Issues (cont'd)

- Class imbalance
- Classifier takes too long
- Classifier doesn't generalize well

Classifier takes too long

- Subsampling:
  - Do not necessarily use all the data
  - Learning curve suggests training size

- Distributed Approach:
  - How to split the data and combine the results
  - Depends on algorithm
  - Distributed-computing frameworks: Hadoop, Mahoot, MapReduce, TensorFlow…

Classifier does not generalize well

- A classifier
  - Has a low error rate on the training set
  - Has high error when you evaluate on a test set

- Solutions
  - Try a smaller set of features
  - Get more training examples
  - Obtain new features

CLASS IMBALANCE

Application 1

- Data source: an automated inspection system for monitoring products and find defective items
- How many items are defective?
- How many items are operational?

Defective products: 4 in 1 million

Defective vs Operational: 4 vs 999,996

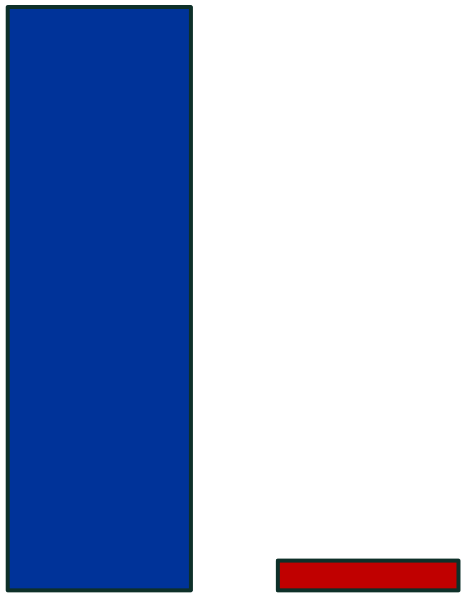
Application 2

- Data source: credit card fraud detection system
- How many transactions are fraudulent?
- How many transactions are legitimate?

Fraud transactions: 1 in 100

Class Imbalance

- A disproportionate number of instances that belong to different classes



Challenges

- First, it can be difficult to find enough samples of a rare class.
- Second, accuracy which is a traditional measure for evaluating classification performance is not good for evaluating models in the case of class imbalance.

Challenges (cont’d)

- In credit card fraud example: what is the accuracy of a model that classifies ALL transactions as legitimate?
- In fact, a correct classification of the rare class has a greater value than a correct classification of the majority class
- Issues:
  - Performance measures need to be modified

Approaches

- Alternative metrics
  - capture different criteria performance than accuracy

- Cost sensitive learning

  - minimize the cost of a model on a training dataset by assigning uneven penalties or costs when making predictions.

- Sampling

Alternative Metrics

- Binary classification:
  - Rare: Positive
  - Majority: Negative

Confusion Matrix:

		Predicted Class	
		+	–
Actual Class	+	f_{++} (TP)	f_{+-} (FN)
	–	f_{-+} (FP)	f_{--} (TN)

Alternative Metrics

- **True Positive Rate**: fraction of positive instances correctly predicted $TPR = TP/(TP + FN)$

- **True Negative Rate**: fraction of negative instances correctly predicted $TNR = TN/(FP + TN)$

- **False Positive Rate**: fraction of negative instances predicted positive $FPR = FP/(TN + FP)$

- **False Negative Rate**: fraction of positive instances predicted negative $FNR = FN/(TP + FN)$

		Predicted Class	
		+	–
Actual Class	+	f_{++} (TP)	f_{+-} (FN)
	–	f_{-+} (FP)	f_{--} (TN)

Alternative Metrics

- **Recall**: fraction of positive records correctly predicted (true positive)

$$r = TP/(TP + FN)$$

- **Precision**: fraction of records that are truly positive in the set predicted as positive (ratio between the True Positives and all the Positives)

$$p = TP/(TP + FP)$$

		Predicted Class	
		+	–
Actual Class	+	f_{++} (TP)	f_{+-} (FN)
	–	f_{-+} (FP)	f_{--} (TN)

•**For example:**

For all the patients who actually have heart disease, recall tells us how many we correctly identified as having a heart disease.

