# 7 Classification

This chapter follows closely: http://cs229.stanford.edu/notes/cs229-notes2.pdf

## 7.1 MLE of Bernoulli model

An important special case of learning problems is **classification** in which features are mapped to a finite number of possible categories. We have previously used general learning machines such as MLPs and SVMs to learn models to predict the class membership from features. We are now going back to more specific probabilistic models. We are again mainly discussing **binary classification**, which is the case of two target classes where the target function (y-values) has only two possible values such as 0 and 1.

Let us first consider a random number which takes the value of 1 with probability $\phi$ and the value 0 with probability $1-\phi$ (the probability of being either of the two choices has to be 1.). That is,

$$p(y) = \phi^y(1-\phi)^{1-y} \tag{7.1}$$

Such a random variable is called Bernoulli distributed, and all Bernoulli distributions are characterized by one parameter $\phi$. Tossing a coin is a good example of a Bernoulli process (a process of generating such random numbers). We can use maximum likelihood estimation (MAP with uniform prior) to estimate the parameter $\phi$ from such trials. That is, let us consider $m$ tosses in which $h$ heads have been found. The log-likelihood of such $m$ trials is

$$\begin{aligned} l(\phi) &= log\prod_i \phi^{y^{(i)}}(1-\phi)^{1-y^{(i)}} && (7.2)\\ &= \log(\phi^h(1-\phi)^{m-h}) && (7.3)\\ &= h\log(\phi) + (m-h)\log(1-\phi). && (7.4)\end{aligned}$$

To find the maximum with respect to $\phi$ we set the derivative of $l$ to zero,

$$\begin{aligned} \frac{\mathrm{d}l}{\mathrm{d}\phi} &= \frac{h}{\phi} - \frac{m-h}{1-\phi} && (7.5)\\ &= \frac{h}{\phi} - \frac{m-h}{1-\phi} && (7.6)\\ &= 0 && (7.7)\\ \rightarrow \quad \phi &= \frac{h}{m} && (7.8)\end{aligned}$$

As you might have expected, the maximum likelihood estimate of the parameter $\phi$ is the fraction of heads in $m$ trials.

## 7.2 Logistic regression (again)

Of course, we usually consider the case of classification when the parameter $\phi$, depends on an attribute $x$, as usual in a stochastic (noisy) way. More specifically,

$$p(y = 1|\mathbf{x}; \theta) = h(\mathbf{x}; \theta) \tag{7.9}$$

$$p(y = 0|\mathbf{x}; \theta) = 1 - h(\mathbf{x}; \theta), \tag{7.10}$$

where $h(\mathbf{x}; \theta$ is here a hypothesis function and not the number of heads as before. We can again combine the probabilities into one expression,

$$p(y|\mathbf{x}; \theta) = (h(\mathbf{x}; \theta))^y (1 - h(\mathbf{x}; \theta))^{1-y} \tag{7.11}$$

The corresponding log-likelihood function is

$$l(\theta) = \sum_{i=1}^{m} y^{(i)}\ log(h(\mathbf{x}; \theta)) + (1 - y^{(i)})\ log(1 - h(\mathbf{x}; \theta)). \tag{7.12}$$

To find the corresponding maximum we can use the gradient ascent algorithm, which is like the gradient descent algorithm with a changed sign,

$$\theta \leftarrow \theta + \alpha \nabla_\theta l(\theta). \tag{7.13}$$

To calculate the gradient we can calculate the partial derivative of the log-likelihood function with respect to each parameters,

$$\frac{\partial l(\theta)}{\partial \theta_j} = \left( y\frac{1}{h} - (1 - y)\frac{1}{1 - h} \right) \frac{\partial h(\theta)}{\partial \theta_j} \tag{7.14}$$

where we dropped indices for better readability.

Let us apply this again to logistic regression. An example of 100 sample points of two classes (crosses and stars) are shown in Fig.7.1. The data suggest that it is far more likely that the class is $y = 0$ for small values of $x$ and that the class is $y = 1$ for large values of $x$, and the probabilities are more similar in between. Thus, we put forward the hypothesis that the transition between the low and high probability region is smooth and qualify this hypothesis as parameterized density function known as a **logistic** or **sigmoid** function

$$h = g(\theta^T \mathbf{x}) = \frac{1}{1 + \exp(-\theta^T \mathbf{x})}. \tag{7.15}$$

As before, we can then treat this density function as function of the parameters $\theta$ for the given data values (likelihood function), and use maximum likelihood estimation to estimate values for the parameters so that the data are most likely. The density function with sigmoidal offset $\theta_0 = 2$ and slope $\theta_1 = 4$ is plotted as solid line in Fig.7.1.

