

Introduction to Classification

Greg Grudic

September 7, 2004

1 Basic Definitions and Probability Formulas

1. **Probability of an Event:** Let A be an event (e.g. $A = it\ rains\ tomorrow$). Then the probability of A occurring is symbolized $\Pr(A)$, where $0 \leq \Pr(A) \leq 1$.
2. **Posterior Probability:** Assume there exist two events A and B . Then the posterior probability of event A given event B is simply that probability that A will occur given that B has occurred. This is symbolized by $\Pr(A|B)$, where $0 \leq \Pr(A|B) \leq 1$.
3. **Product Rule:** Assume two events and called them A and B . The probability that both A and B will occur is symbolized by $\Pr(A \cap B)$, where

$$\Pr(A \cap B) = \Pr(A|B) \Pr(B) = \Pr(B|A) \Pr(A) \quad (1)$$

This is often referred to as the *conjunction* of two events.

4. **Sum Rule:** Assume two events and called them A and B . The probability that either event will occur is symbolized by $\Pr(A \cup B)$ where

$$\Pr(A \cup B) = \Pr(A) + \Pr(B) - \Pr(A \cap B) \quad (2)$$

This is often referred to as the *disjunction* of two events.

5. **Bayes Theorem:** Assume two events and called them A and B . The posterior probability $\Pr(A|B)$ of A given B is

$$\Pr(A|B) = \frac{\Pr(B|A) \Pr(A)}{\Pr(B)} \quad (3)$$

6. **Theorem of Total Probability:** Assume an event B and a set of events A_1, \dots, A_n . If A_1, \dots, A_n are mutually exclusive with $\sum_{i=1}^n \Pr(A_i) = 1$ then

$$\Pr(B) = \sum_{i=1}^n \Pr(B|A_i) \Pr(A_i) \quad (4)$$

2 Assumptions on Data

Let $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$ be a set of N data examples. This data is assumed to be generated from two or more classes as defined below.

2.1 Binary Classification

A point $\mathbf{x} \in D \subseteq \mathfrak{R}^d$ can belong to one of *TWO* classes (hence), which we will label by $y \in \{-1, +1\}$. Assume the prior probability of class $y = 1$ is p_+ and the prior probability of class $y = -1$ is p_- , where $0 < p_+ < 1$, $0 < p_- < 1$, and $p_+ + p_- = 1$. Assume that class $y = 1$ is independently identically distributed (i.i.d.) from the probability density function (pdf) $h_+(\mathbf{x})$, where $h_+(\mathbf{x}) \geq 0$, $\forall \mathbf{x} \in D$, and $\int_D h_+(\mathbf{x})d\mathbf{x} = 1$. Similarly, assume that class $y = -1$ is i.i.d. from the pdf $h_-(\mathbf{x}) \geq 0$, where $h_-(\mathbf{x})$, $\forall \mathbf{x} \in D$, and $\int_D h_-(\mathbf{x})d\mathbf{x} = 1$. Then, the posterior probability of class $y = 1$ given \mathbf{x} is:

$$\Pr(y = 1 | \mathbf{x}) = \frac{\Pr(y=1)\Pr(\mathbf{x}|y=1)}{\Pr(y=1)\Pr(\mathbf{x}|y=1)+\Pr(y=-1)\Pr(\mathbf{x}|y=-1)} = \frac{p_+h_+(\mathbf{x})}{p_+h_+(\mathbf{x})+p_-h_-(\mathbf{x})} \quad (5)$$

And, the posterior probability of class $y = -1$ given \mathbf{x} is:

$$\Pr(y = -1 | \mathbf{x}) = 1 - \Pr(y = +1 | \mathbf{x}) = \frac{p_-h_-(\mathbf{x})}{p_+h_+(\mathbf{x})+p_-h_-(\mathbf{x})}$$

2.2 Multi-Class Classification

Multi-Class classification assumes more than two classes. We can symbolize K classes by $y \in \{c_1, \dots, c_K\}$. The prior probability of each class is symbolized by p_k and it is assumed to be iid from a pdf distribution $h_k(\mathbf{x})$. Then, the posterior probability of class $y = c_k$ given \mathbf{x} is:

$$\Pr(y = c_k | \mathbf{x}) = \frac{p_k h_k(\mathbf{x})}{\sum_{i=1}^K p_i h_i(\mathbf{x})} \quad (6)$$