The measure of patients that we correctly identify having a heart disease out of all the patients we predicted they have heart disease. -- precision

Alternative Metrics

- **Recall**: fraction of positive records correctly predicted (true positive)

  $$r = TP/(TP + FN)$$

- **Precision**: fraction of records that are truly positive in the set predicted as positive

  $$p = TP/(TP + FP)$$

- A model can usually maximize one but not the other
- Building a model that maximizes both is difficult

- **F_1 measure**:

  $$F_1 = 2rp/(r + p) = 2/(1/r + 1/p)$$

```
from sklearn.metrics import precision_recall_fscore_support
```

The support is the number of occurrences of each class

Credit Card Fraud Example

- **Recall**:

  $r = TP/(TP + FN) = 1/5 = 0.2$

- **Precision**:

  $p = TP/(TP + FP) = 1/1 = 1$

- **F_1 measure**:

  $F_1 = 2rp/(r + p) = 2*0.2*1/1.2 = 0.33$

- **Error Rate**:

  $\varepsilon = 4/100 = 0.04$

		Predicted Class	
		+	–
Actual Class	+	1 (TP)	4 (FN)
	–	0 (FP)	95 (TN)

Credit Card Fraud Example

- **Recall**:

  $r = TP/(TP + FN) = 4/5 = 0.8$

- **Precision**:

  $p = TP/(TP + FP) = 4/4 = 1$

- **F_1 measure**:

  $F_1 = 2rp/(r + p) = 2/(1/r + 1/p) = 2*0.8*1/1.8 = 0.88$

- **Error Rate**:

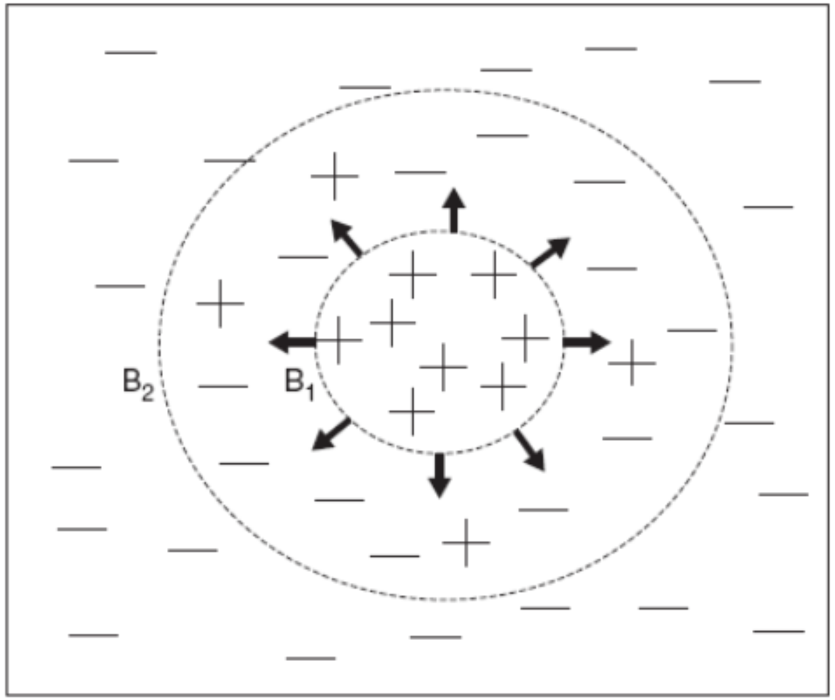
  $\varepsilon = 1/100 = 0.01$

		Predicted Class	
		+	–
Actual Class	+	4 (TP)	1 (FN)
	–	0 (FP)	95 (TN)

