Machine Learning Lecture Notes

Predrag Radivojac

April 3, 2015

1 Introduction to prediction problems

We start by defining a data set $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_n, y_n)\}$, where $\mathbf{x}_i \in \mathcal{X}$ is the i-th object and $y_i \in \mathcal{Y}$ is the corresponding target designation. We usually assume that $\mathcal{X} = \mathbb{R}^k$, in which case $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ik})$ is a k-dimensional vector called data point (or example). Each dimension of $\mathbf{x}_i$ is typically called a feature or an attribute.

We generally distinguish between two related but different types of prediction problems: classification and regression. Generally speaking, we have a classification problem if $\mathcal{Y}$ is discrete and a regression problem when $\mathcal{Y}$ is continuous. The *classification problem* refers to constructing a function that for a previously unseen data point $\mathbf{x}$ infers (predicts) its class label y. A particular function or algorithm that infers class labels, and is typically optimized to minimize or maximize some objective (or fitness) function, is referred to as a *classifier* or a *classification model*. The cardinality of $\mathcal{Y}$ in classification problems is usually small, e.g. $\mathcal{Y} = \{\text{healthy, disease}\}$. On the other hand, the *regression problem* refers to constructing a model that for a previously unseen data point $\mathbf{x}$ approximates the target value y as closely as possible, where often times $\mathcal{Y} = \mathbb{R}$. Regression problems are sometimes seen as fitting.

An example of a data set for classification with $n = 3$ data points and $k = 5$ features is shown in Table 1. Problems in which $|\mathcal{Y}| = 2$ are referred to as binary classification problems, whereas problems in which $|\mathcal{Y}| > 2$ are referred to as multi-class classification problems. This can be more complex as, for instance, in classification of text documents into categories such as $\{\text{sports, medicine, travel}, \ldots\}$. Here, a single document may be related to more than one value in the set; e.g. an article on sports medicine. To account for this, we can certainly say that $\mathcal{Y} = \mathcal{P}(\{\text{sports, medicine, travel}, \ldots\})$ and treat the problem as multi-class classification, albeit with a very large output space. However, it is often easier to think that more than one value of the output space can be associated with any particular input. We refer to this learning task as multi-label classification and set $\mathcal{Y} = \{\text{sports, medicine, travel}, \ldots\}$. Finally, $\mathcal{Y}$ can be a set of structured outputs, e.g. strings, trees, or graphs. This classification scenario is usually referred to as structured-output learning. The cardinality of the output space in structured-output learning problems is often very high. An example of a regression problem is shown in Table 2.

	wt [kg]	ht [m]	T [°C]	sbp [mmHg]	dbp [mmHg]	y
$\mathbf{x}_1$	91	1.85	36.6	121	75	-1
$\mathbf{x}_2$	75	1.80	37.4	128	85	$+1$
$\mathbf{x}_3$	54	1.56	36.6	110	62	-1

Table 1: An example of a binary classification problem: prediction of a disease state for a patient. Here, features indicate weight (wt), height (ht), temperature (T), systolic blood pressure (sbp), and diastolic blood pressure (dbp). The class labels indicate presence of a particular disease, e.g. diabetes. This data set contains one positive data point ($\mathbf{x}_2$) and two negative data points ($\mathbf{x}_1$, $\mathbf{x}_3$). The class label shows a disease state, i.e. $y_i = +1$ indicates the presence while $y_i = -1$ indicates absence of disease.

	size [sqft]	age [yr]	dist [mi]	inc [$]	dens [ppl/mi^2]	y
$\mathbf{x}_1$	1250	5	2.85	56,650	12.5	2.35
$\mathbf{x}_2$	3200	9	8.21	245,800	3.1	3.95
$\mathbf{x}_3$	825	12	0.34	61,050	112.5	5.10

Table 2: An example of a regression problem: prediction of the price of a house in a particular region. Here, features indicate the size of the house (size) in square feet, the age of the house (age) in years, the distance from the city center (dist) in miles, the average income in a one square mile radius (inc), and the population density in the same area (dens). The target indicates the price a house is sold at, e.g. in hundreds of thousands of dollars.

In both prediction scenarios, we assume that the features are easy to collect for each object (e.g. by measuring the height of a person or the square footage of a house), while the target variable is difficult to observe or expensive to collect. Such situations usually benefit from the construction of a computational model that predicts targets from a set of input values. The model is trained using a set of input objects for which target values have already been collected.