3 Classification Models

The goal of classification is to find a model that minimizes the number of classification errors on future or test data (i.e. *data that has not been used to build the classification model*). A classification model is defined by $M(\mathbf{x})$ and a class prediction for an input \mathbf{x} is given by:

$$\hat{y} = M(\mathbf{x}) \quad (7)$$

Given a set of M test points defined by $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_M, y_M)\}$, the class prediction for each point is given by

$$\hat{y}_i = M(\mathbf{x}_i) \quad (8)$$

The model error rate is:

$$\text{error} = \frac{1}{M} \sum_{i=1}^M \delta(\hat{y}_i \neq y_i)$$

where,

$$\delta(\hat{y}_i \neq y_i) = \begin{cases} 1 & \text{if } \hat{y}_i \neq y_i \\ 0 & \text{otherwise} \end{cases}$$

3.1 Generative Models

If the prior for each class, p_k , and the pdf function, $h_k(\mathbf{x})$, are known, then a classifier that minimizes the classification error rate is given by:

$$\hat{y} = \arg \max_{C_k} \{p_k h_k(\mathbf{x})\}$$

This is called the *Maximum a Posteriori* (MAP) classifier, and we will cover this in class in more detail when we study Bayesian Learning. The MAP classifier can be readily derived from Equation (6).

MAP theory is the main motivation behind generative classification models. A generative model is one that attempts to build models of p_k and $h_k(\mathbf{x})$ from the training data. These approximations will be denoted by \hat{p}_k and $\hat{h}_k(\mathbf{x})$ respectively, and a generative model has the form¹:

$$\hat{y} = \arg \max_{C_k} \{\hat{p}_k \hat{h}_k(\mathbf{x})\}$$

Estimating the prior p_k from training data is relatively simple and is done as follows:

$$\hat{p}_k = \frac{1}{N} \sum_{i=1}^N \delta(y_i = c_k)$$

However, estimating $\hat{h}_k(\mathbf{x})$ can be challenging and usually requires many assumptions. The common frameworks for estimating $\hat{h}_k(\mathbf{x})$ are described next.

3.1.1 The Single Gaussian Assumption

One common assumption is to model each class as being generated by a multivariate Gaussian density:

$$\hat{h}_k(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^d |\hat{\Sigma}_{kj}|}} \exp \left[-\frac{1}{2} (\mathbf{x} - \hat{\mu}_k)^T \hat{\Sigma}_k^{-1} (\mathbf{x} - \hat{\mu}_k) \right]$$

where d is the dimension of the problem, $\hat{\mu}_k$ is a vector containing the mean values of all the inputs for class k , $\hat{\Sigma}_k$ is the covariance matrix of class k , and $|\hat{\Sigma}_k|$ is the determinant of $\hat{\Sigma}_k$. $\hat{\Sigma}_k$ and $\hat{\mu}_k$ can both be estimated from data (see Chapter 4.3 of [1]). This Gaussian assumption leads to two commonly used classification models: Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA).

3.1.2 The Mixture of Gaussians Assumption

The Gaussian Mixture Model uses $J > 1$ Gaussian distributions and takes on the following form:

$$\hat{h}_k(\mathbf{x}) = \frac{1}{J} \sum_{j=1}^J \left\{ \frac{1}{\sqrt{(2\pi)^d |\hat{\Sigma}_{kj}|}} \exp \left[-\frac{1}{2} (\mathbf{x} - \hat{\mu}_{kj})^T \hat{\Sigma}_{kj}^{-1} (\mathbf{x} - \hat{\mu}_{kj}) \right] \right\}$$

¹This is a generative classifier because knowing \hat{p}_k and $\hat{h}_k(\mathbf{x})$ allows us to actually generate data.

where d is the dimension of the problem, $\hat{\mu}_{kj}$ is the center of Gaussian j for class k , $\hat{\Sigma}_{kj}$ is the covariance matrix of Gaussian j of class k . The covariances $\hat{\Sigma}_{kj}$ and means $\hat{\mu}_{kj}$ are usually estimated using local neighborhoods of data. A simpler implementation of this is termed the Parzen Window Density Estimator (see Chapter 6.6 of [1] for more details), which has the form:

$$\hat{h}_k(\mathbf{x}) = \frac{1}{J} \sum_{j=1}^J \left\{ \frac{1}{\sqrt{(2\sigma_{kj}^2\pi)^d}} \exp \left[-\frac{1}{2\sigma_{kj}^2} \|\mathbf{x} - \hat{\mu}_{kj}\|^2 \right] \right\}$$

where σ_{kj} is the standard deviation at Gaussian j of class k , which computationally much easier to estimate than the full covariance matrix.