We can now calculate the derivative of the hypothesis $h$ with respect to the parameters for the specific choice of the logistic functions. This is given by

$$\frac{\partial h}{\partial \theta} = \frac{\partial}{\partial \theta} \frac{1}{1 + e^{-\theta x}} \tag{7.16}$$

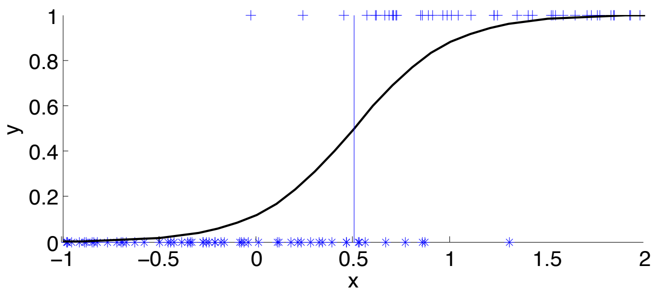

**Fig. 7.1** Binary random numbers (stars) drawn from the density $p(y=1) = \frac{1}{1+\exp(-\theta_1 x - \theta_0)}$ (solid line).

$$= \frac{1}{(1+e^{-\theta x})^2} e^{-\theta x}(-x) \tag{7.17}$$

$$= \frac{1}{(1+e^{-\theta x})}(1 - \frac{1}{(1+e^{-\theta x})})(-x) \tag{7.18}$$

$$= -h(1-h)x \tag{7.19}$$

Using this in equation 7.14 and inserting it into equation 7.13 with the identity

$$\left(y\frac{1}{h} - (1-y)\frac{1}{1-h}\right) h(1-h) = y(1-h) - (1-y)h \tag{7.20}$$

$$= y - yh - h + yh \tag{7.21}$$

$$= y - h \tag{7.22}$$

gives the learning rule

$$\theta_j \leftarrow \theta_j + \alpha(y^{(i)} - h(\mathbf{x}^{(i)}, \theta)x_j^{(i)} \tag{7.23}$$

If we think of the hypothesis function $h$ implemented by a perceptron, then we see that this is just the basic perceptron learning rule. Our derivation shows how such a learning rule relates to assumptions of a probabilistic model.

How can we use the knowledge (estimate) of the density function to do classification? The obvious choice is to predict the class with the higher probability, given the input attribute. This **bayesian decision point**, $x^t$, or **dividing hyperplane** in higher dimensions, is give by

$$p(y=1|x^t) = p(y=0|x^t) = 0.5 \rightarrow x^t\theta^T\mathbf{x^t} = 0. \tag{7.24}$$

We have here considered binary classification with linear decision boundaries as logistic regression, and we can also generalize this method to problems with non-linear decision boundaries by considering hypothesis with different functional forms, which leads to classification with deep networks as discussed before.

## 7.3 Generative models

In the previous sections we have introduced the idea that understanding the world should be based on a model of the world in a probabilistic sense. That is, building a good recognition system means estimating a large density function about labels in of objects from sensory data. This is what we have done so far; we used mainly (stochastic) models as a recognition model that take feature values $\mathbf{x}$ and make a prediction of an output (label) $y$. In the probabilistic formulation, the models where formulated as parameterized functions that represent the conditional probability $p(y|\mathbf{x};\theta)$. In other words, the aim of such a model is to discriminate between classes based on the feature values and are hence called **discriminative models**. Building a discriminative model directly from example data can be a daunting task as we have to learn how each item is distinguished from every other possible item. Indeed, we have mainly used simple models in low dimensions to illustrate the ideas, and many real world problems have rather much larger dimensions.

A different strategy, which seems much more resembling human learning, is to learn first about the nature of specific classes and then use this knowledge when faced with a classification task. For example, we might first learn about chairs, and independently about tables, and when we are shown pictures with different furnitures we can draw on our knowledge to classify them. Thus, in this chapter we start discussing **generative models** of individual classes, given by $p(\mathbf{x}|y;\theta)$.