Observe that there does not exist a strict distinction between classification and regression. For example, if the output space is $\mathcal{Y} = \{0, 1, 2\}$, we need not treat this problem as classification. This is because there exists a relationship of order among elements of $\mathcal{Y}$ that can simplify model development. For example, we can take $\mathcal{Y} = [0, 2]$ and simply develop a regression model from which we can recover the original discrete values by rounding the raw prediction outputs. The selection of a particular way of modeling depends on the analyst and their knowledge of the domain as well as technical aspects of learning.

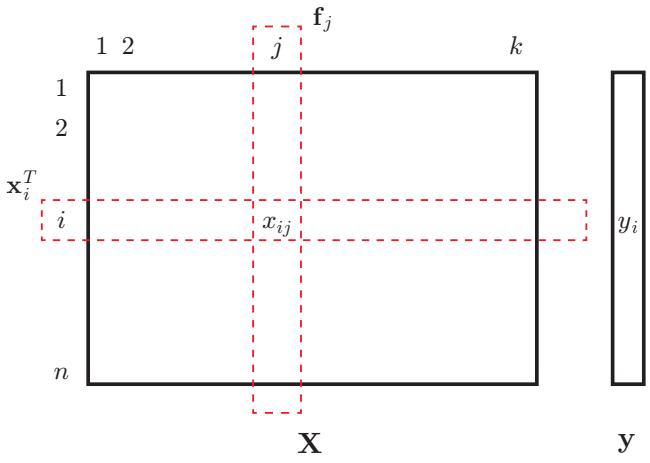

Figure 1: Data set representation and notation. $\mathbf{X}$ is an n-by-k matrix representing features and data points, whereas y is an n-by-1 vector of targets.

2 Useful notation

In the machine learning literature we use k-tuples $\mathbf{x} = (x_1, x_2, \ldots, x_k)$ to denote data points. However, often times we can benefit from the algebraic notation, where we consider each data point $\mathbf{x}$ to be a column vector in the k-dimensional Euclidean space. In the algebraic notation $\mathbf{x} = [x_1\ x_2\ \ldots\ x_k]^T$, where T is the transpose operator. Here, a linear combination of features and some set of coefficients $\mathbf{w} = (w_1, w_2, \ldots, w_k) \in \mathbb{R}^k$

$$\sum_{i=1}^{k} w_i x_i = w_1 x_1 + w_2 x_2 + ... + w_k x_k$$

can be expressed using an inner (dot) product of column vectors $\mathbf{w}^T\mathbf{x}$. A linear combination $\mathbf{w}^T\mathbf{x}$ results in a single number. Another useful notation for such linear combinations will be $\langle \mathbf{w}, \mathbf{x} \rangle$.

We will also use an n-by-k matrix $\mathbf{X} = (\mathbf{x}_1^T, \mathbf{x}_2^T, \ldots, \mathbf{x}_n^T)$ to represent the entire set of data points and $\mathbf{y}$ to represent a column vector of targets. For example, the i-th row of $\mathbf{X}$ represents data point $\mathbf{x}_i^T$. Finally, the j-th column of $\mathbf{X}$, denoted as $\mathbf{f}_j$, is an n-by-1 vector which contains the values of feature j for all data points. The notation is further presented in Figure 1.

3 Optimal classification and regression models

Our goal now is to establish the performance criteria that will be used to evaluate predictors $f : \mathcal{X} \to \mathcal{Y}$ and subsequently define optimal classification and regression models. We start with classification and consider a situation where the joint probability distribution $p(\mathbf{x}, y)$ is either known or can be learned from data. Also, we will consider that we are given a cost function $c : \mathcal{Y} \times \mathcal{Y} \to [0, \infty)$,

where for each prediction $\hat{y}$ and true target value y the classification cost can be expressed as a constant $c(\hat{y}, y)$, regardless of the input $\mathbf{x} \in \mathcal{X}$ given to the classifier. This cost function can simply be stored as a $|\mathcal{Y}| \times |\mathcal{Y}|$ cost matrix.

The criterion for optimality of a classifier will be probabilistic. In particular, we are interested in minimizing the expected cost

$$\begin{aligned} E[C] &= \int_{\mathcal{X}} \sum_{y} c(\hat{y}, y) p(\mathbf{x}, y) d\mathbf{x} \\ &= \int_{\mathcal{X}} p(\mathbf{x}) \sum_{y} c(\hat{y}, y) p(y|\mathbf{x}) d\mathbf{x}, \end{aligned}$$

where the integration is carried out over the entire input space $\mathcal{X} = \mathbb{R}^k$. From this equation, we can see that the optimal classifier can be expressed as

$$f_{\mathrm{BR}}(\mathbf{x}) = \arg\min_{\hat{y} \in \mathcal{Y}} \left\{ \sum_{y} c(\hat{y}, y) p(y|\mathbf{x}) \right\}.$$