We will discuss Gaussian Mixture Models further when we cover Bayesian learning algorithms.

3.1.3 Why Generative Models?

One desirable property of a generative model is that it directly computes estimates of $\Pr(y = c_k | \mathbf{x})$ - this is important in many applications, especially medical (i.e. what is the probability that a patient has cancer given a set of lab test results). Another is that if we have good prior knowledge about $\hat{h}_k(\mathbf{x})$, relatively few training data points are required to obtain good classification models. Because of this generative models are currently being widely applied in bioinformatics domains where data is very expensive to generate.

3.2 Discriminative Models

A Discriminative Classification Model starts with the premise that modelling $\hat{h}_k(\mathbf{x})$ is too difficult, and in fact unnecessary. In order to build a classifier, all I need to model is the decision (or discriminating) boundary between the classes - I need not bother with density estimation at all. If we assume a binary classification problem with classes labelled by $y \in \{-1, +1\}$, then a linear classifier can be defined by:

$$\hat{y} = M(\mathbf{x}) = \text{sgn} \left[\hat{\beta}_0 + (\hat{\beta}_1, \dots, \hat{\beta}_d) \mathbf{x}^T \right]$$

where

$$\text{sgn}[A] = \begin{cases} 1 & \text{if } A \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

Therefore the boundary between two classes is defined by the following:

$$\hat{\beta}_0 + (\hat{\beta}_1, \dots, \hat{\beta}_d) \mathbf{x}^T = 0$$

Many algorithms have been proposed for estimating the coefficients $(\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_d)$ from training data. In the next section, we will discuss one of the earlier ones.

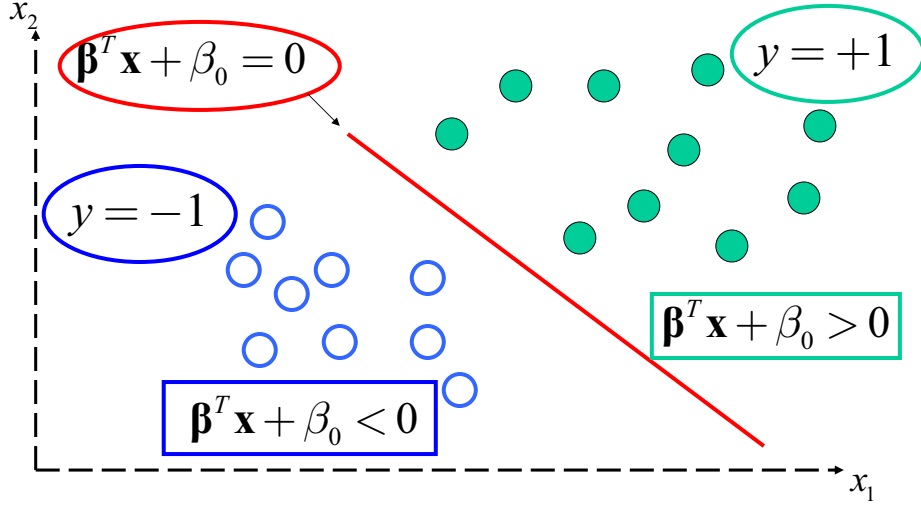


Figure 1: Separating Hyperplane for a Binary Classification Problem

3.2.1 Rosenblatt's Perceptron Learning Algorithm

The perceptron algorithm dates back to the 1950's and is the motivation behind modern neural network algorithms. The algorithm works by minimizing the distance of misclassified point to the decision boundary - i.e. it minimizes the following function:

$$D(\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_d) = - \sum_{i \in M} y_i (\hat{\beta}_0 + (\hat{\beta}_1, \dots, \hat{\beta}_d) \mathbf{x}^T)$$

where M is the set of misclassified points. This function is minimized by gradient descent on the $(\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_d)$ coefficients. The gradients are defined as follows: for $\hat{\beta}_0$

$$\frac{\partial D(\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_d)}{\partial \hat{\beta}_0} = - \sum_{i \in M} y_i$$

and for $(\hat{\beta}_1, \dots, \hat{\beta}_d)$ (i.e. $j = 1, \dots, d$)

$$\frac{\partial D(\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_d)}{\partial \hat{\beta}_j} = - \sum_{i \in M} y_i x_{ij}$$

