DAT630
Classification
Alternative Techniques

Introduction to Data Mining, Chapter 5

14/09/2015

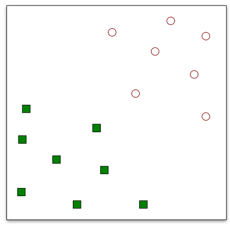
Krisztian Balog | University of Stavanger

Outline

- Alternative classification techniques
  - Rule-based
  - Nearest neighbors
  - Naive Bayes
  - SVM
  - Ensemble methods
  - Artificial neural networks
- Class imbalance problem
- Multiclass problem

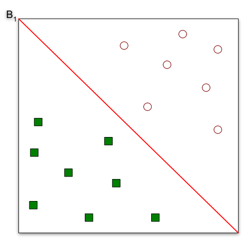
Support Vector Machine

Support Vector Machine (SVM)

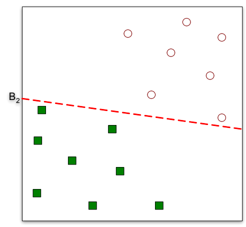


- Find a linear hyperplane (decision boundary) that will separate the data

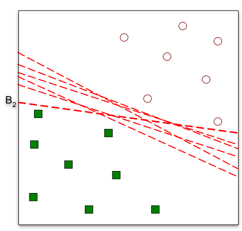
One possible solution



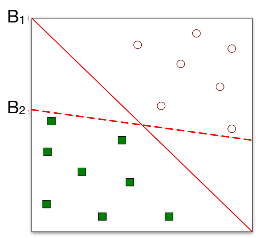
Another possible solution



Other possible solutions

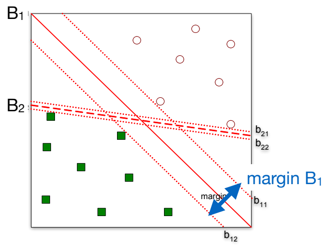


Which one is better? B_1 or B_2?



- How do you define better?

Max. Margin Hyperplanes



- Find the hyperplane that **maximizes** the margin

Rationale

- Decision boundaries with large margins tend to have better generalization errors
  - If the margin is small, any slight perturbation to the decision boundary can have a significant impact on classification
  - Small margins are more suspectible to overfitting
- A more formal explanation can be obtained using structural risk minimization

Linear SVM Separable Case

- Search for a hyperplane with the largest margin
  - Also known as **maximal margin classifier**
- Key concepts
  - Linear decision boundary
  - Margin
- Binary classification problem with N training examples
  - Each example is a tuple (**x_i**, y_i), where x_i corresponds to the attribute set for the *i*th example
  - Class label *y* by convention is -1 or 1

Remember

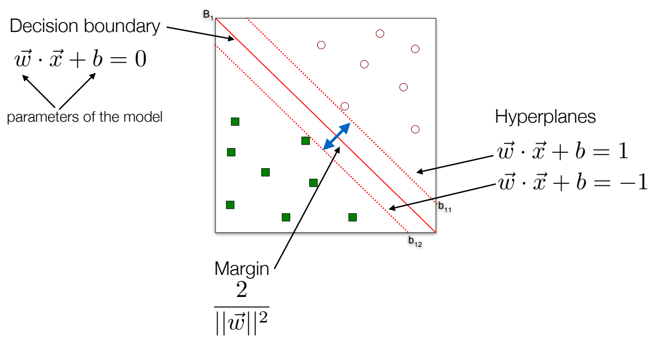
- Dot product of two vectors (of equal length)

$$\vec{a} \cdot \vec{b} = \sum_{i=1}^{n} a_i b_i$$

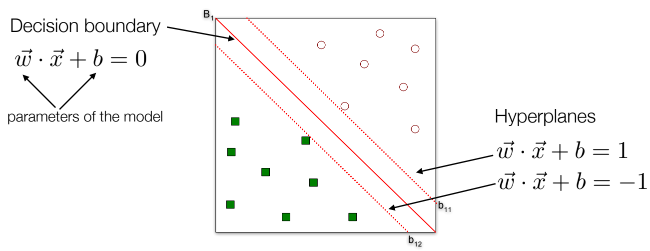
- Dot product of a vector with itself

$$\vec{a} \cdot \vec{a} = ||\vec{a}||^2$$

Key Concepts



Predicting the Class Label



Predicting the class label y for a test example z

$$y = \begin{cases} 1 & if \ \vec{w} \cdot \vec{z} + b \geq 0 \\ -1 & if \ \vec{w} \cdot \vec{z} + b < 0 \end{cases}$$

