variable**
- In **logistic regression** the dependent variable is binary (1 or 0) and independent variables are either binary or continuous
  - Independent variables = feature vector
  - Dependent variable = label
  - https://scikit-learn.org/stable/index.html
- Extensions of logistic regression
  - Multinomial – categorical
  - Ordered

Logistic regression in scikit-learn

- Class LogisticRegression
  - Parameters – control type of logistic regression to be performed
  - Attributes
    - classes_ - list of classes known to the model
    - coef_ - coefficients of features
- Function fit()
  - Accepts a list of feature vectors and associated list of labels
  - Returns a LogisticRegression object
- Function predict_proba()
  - Accepts a feature vector
  - Returns probability for each possible outcome

Comparing results (with Logistic Regression)

	KNN	Prevalence	Reduced KNN	Logistic Regression
Accuracy	0.651	0.581	0.676	0.651
Sensitivity	0.699	0.580	0.727	0.693
Specificity	0.582	0.583	0.606	0.592
Positive Predict	0.703	0.653	0.714	0.705
Negative Predict	0.578	0.506	0.621	0.579

Receiver Operating Characteristic (ROC) curve

- Visualize sensitivity (true positive) vs false positive (1 – sensitivity)
- Area Under the ROC curve
  - Probability the model will assign a **positive** value to a positive example than to a negative example
  - **Discrimination** of the model