Generative models can be useful in its own right, and are also important to guide learning as discussed later, but for now we are mainly interested in using these models for classification. Thus, we need to ask how we can combine the knowledge about the different classes to do classification. Of course, the answer is provided by Bayes' theorem. In order to make a discriminative model from the generative models, we need to the **class priors know**, e.g. what the relative frequencies of the classes is, and can then calculate the probability that an item with features $\mathbf{x}$ belong to a class $y$ as

$$p(y|\mathbf{x};\theta) = \frac{p(\mathbf{x}|y;\theta)p(y)}{p(x)}. \tag{7.25}$$

We can use this directly in the case of classification. The Bayesian decision criterion of predicting the class with the largest posterior probability is then:

$$\begin{aligned} \arg\max_y p(y|\mathbf{x};\theta) &= \arg\max_y \frac{p(\mathbf{x}|y;\theta)p(y)}{p(x)} && (7.26) \\ &= \arg\max_y p(\mathbf{x}|y;\theta)p(y), && (7.27) \end{aligned}$$

where we have used the fact that the denominator does not depend on $y$ and can hence be ignores. In the case of binary classification, this reads:

$$\arg\max_y p(y|\mathbf{x};\theta) = \arg\max_y (p(\mathbf{x}|y=0;\theta)p(y=0), p(\mathbf{x}|y=1;\theta)p(y=1). \tag{7.28}$$

While using generative models for classification seem to be much more elaborate, we will see later that there are several arguments which make generative models attractive for machine learning, and we will arue that generative models are do more closely resemble human brain processing principles.

## 7.4 Discriminant analysis

We will now discuss some common examples of using generative models in classification. The methods in this section go back to a paper by R. Fisher in 1936. In the following examples we consider that there are $k$ classes, and we first assume that each class has members which are Gaussian distribution over the $n$ feature value. An example for $n = 2$ is shown in Fig.7.2A.

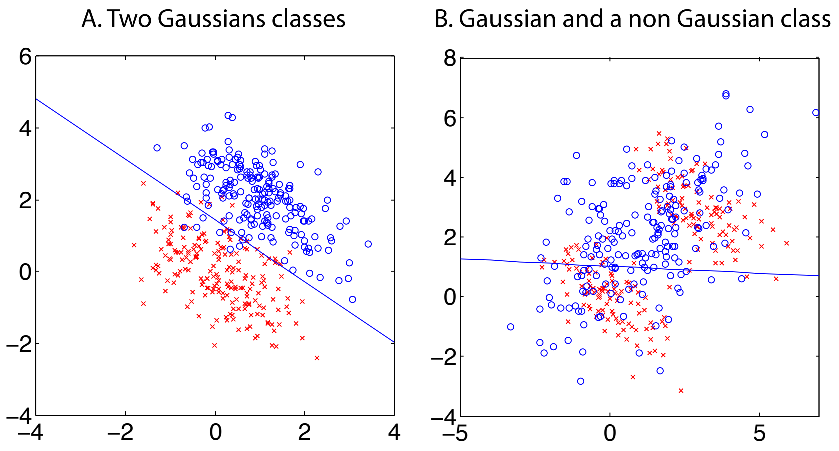

**Fig. 7.2** Linear Discriminant analysis on a two class problem with different class distributions.

Each of the classes have a certain class prior

$$p(y = k) = \phi_k \tag{7.29}$$

, and each class itself is multivariate Gaussian distributed, generally with different means, $\mu_k$ and variances, $\Sigma_k$,

$$p(\mathbf{x}|y = k) = \frac{1}{\sqrt{2\pi}^n \sqrt{|\Sigma_0|}} e^{-\frac{1}{2}(\mathbf{x}-\mu_k)^T \Sigma_k^{-1} (\mathbf{x}-\mu_k)} \tag{7.30}$$

$$\tag{7.31}$$

Since we have supervised data with examples for each class, we can use maximum likelihood estimation to estimate the most likely values for the parameters $\theta = (\phi_k, \mu_k, \Sigma_k)$. For the class priors, this is simply the relative frequency of the training data,

$$\phi_k = \frac{1}{m} \sum_{i \in K} 1 \tag{7.32}$$

where $K = \{i, for y^{(i)} = k\}$ is the set of all indices with examples from class $k$. The estimates of the means and variances within each class are given by

$$\mu_k = \frac{1}{|K|} \sum_{i \in K} \mathbf{x}^{(i)} \tag{7.33}$$

$$\Sigma_k = \frac{1}{|K|}\sum_{i\in K}(x^{(i)} - \mu_{y^{(i)}})(x^{(i)} - \mu_{y^{(i)}})^T, \tag{7.34}$$

where $|K|$ are the number of examples with class label $k$.