Learning the Model

- Estimating the parameters w and b of the decision boundary from training data
- Maximalizing the margin of the decision boundary
  - Equivalent to **minimizing** the **objective function**

$$L(w) = \frac{||\vec{w}||^2}{2}$$

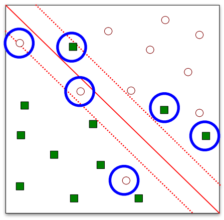
- Subjected to the following constraints
  - All training instances classified correctly

$$f(\vec{x_i}) = \begin{cases} 1 & if \ \vec{w} \cdot \vec{x_i} + b \geq 1 \\ -1 & if \ \vec{w} \cdot \vec{x_i} + b \leq -1 \end{cases}$$

Learning the Model

- Constrained optimization problem
- Numerical approaches are used to solve it
  - Lagrange multiplier method
  - Karush-Kuhn-Tucker conditions
  - …

What if the problem is not linearly separable?



Linear SVM Nonseparable Case

- Learn a decision boundary that is tolerable to small training errors by using a **soft margin** approach
- Construct a linear decision boundary even in situations where the classes are not linearly separable
- Consider the trade-off between the width of the margin and the number of training errors committed by the linear decision boundary

Nonseparable Case

- Introduce slack variables
- Need to minimize

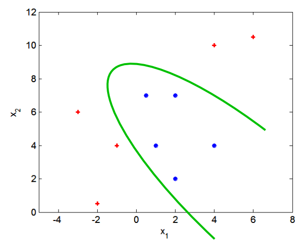
user-specified parameters (penalty for misclassifying the traiing instances)

$$L(w) = \frac{||\vec{w}||^2}{2} + C(\sum_{i=1}^{N} \zeta_i)^k$$

- Subject to

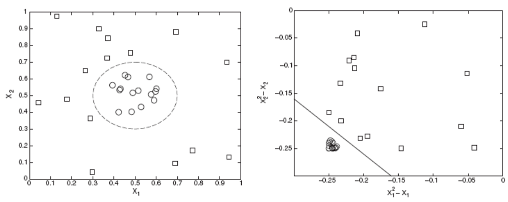
$$f(\vec{x_i}) = \begin{cases} 1 & if\ \vec{w} \cdot \vec{x_i} + b \geq 1 - \zeta_i \\ -1 & if\ \vec{w} \cdot \vec{x_i} + b \leq -1 + \zeta_i \end{cases}$$

What if the decision boundary is not linear?



Nonlinear SVM

- Trick: transform data from original coordinate space in **x** into a new space $\Phi(x)$ so that a linear decision boundary can be used



(a) Decision boundary in the original two-dimensional space.

(b) Decision boundary in the transformed space.

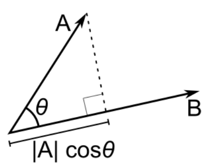
Problems with Transformation

- Not clear what type of mapping should be used to ensure that a linear decision boundary can be constructed in the transformed space
- Even if the appropriate mapping function is known, solving the constrained optimization problem in the high-dimensional feature space is computationally expensive

The dot product

- The dot product is often regarded as a measure of similarity between two input vectors
- Geometrical interpretation

$$A \cdot B = ||A||\ ||B|| \cos \theta$$



Kernel trick

- The dot product can also be regarded as a measure of similarity in the transformed space
- The kernel trick is a method for computing similarity in the transformed space using the original attribute set
  - The similarity function K which is computed in the original attribute space is known as the **kernel function**

Kernel functions

- **Mercer's theorem** ensures that the kernel functions can always be expressed as the dot product between two input vectors in some high-dimensional space
- Examples $K(\vec{x},\vec{y}) = (\vec{x}\cdot\vec{y}+1)^p$
  $K(\vec{x},\vec{y}) = tanh(k\vec{x}\cdot\vec{y}-\delta)$
- Computing the dot products using kernel functions is considerably cheaper than using the transformed attribute set