for any $\mathbf{x} \in \mathcal{X}$. We will refer to this classifier as the *Bayes risk classifier*. One example where the Bayes risk classifier can be useful is the medical domain. Suppose our goal is to decide whether a patient with a particular set of symptoms ($\mathbf{x}$) should be sent for an additional lab test ($y = 1$ if yes and $y = -1$ if not), with cost c_{lab}, in order to improve diagnosis. However, if we do not perform a lab test and the patient is later found to have needed the test for proper treatment, we may incur a significant penalty, say c_{lawsuit}. If $c_{\mathrm{lawsuit}} \gg c_{\mathrm{lab}}$, as it is expected to be, then the classifier needs to appropriately adjust its outputs to account for the cost disparity in different forms of incorrect prediction.

In many practical situations, however, it may not be possible to define a meaningful cost matrix and, thus, a reasonable criterion would be to minimize the probability of a classifier's error $P(f(\mathbf{x}) \neq y)$. This corresponds to the situation where the cost function is defined as

$$c(\hat{y}, y) = \begin{cases} 0 & \text{when } y = \hat{y} \\ \\ 1 & \text{when } y \neq \hat{y} \end{cases}$$

After plugging these values in the definition for $f_{\mathrm{BR}}(\mathbf{x})$, the Bayes risk classifier simply becomes the maximum a posteriori (MAP) classifier. That is,

$$f_{\mathrm{MAP}}(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} \{p(y|\mathbf{x})\}.$$

Therefore, if $p(y|\mathbf{x})$ is known or can be accurately learned, we are fully equipped to make the prediction that minimizes the total cost. In other words, we have converted the problem of minimizing the expected classification cost or probability of error, into the problem of learning functions, more specifically learning probability distributions.

The analysis for regression is a natural extension of that for classification. Here too, we are interested in minimizing the expected cost of prediction of the true target y when a predictor $f(\mathbf{x})$ is used. The expected cost can be expressed as

$$E[C] = \int_{\mathcal{X}} \int_{\mathcal{Y}} c(f(\mathbf{x}), y) p(\mathbf{x}, y) d\mathbf{x} dy,$$

where $c : \mathbb{R} \times \mathbb{R} \to [0, \infty)$ is again some cost function between the predicted value $f(\mathbf{x})$ and the true value y. For simplicity, we will consider

$$c(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2,$$

which results in

$$\begin{aligned} E[C] &= \int_{\mathcal{X}} \int_{\mathcal{Y}} (f(\mathbf{x}) - y)^2 p(\mathbf{x}, y) d\mathbf{x} dy \\ &= \int_{\mathcal{X}} p(\mathbf{x}) \underbrace{\int_{\mathcal{Y}} (f(\mathbf{x}) - y)^2 p(y|\mathbf{x}) dy}_{g(f(\mathbf{x}))} d\mathbf{x}. \end{aligned}$$

Assuming $f(\mathbf{x})$ is flexible enough to be separately optimized for each unit volume $d\mathbf{x}$, we see that minimizing $E[C]$ leads us to the problem of minimizing

$$g(u) = \int_{\mathcal{Y}} (u - y)^2 p(y|\mathbf{x}) dy,$$

where we used a substitution $u = f(\mathbf{x})$. We can now differentiate g with respect to u as

$$\frac{\partial g(u)}{\partial u} = 2 \int_{\mathcal{Y}} (u - y) p(y|\mathbf{x}) dy$$

which results in the optimal solution

$$\begin{aligned} f^*(\mathbf{x}) &= \int_{\mathcal{Y}} y p(y|\mathbf{x}) dy \\ &= E[y|\mathbf{x}]. \end{aligned}$$

Therefore, the optimal regression model in the sense of minimizing the square error between the prediction and the true target is the conditional expectation $E[y|\mathbf{x}]$. It may appear that in the above equations, setting $f(\mathbf{x}) = y$ would always lead to $E[C] = 0$. Unfortunately, this would be an invalid operation because for a single input $\mathbf{x}$ there may be multiple possible outputs y and they can certainly appear in the same data set. This, on the other hand, is not allowed for $f(\mathbf{x})$ that must always have the same output for the same input. $E[C] = 0$ can only be achieved if $p(y|\mathbf{x})$ is a delta function for every $\mathbf{x}$.

Having found the optimal regression model, we can now write the expected cost in the cases of both optimal and suboptimal models $f(\mathbf{x})$. That is, we are interested in expressing $E[C]$ when

1. $f(\mathbf{x}) = E[y|\mathbf{x}]$
2. $f(\mathbf{x}) \neq E[y|\mathbf{x}]$.