With these estimates, we can calculate the optimal (in a Bayesian sense) decision rule, $G(x;\theta)$, as a function of $\mathbf{x}$ with parameters $\theta$, namely

$$\begin{aligned}
G(x) &= arg\max_k p(y = k|\mathbf{x}) && (7.35)\\
&= arg\max_k [p(\mathbf{x}|y = k;\theta)p(y = k)] && (7.36)\\
&= arg\max_k [log(p(\mathbf{x}|y = k;\theta)p(y = k))] && (7.37)\\
&= arg\max_k [-log(\sqrt{2\pi}^n\sqrt{|\Sigma_0|}) - \frac{1}{2}(\mathbf{x}-\mu_k)^T\Sigma_k^{-1}(\mathbf{x}-\mu_k) + log(\phi_k)] && (7.38)\\
&= arg\max_k [-\frac{1}{2}\mathbf{x}^T\Sigma_k^{-1}\mathbf{x} - \frac{1}{2}\mu_k^T\Sigma_k^{-1}\mu_k + \mathbf{x}^T\Sigma_k^{-1}\mu_k + log(\phi_k)], && (7.39)
\end{aligned}$$

since the first term in equation 7.38 does not depend on $k$ and we can multiply out the other terms. With the maximum likelihood estimates of the parameters, we have all we need to make this decision.

In order to calculate the decision boundary between classes $l$ and $k$, we make the common additional assumption that the covariance matrices of the classes are the same,

$$\Sigma_k =: \Sigma. \tag{7.40}$$

The decision point between the two classes with equal class priors is then given by the point where the probabilities for the two classes (eq.7.39) is the same. This gives

$$log(\frac{\phi_k}{\phi_l}) - \frac{1}{2}(\mu_k - \mu_l)^T\Sigma^{-1}(\mu_k + \mu_l) + \mathbf{x}\Sigma^{-1}(\mu_k - \mu_l) = 0. \tag{7.41}$$

The first two terms do not depend on $x$ and can be summarized as constant $\mathbf{a}$. We can also introduce the vector

$$\mathbf{w} = \Sigma^{-1}(\mu_k - \mu_l). \tag{7.42}$$

With these simplifying notations is it easy to see that this decision boundary is a linear,

$$\mathbf{a} + \mathbf{wx} = 0, \tag{7.43}$$

and this method with the Gaussian class distributions with equal variances is called **Linear Discriminant Analysis (LDA)**. The vector $\mathbf{w}$ is perpendicular to the decision surface. Examples are shown in Figure 7.2. If we do not make the assumption of equal variances of the classes, then we have a quadratic equation for the decision boundary, and the method is then called **Quadratic Discriminant Analysis (GDA)**. With the assumptions of LDA, we can calculate the contrastive model directly using Bayes rule.

$$p(y = k|\mathbf{x};\theta) = \frac{\phi_k\frac{1}{\sqrt{2\pi}^n\sqrt{|\Sigma|}}e^{-\frac{1}{2}(\mathbf{x}-\mu_k)^T\Sigma_k^{-1}(\mathbf{x}-\mu_k)}}{\phi_k\frac{1}{\sqrt{2\pi}^n\sqrt{|\Sigma|}}e^{-\frac{1}{2}(\mathbf{x}-\mu_k)^T\Sigma_k^{-1}(\mathbf{x}-\mu_k)} + \phi_l\frac{1}{\sqrt{2\pi}^n\sqrt{|\Sigma|}}e^{-\frac{1}{2}(\mathbf{x}-\mu_l)^T\Sigma_l^{-1}(\mathbf{x}-\mu_l)}} \tag{7.44}$$

$$= \frac{1}{1 + \frac{\phi_l}{\phi_k} exp^{-\theta^T x}}, \tag{7.45}$$

where $\theta$ is an appropriate function of the parameters $\mu_k$, $\mu_l$, and $\Sigma$. Thus, the contrastive model is equivalent to logistic regression discussed in the previous chapter, although we use different parametrisations and the two methods will therefore usual give different results on specific data sets. So which method should be used? In LDA we made the assumption that each class is Gaussian distributed. If this is the case, then LDA is the best method we can use. Discriminant analysis is also popular since it often works well even when the classes are not strictly Gaussian. However, as can be seen in Figure 7.2B, it can produce quite bad results if the data are multimodal distributed. Logistic regression is somewhat more general since it does not make the assumption that the class distributions are Gaussian. However, as ;long as we consider only linear models, logistic regression would have also problems with the data shown in Figure 7.2B.

Finally, we should note that Fisher's original method was slightly more general than the examples discussed here since he did not assume Gaussian distributions. Instead considered within-class variances compared to between-class variances, something which resembles a signal-to-noise ratio. In **Fisher discriminant analysis (FDA)**, the separating hyperplane is defined as

$$\mathbf{w} = (\Sigma_k + \Sigma_l)^{-1}(\mu_k - \mu_l). \tag{7.46}$$

which is the same as in LDA in the case of equal covariance matrices.