Cost Sensitive Learning

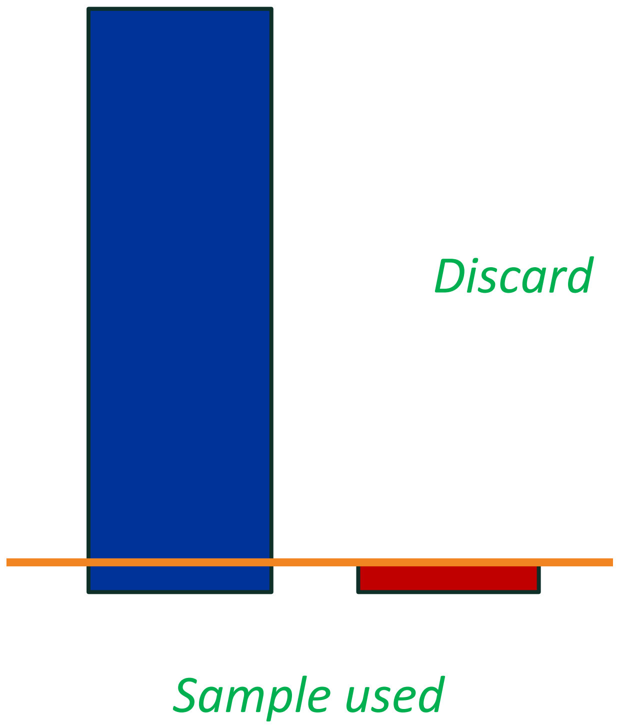
		Predicted Class	
		+	–
Actual Class	+	-1	100
	–	1	0

- Incorporate cost in the process of building the model
- Decisions tree:
  - Select the attribute for the split
  - Decide whether to prune a subtree
- Nearest Neighbor:
  - Update decision boundary based on cost



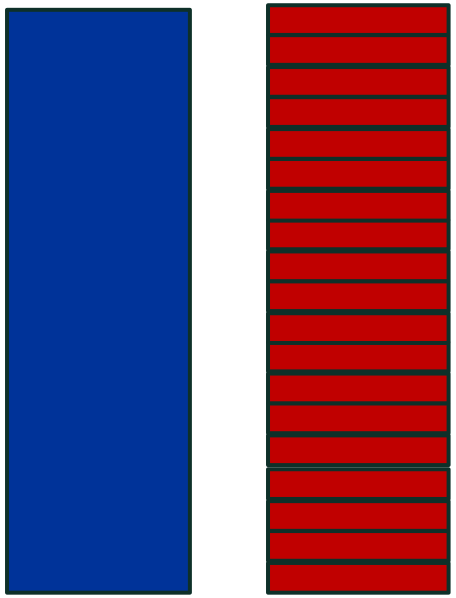
Sampling-Based Approaches

- Modify distribution so rare classes are well represented
- **Undersampling**:
  - Choose all positive records
  - Randomly choose an equal number of negative records
- Problem: might drop some important negative records
- Solution: Perform undersampling multiple times



Sampling-Based Approaches

- **Oversampling**:
  - Choose all negative records
  - Replicate positive records until both sets have equal number of records
- Problem: if data is noisy, noise may be replicated
- Added examples: provide no new information
- But: prevent learning algorithm from pruning important parts of the model because of not enough data points