This leads to the following update rule for the coefficients:

$$\begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_d \end{pmatrix} \leftarrow \begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_d \end{pmatrix} + \rho \begin{pmatrix} y_i \\ y_i x_{i1} \\ \vdots \\ y_i x_{id} \end{pmatrix}$$

Where $\rho > 0$ is the learning rate and can be arbitrarily set to 1. This algorithm has an number of interesting properties. If the problem is linearly separable, the algorithm will converge to a separating hyperplane in a finite number of steps. However, it also has some fundamental problems, namely (see [1], chapter 4.5):

1. If the data is not separable, the algorithm will cycle forever.
2. If the data is separable, the finite number of steps can be very large (depends on the size of the gap between the two classes).
3. There are infinitely many hyperplanes that satisfy the minimization criteria when the data is separable. Which one is best? One solution to this problem is proposed by Support Vector Theory, which we will study later.

3.2.2 Nonlinear Separating Boundaries

When the data is *not* linearly separable, nonlinear models can be used:

$$\hat{y} = M(\mathbf{x}) = \text{sgn} \left[\hat{\beta}_0 + \left(\hat{\beta}_1, \dots, \hat{\beta}_d \right) \Phi(\mathbf{x})^T \right]$$

Where, as in regression, the p basis functions $\Phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \dots, \phi_p(\mathbf{x}))^T$ are nonlinear functions of the inputs \mathbf{x} . Examples include $\phi_1(\mathbf{x}) = x_1x_2$, $\phi_2(\mathbf{x}) = x_1^2$, $\phi_3(\mathbf{x}) = \sin(x_3)$, etc.

A commonly used basis function is a kernel, which, as in regression, has the following form:

$$\varphi_i(\mathbf{x}) = K(\mathbf{x}_i, \mathbf{x})$$

where \mathbf{x}_i for $i = 1, \dots, N$ are training example inputs. Therefore, the kernel model looks like:

$$\hat{y} = \hat{\beta}_0 + \sum_{i=1}^N \hat{\beta}_i K(\mathbf{x}_i, \mathbf{x})$$

Typically used kernel functions include the Gaussian Kernel:

$$K(\mathbf{a}, \mathbf{b}) = \exp \left[-\frac{\|\mathbf{a} - \mathbf{b}\|^2}{2\sigma^2} \right]$$

where $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$, the kernel parameter is $\sigma > 0$, and

$$\|\mathbf{a} - \mathbf{b}\|^2 = \sum_{j=1}^d (a_j - b_j)^2$$

The polynomial kernel:

$$K(\mathbf{a}, \mathbf{b}) = (q + \mathbf{a}^T \mathbf{b})^k$$

where $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$, the kernel parameter are $k = 1, 2, 3, \dots$ and $q \in \mathbb{R}$. The sigmoid Kernel:

$$K(\mathbf{a}, \mathbf{b}) = \tanh(\kappa \mathbf{a}^T \mathbf{b} + \theta)$$

where $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$, the kernel parameter are $\kappa \in \mathbb{R}$ and $\theta \in \mathbb{R}$.

4 Practical Considerations

There are many practical considerations when building classification models. These include:

1. Scale all inputs to the same range. Usually -1 to $+1$ before building the model.
2. Get ride of highly correlated inputs.
3. Feature (i.e. input) selection. Features that do not improve the model should be discarded.

Does this list seem familiar?

5 Other Classification Formulations

We will cover a number of other classification formulations in class. These include Gaussian Process Classification, Support Vector Machines, Classification Trees, Nearest Neighbor Classification, bagging, boosting, and random forests.

References

- [1] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: data mining, inference and prediction*. Springer, 2001.