Example

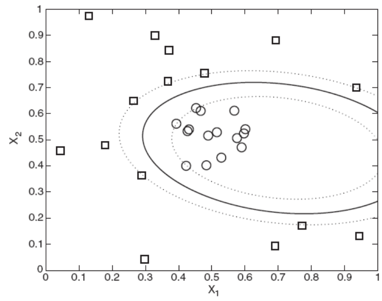


Figure 5.29. Decision boundary produced by a nonlinear SVM with polynomial kernel.

Categorical attributes

- SVM can be applied to categorical attributes by introducing "dummy" variables for each categorical attribute value
  - E.g., Martial status = {Single, Married, Divorced}
  - Three binary attributes: isSingle, isMarried, isDivorced

Summary

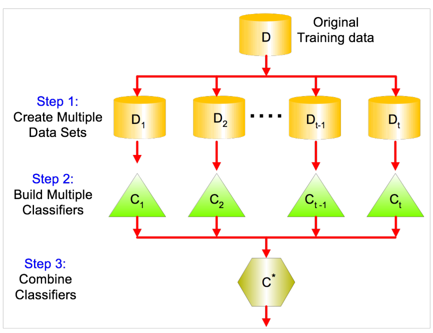
- SVM is one of the most widely used classification algorithms
- The learning problem is formulated as a convex optimization problem
  - Possible to find global minimum of the objective function as opposed to other classification methods
- User parameters
  - Type of kernel function
  - Cost function (C) for introducing each slack variable

Ensemble Methods

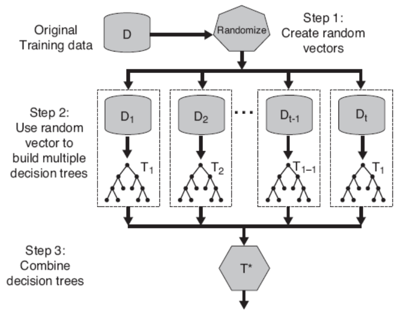
Ensemble Methods

- Construct a set of classifiers from the training data
- Predict class label of previously unseen records by aggregating predictions made by multiple classifiers

General Idea



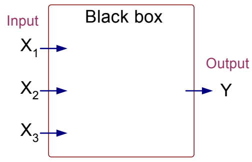
Random Forests



Artificial Neural Networks

Artificial Neural Networks (ANN)

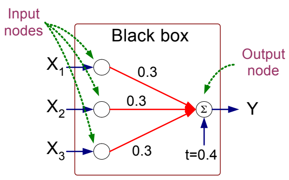
X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0



Output Y is 1 if at least two of the three inputs are equal to 1.

Artificial Neural Networks (ANN)

X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0

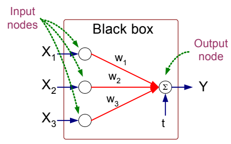


$$Y = I(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4 > 0)$$

$$\text{where } I(z) = \begin{cases} 1 & \text{if } z \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

Artificial Neural Networks (ANN)

- Model is an assembly of inter-connected nodes and weighted links
- Output node sums up each of its input value according to the weights of its links
- Compare output node against some threshold t

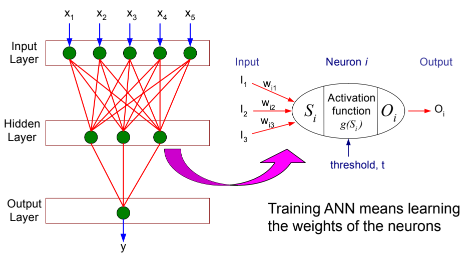


Perceptron Model

$$Y = I(\sum_i w_i X_i - t) \quad \text{or}$$

$$Y = sign(\sum_i w_i X_i - t)$$

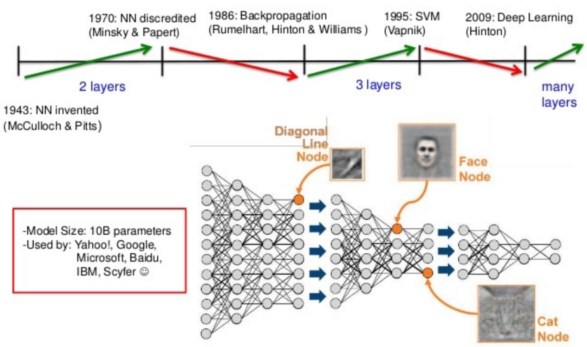
General Structure of ANN



Summary

- Choosing the appropriate topology is important
- Can handle redundant features well
- Sensitive to the presence of noise
- Training is a time consuming process