Replicate positive class

CLASSIFICATION – MULTICLASS CLASSIFICATION

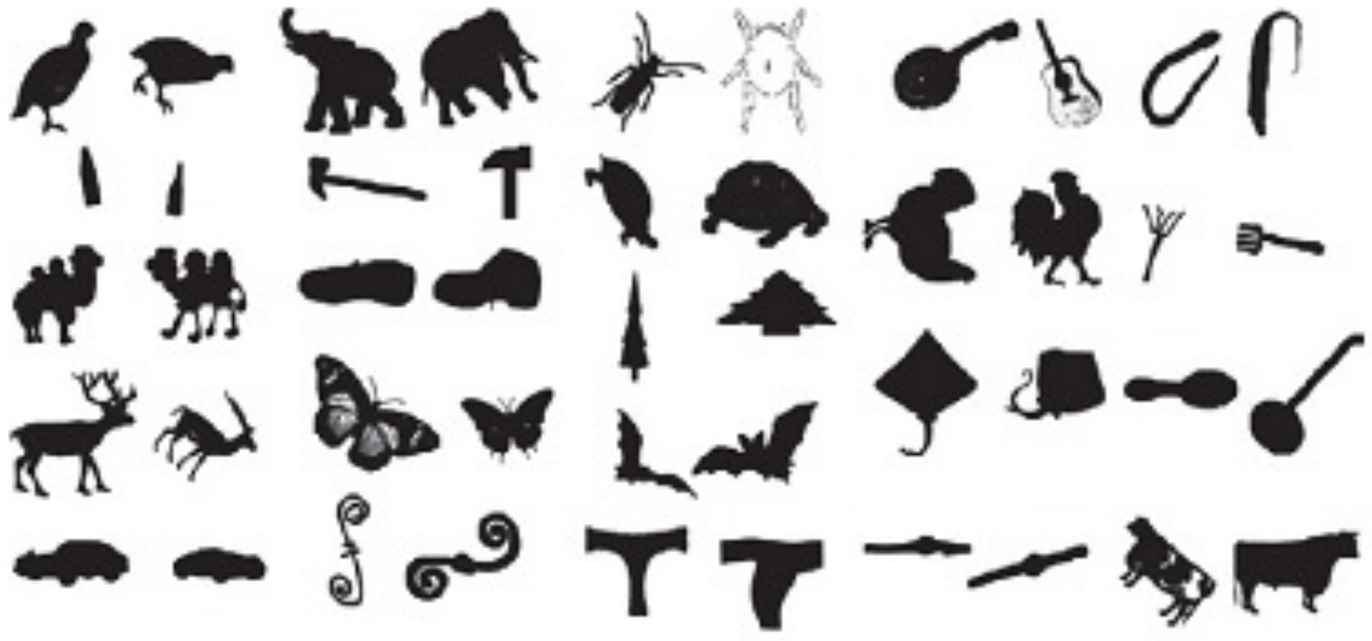
Multiclass Classification

- Character recognition

z a q v u n

Multiclass Classification

- Image recognition



http://www.cis.temple.edu/~latecki/research.html

Multiclass Classification Approaches

- One versus All (OVA)
- One versus One (OVO)
- Error correcting codes

One Versus All

- $Y = \{y_1, y_2, \ldots, y_K\}$: the set of class labels
- Classifier building:
  - For each y_i, create a binary problem such that:
    - Instances belonging to y_i are positive
    - Instances not belonging to y_i are negative
- Tuple Classification:
  - Classify the tuple using each classifier
  - If classifier i returns a positive label, y_i gets one vote
  - If classifier i returns a negative label, all classes except y_i get a vote
  - Assign the class with the most votes

One Versus All - Example

Input Instances

X1	A
X2	B
X3	A
X4	C
X5	C
X6	D
X7	B
X8	A

Instances for C_A

X1	+
X2	-
X3	+
X4	-
X5	-
X6	-
X7	-
X8	+

Instances for C_B

X1	-
X2	+
X3	-
X4	-
X5	-
X6	-
X7	+
X8	-

Instances for C_C

X1	-
X2	-
X3	-
X4	+
X5	+
X6	-
X7	-
X8	-

Instances for C_D

X1	-
X2	-
X3	-
X4	-
X5	-
X6	+
X7	-
X8	-

One Versus All - Example

Classify test tuple X: (-, +, -, -)

	C_A	C_B	C_C	C_D	
	-	+	-	-	Votes
A			1	1	2
B	1	1	1	1	4
C	1			1	2
D	1		1		2

Classification results through all the One vs. All classifiers

Classify test tuple X: (+, -, +, -)

	C_A	C_B	C_C	C_D	
	+	-	+	-	Votes
A	1	1		1	3
B				1	1
C		1	1	1	3
D		1			1

Randomly break the tie

One Versus One

- $Y = \{y_1, y_2, \ldots, y_K\}$: the set of class labels
- Classifier building:
  - For each pair y_i and y_j create a binary problem:
    - Keep instances belonging to y_i and y_j
    - Ignore other instances
- Tuple Classification:
  - Classify the tuple using each classifier C_{ij}
  - If classifier C_{ij} returns *i* label, y_i gets one vote
  - If it returns *j*, y_j gets one vote
  - Assign the class with the most votes

One Versus One - Example

Input Instances

X1	A
X2	B
X3	A
X4	C
X5	C
X6	D
X7	B
X8	A

Instances for C_{AB}

X1	A
X2	B
X3	A
X7	B
X8	A

Instances for C_{AC}

X1	A
X3	A
X4	C
X5	C
X8	A

Instances for C_{AD}

X1	A
X3	A
X6	D
X8	A

Instances for C_{BC}

X2	B
X4	C
X5	C
X7	B

Instances for C_{BD}

X2	B
X6	D
X7	B

Instances for C_{CD}

X4	C
X5	C
X6	D

One Versus One - Example

- Classify test tuple X: (B, A, D, B, D, D)