We have already found $E[C]$ when $f(\mathbf{x}) = E[y|\mathbf{x}]$. The expected cost can be simply expressed as

$$E[C] = \int_{\mathcal{X}} \int_{\mathcal{Y}} (E[y|\mathbf{x}] - y)^2 p(\mathbf{x}, y) d\mathbf{x} dy,$$

which corresponds to the (weighted) squared error between the optimal prediction and the target everywhere in the feature space. This is the best scenario in regression; we cannot achieve a lower cost on average.

The next situation is when $f(\mathbf{x}) \neq E[y|\mathbf{x}]$. Here, we will proceed by decomposing the squared error as

$$\begin{aligned}(f(\mathbf{x}) - y)^2 &= (f(\mathbf{x}) - E[y|\mathbf{x}] + E[y|\mathbf{x}] - y)^2 \\ &= (f(\mathbf{x}) - E[y|\mathbf{x}])^2 + 2\underbrace{(f(\mathbf{x}) - E[y|\mathbf{x}])(E[y|\mathbf{x}] - y)}_{g(\mathbf{x},y)} + (E[y|\mathbf{x}] - y)^2\end{aligned}$$

We will now take a more detailed look at $g(\mathbf{x}, y)$ when placed under the expectation over $p(\mathbf{x}, y)$

$$\begin{aligned}E_{\mathbf{x},y}[g(\mathbf{x}, y)] &= \int_{\mathcal{X}} \int_{\mathcal{Y}} (f(\mathbf{x}) - E[y|\mathbf{x}])(E[y|\mathbf{x}] - y) p(\mathbf{x}, y) d\mathbf{x} dy \\ &= \int_{\mathcal{X}} \int_{\mathcal{Y}} f(\mathbf{x}) E[y|\mathbf{x}] p(\mathbf{x}, y) d\mathbf{x} dy - \int_{\mathcal{X}} \int_{\mathcal{Y}} f(\mathbf{x}) y p(\mathbf{x}, y) d\mathbf{x} dy + \\ &+ \int_{\mathcal{X}} \int_{\mathcal{Y}} E[y|\mathbf{x}] y p(\mathbf{x}, y) d\mathbf{x} dy - \int_{\mathcal{X}} \int_{\mathcal{Y}} E[y|\mathbf{x}] E[y|\mathbf{x}] p(\mathbf{x}, y) d\mathbf{x} dy \\ &= \int_{\mathcal{X}} f(\mathbf{x}) E[y|\mathbf{x}] p(\mathbf{x}) d\mathbf{x} - \int_{\mathcal{X}} f(\mathbf{x}) p(\mathbf{x}) \int_{\mathcal{Y}} y p(y|\mathbf{x}) dy d\mathbf{x} + \\ &+ \int_{\mathcal{X}} E[y|\mathbf{x}] E[y|\mathbf{x}] p(\mathbf{x}) d\mathbf{x} - \int_{\mathcal{X}} E[y|\mathbf{x}] E[y|\mathbf{x}] p(\mathbf{x}) d\mathbf{x} \\ &= 0.\end{aligned}$$

Therefore, we can now express the expected cost as

$$\begin{aligned}E[C] &= \int_{\mathcal{X}} \int_{\mathcal{Y}} (f(\mathbf{x}) - y)^2 p(\mathbf{x}, y) d\mathbf{x} dy \\ &= \int_{\mathcal{X}} (f(\mathbf{x}) - E[y|\mathbf{x}])^2 p(\mathbf{x}) d\mathbf{x} + \int_{\mathcal{X}} \int_{\mathcal{Y}} (E[y|\mathbf{x}] - y)^2 p(\mathbf{x}, y) d\mathbf{x} dy,\end{aligned}$$

where the first term is the "distance" between the trained model $f(\mathbf{x})$ and the optimal model $E[y|\mathbf{x}]$ and the second term is the "distance" between the optimal model $E[y|\mathbf{x}]$ and the correct target value y.

To sum up, we argued here that optimal classification and regression models critically depend on knowing or accurately learning the posterior distribution $p(y|\mathbf{x})$. This task can be solved in different ways, but a straightforward approach is to assume a functional form for $p(y|\mathbf{x})$, say $p(y|\mathbf{x}, \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is a set of weights or parameters that are to be learned from the data. Alternatively, we can learn the class-conditional and prior distributions, $p(\mathbf{x}|y)$ and $p(y)$, respectively. Using

$$\begin{aligned} p(y|\mathbf{x}) &= \frac{p(\mathbf{x}, y)}{p(\mathbf{x})} \\ &= \frac{p(\mathbf{x}|y)p(y)}{\sum_y p(\mathbf{x}, y)} \\ &= \frac{p(\mathbf{x}|y)p(y)}{\sum_y p(\mathbf{x}|y)p(y)} \end{aligned}$$

we can see that these two learning approaches are equivalent in theory. The choice depends on our prior knowledge and/or preferences.