Deep Learning: Neural Nets Strike Back(again)



Class Imbalance Problem

Class Imbalance Problem

- Data sets with imbalanced class distributions are quite common in real-world applications
  - E.g., credit card fraud detection
- Correct classification of the rare class has often greater value than a correct classification of the majority class
- The accuracy measure is not well suited for imbalanced data sets
- **We need alternative measures**

Confusion Matrix

		Predicted class	
		Positive	**Negative**
Actual class	**Positive**	**True Positives (TP)**	**False Negatives (FN)**
	Negative	**False Positives (FP)**	**True Negatives (TN)**

Additional Measures

- **True positive rate** (or **sensitivity**)
  - Fraction of positive examples predicted correctly

$$TPR = \frac{TP}{TP + FN}$$

- **True negative rate** (or **specificity**)
  - Fraction of negative examples predicted correctly

$$TNR = \frac{TN}{TN + FP}$$

Additional Measures

- **False positive rate**
  - Fraction of negative examples predicted as positive

$$FPR = \frac{FP}{TN + FP}$$

- **False negative rate**
  - Fraction of positive examples predicted as negative

$$FNR = \frac{FN}{TP + FN}$$

Additional Measures

- **Precision**
  - Fraction of positive records among those that are classified as positive

$$P = \frac{TP}{TP + FP}$$

- **Recall**
  - Fraction of positive examples correctly predicted (same as the true positive rate)

$$R = \frac{TP}{TP + FN}$$

Additional Measures

- **F1-measure**
  - Summarizing precision and recall into a single number
  - Harmonic mean between precision and recall

$$F1 = \frac{2RP}{R + P}$$

Multiclass Problem

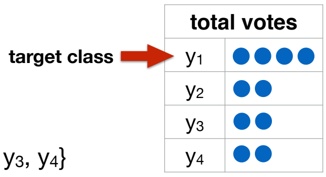
Multiclass Classification

- Many of the approaches are originally designed for binary classification problems
- Many real-world problems require data to be divided into more than two categories
- Two approaches
  - One-against-rest (1-r)
  - One-against-one (1-1)
- Predictions need to be combined in both cases

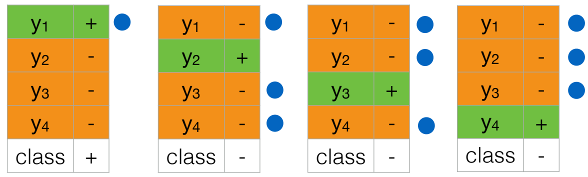
One-against-rest

- $Y=\{y_1, y_2, \dots y_K\}$ classes
- For each class y_i
  - Instances that belong to y_i are positive examples
  - All other instances are negative examples
- Combining predictions
  - If an instance is classified positive, the positive class gets a vote
  - If an instance is classified negative, all classes except for the positive class receive a vote

Example



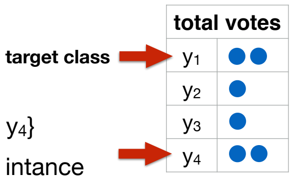
- 4 classes, $Y=\{y_1, y_2, y_3, y_4\}$
- Classifying a given test intance



One-against-one

- $Y=\{y_1, y_2, \dots y_K\}$ classes
- Construct a binary classifier for each pair of classes (y_i, y_j)
  - $K(K-1)/2$ binary classifiers in total
- Combining predictions
  - The positive class receives a vote in each pairwise comparison

Example



- 4 classes, $Y=\{y_1, y_2, y_3, y_4\}$
- Classifying a given test intance