## 7.5 Naive Bayes

In the previous example we used two dimensional feature vectors to illustrate the classification problems with two dimensional plots. However, most machine learning applications work with high dimensional feature vectors, and we will here discuss an important example of a Bayesian model that is often used with high-dimensional data.

To discuss this we follow an example, that of making a spam filter that classifies email messages as either spam ($y = 1$) or non-spam ($y = 0$) emails. To do this we need first a method to represent the problem in a suitable way. We chose here to represent a text (email in this situation) as **vocabulary** vector. A vocabulary is simply the list of all possible words that we consider, and the text is represented by this vector with entries 1 if the word can be found in the list or an entry 0 if not, e.g.

$$\mathbf{x} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 1 \\ \cdot \\ \cdot \\ 0 \end{pmatrix} \begin{array}{c} \text{a} \\ \text{aardvark} \\ \text{aardwolf} \\ \cdot \\ \cdot \\ \cdot \\ \text{buy} \\ \cdot \\ \cdot \\ \text{zygmurgy} \end{array} \tag{7.47}$$

We are here only considering values 0 and 1 instead of, for example, counting how often the corresponding word appears. The later is usually called a 'bag of words'. The difference is that each entry is a binomial random variable and would be a multinomial in the other example, though the methods generalize directly to the other case. Note that this feature vector is typically very high dimensional. Let us consider here that our vocabulary has 50.000 word, which is a typical size of common languages.

We now want to build a discriminative model from some training examples. That is, we want to model

$$p(\mathbf{x}|y) = p(x_1, x_2, ..., x_{50000}|y). \tag{7.48}$$

This is a very high dimensional density function which has $2^{50.000} - 1$ parameters (the -1 comes from the normalization condition). We can factorize this conditional density function with the chain rule

$$p(x_1, x_2, ..., x_{50000}|y) = p(x_1|y)p(x_2|y, x_1)...p(x_{50000}|y, x_1, ...., x_{49999}). \tag{7.49}$$

While the right hand side has only 50.000 factors, there are still $2^{50.000} - 1$ parameters we have to learn. We now make a strong assumption namely that all the words are conditionally independent in each text, that is,

$$p(x_1|y)p(x_2|y, x_1)...p(x_{50000}|y, x_1, ...., x_{49999}) = p(x_1|y)p(x_2|y)...p(x_{50000}|y). \tag{7.50}$$

This is called the **Naive Bayes (NB) assumption**. Hence, we can write the conditional probability as a factor of terms with 50.000 parameters

$$p(\mathbf{x}|y) = \prod_{j=1}^{50000} p(x_j|y). \tag{7.51}$$

To estimate these parameters we can apply maximum likelihood estimation, which gives

$$\phi_{j,y=1} = \frac{1}{|\{y=1\}|} \sum_{i \in \{y=1\}} x_j^{(i)} \tag{7.52}$$

$$\phi_{j,y=0} = \frac{1}{|\{y=0\}|} \sum_{i \in \{y=0\}} x_j^{(i)} \tag{7.53}$$

$$\phi_y = \frac{|\{y=1\}|}{m}. \tag{7.54}$$

The first equation is the probability that the word $j$ appears in a spam text, the second equation is that the word $j$ appears in a non-spam text, and the third equation specifies the frequency of spam examples in the data set.

With these parameters we can now calculate the probability that email $\mathbf{x}$ is spam as

$$p(y=1|\mathbf{x}) = \frac{\prod_{j=1}^{50.000} \phi_{j,y=1}\phi_y}{\prod_{j=1}^{50.000} \phi_{j,y=1}\phi_y + \prod_{j=1}^{50.000} \phi_{j,y=0}(1-\phi_y)}. \tag{7.55}$$

In practice this often works to some extent, at least when the Naive Bayes assumption is appropriate. Of course, words in a text should be highly correlated, but the gist here is that the pure frequency of words has some correlates with the type of text.

Finally, a note that there is a slight problem if some of the words, say $x_{100}$, are not part of the training set. In this case we get an estimate that the probability of this word every occurring is zero, $\phi_{100,y=1} = 0$ and $\phi_{100,y=0} = 0$, and hence $p(y = 1|x) = \frac{0}{0}$. A common trick, called **Laplace smoothing** is to add one occurrence of this word in every case, which will insert a small probability proportional to the number of training examples to the estimates,

$$\phi_{j,y=1} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = 1\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\} + 2} \tag{7.56}$$

$$\phi_{j,y=0} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = 0\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\} + 2}. \tag{7.57}$$

It is interesting to compare the Naive Bayes classification with other classification methods.