	AB	AC	AD	BC	BD	CD	
R_X	B	A	D	B	D	D	Votes
A		1					1
B	1			1			2
C							0
D			1		1	1	3

Characteristics

- One vs All:
  - Builds k classifiers for a k class problem
  - Full training set for each classifier
- One vs One:
  - Builds k(k-1)/2 classifiers
  - Subset of training set for each classifier
- Sensitive to binary classification errors

Error correcting codes

- Idea: Add redundancy to increase chances of detecting errors
- Training:
  - Represent each y_i by a unique n bit codeword
  - Build n binary classifiers, each to predict one bit
- Testing
  - Run each classifier on the test instance to predict its bit vector
  - Assign, to the test instance, the codeword with the closest Hamming distance to the output codeword
- Hamming distance: number of bits that differ

Example

- Given: $Y = \{y_1, y_2, y_3, y_4\}$
- Encode each y_i as:

Class	Codeword						
y_1	1	1	1	1	1	1	1
y_2	0	0	0	0	1	1	1
y_3	0	0	1	1	0	0	1
y_4	0	1	0	1	0	1	0

- Need to train 7 classifiers
  - Generate 7 training sets.
  - For example, given Record $<X, y_2>$, add:
    - <X, 0> in the training set of classifiers 1..4
    - <X, 1> in the training set for 5..7

Data transformation - Example

Input Instances

X1	y2
X2	y3

y_2	0	0	0	0	1	1	1
y_3	0	0	1	1	0	0	1

Instances for C_1

X1	0
X2	0

Instances for C_2

X1	0
X2	0

Instances for C_3

X1	0
X2	1

Instances for C_4

X1	0
X2	1

Instances for C_5

X1	1
X2	0

Instances for C_6

X1	1
X2	0

Instances for C_7

X1	1
X2	1

Example:

- Test instance result: (0, 1, 1, 1, 1, 1, 1)

Test	0	1	1	1	1	1	1
y_1	1	1	1	1	1	1	1
D	1	0	0	0	0	0	0

Hamming Distance = 1

Test	0	1	1	1	1	1	1
y_2	0	0	0	0	1	1	1
D	0	1	1	1	0	0	0

Hamming Distance = 3

Test	0	1	1	1	1	1	1
y_3	0	0	1	1	0	0	1
D	0	1	0	0	1	1	0

Hamming Distance = 3

Test	0	1	1	1	1	1	1
y_4	0	1	0	1	0	1	0
D	0	0	1	0	1	0	1

Hamming Distance = 3

Classify as y_1

Design issues

- How to design the appropriate set of codewords for each class
- Minimum codeword length to represent k classes $n = \log_2 k$
- It is required that both the row-wise and column-wise separation are large
  - Each individual codeword should be separated from each of the other codewords with a large Hamming distance
  - Large row-wise separation: more tolerance for errors
  - Large column wise separation: binary classifiers are mutually independent

Exam 1 (10/8)

No Textbook;
No Notes;
No Slides;
No ChatGPT

- Week 1 to Week 6
  - Preprocessing
  - Classification
  - Association Mining

1-Introduction
2-Data Preprocessing (Part 1)
3-Data Preprocessing (Part 2)
4-Classification (Decision Trees)
5-Classification (SVM)
6-Classification (Naive Bayes)
7-Classification (KNN)
8-Classification (Neural Networks)
9-Classification (Ensemble; Classifier Comparison)
10-Classification (Class imbalance; Multi-class)
11 + 12: Association Mining (Next week)

- Textbook to refer for preparation
  - Tan et al. 1st edition (Ch. 1-5, 6.1-6.3, 7.1-7.3)
  - Tan et al. 2nd edition (Ch. 1-4, 5.1-5.3, 6.1-6.3)
  - Shmueli et al. 3rd edition (2.2, 4.1-4.8, 5.3, Ch. 7-9.6, Ch. 11, 13.1, 14.1)

Exam 1 (10/8)

- Question types:
  - Multiple choice
  - True/false
  - Short answer

- Kinds of questions:
  - Definitions
  - When to use technique

Example Question:

- **What is underfitting and how do you overcome it?**

- **What are training, validation, and test sets, and why is it important to distinguish between them?**

- **All classification algorithms are equally effective across various datasets.**
  True or False?

Exam 1 (10/8)

- **Not** on the exam
  - Memorization of formulas
  - Solving formulas
  - Deep learning