Models obtained by directly estimating $p(y|\mathbf{x})$ are called *discriminative models* and models obtained by directly estimating $p(\mathbf{x}|y)$ and $p(y)$ are called *generative models*. Direct estimation of the joint distribution $p(\mathbf{x}, y)$ is relatively rare as it may be more difficult to hypothesize its parametric or non-parametric form from which the parameters are to be found. Finally, in some situations we can train a model without an explicit probabilistic approach in mind. In these cases, we typically aim to show a good performance of an algorithm in practice, say on a large number of data sets, according to a performance measure relevant to the problems at hand.

4 Bayes Optimal Models

We saw earlier that optimal prediction models reduce to the learning of the posterior distribution $p(y|\mathbf{x})$ which is then used to minimize the expected cost (risk, loss). However, in practice, the probability distribution $p(y|\mathbf{x})$ must be modeled using a particular functional form and a set of tunable coefficients. When $\mathcal{X} = \mathbb{R}^k$ and $\mathcal{Y} = \{0, 1\}$, one such example is used in logistic regression, where

$$p(1|\mathbf{x}) = \frac{1}{1 + e^{-(w_0 + \sum_{j=1}^{k} w_j x_j)}}$$

and $p(0|\mathbf{x}) = 1 - p(1|\mathbf{x})$. Here $(w_0, w_1, \ldots, w_k) \in \mathbb{R}^{k+1}$ is a set of weights that are to be inferred from a given data set $\mathcal{D}$ and $\mathbf{x} = (x_1, \ldots, x_k) \in \mathbb{R}^k$ is an input data point. A number of other types of functional relationships can be used

as well, providing a vast set of possibilities for modeling distributions. We will adjust our notation to more precisely denote the posterior distribution as

$$p(y|\mathbf{x}) = p(y|\mathbf{x}, f),$$

where f is a particular function from some function (hypothesis) space $\mathcal{F}$. We can think of $\mathcal{F}$ as a set of all functions from a specified class, say for all $(w_0, w_1, \ldots, w_k) \in \mathbb{R}^{k+1}$ in the example above, but we can also extend the functional class beyond simple parameter variation to incorporate non-linear decision surfaces.

In a typical learning problem, we are given a data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ and are asked to model $p(y|\mathbf{x})$. For this purpose, we will think of $\mathcal{D}$ as a realization of a random variable D and will assume that $\mathcal{D}$ was drawn according to the true underlying distribution $p(\mathbf{x}, y)$. Thus, our task is to express $p(y|\mathbf{x}, \mathcal{D})$. Using the sum and product rules, will rewrite our original task of estimating $p(y|\mathbf{x})$ as

$$\begin{aligned} p(y|\mathbf{x}, \mathcal{D}) &= \int_{\mathcal{F}} p(y|\mathbf{x}, f, \mathcal{D}) p(f|\mathbf{x}, \mathcal{D}) df \\ &= \int_{\mathcal{F}} p(y|\mathbf{x}, f) p(f|\mathbf{x}, \mathcal{D}) df. \end{aligned}$$

Here we used conditional independence between output Y and the data set D once a particular model f was selected based on $\mathcal{D}$; thus $p(y|\mathbf{x}, f, \mathcal{D}) = p(y|\mathbf{x}, f)$. This equation, gives us a sense that the optimal decision can be made through a mixture of distributions $p(y|\mathbf{x}, f)$, where the weights are given as posterior densities $p(f|\mathbf{x}, \mathcal{D})$. In finite hypothesis spaces $\mathcal{F}$ we have that

$$p(y|\mathbf{x}, \mathcal{D}) = \sum_{f \in \mathcal{F}} p(y|\mathbf{x}, f) p(f|\mathbf{x}, \mathcal{D}),$$

and $p(f|\mathbf{x}, \mathcal{D})$ are posterior probabilities. We may further assume that $p(f|\mathbf{x}, \mathcal{D}) = p(f|\mathcal{D})$, in which case the weights can be precomputed based on the given data set $\mathcal{D}$. This leads to more efficient calculations of the posterior probabilities.

In classification, we can rewrite our original MAP classifier as

$$f_{\text{MAP}}(\mathbf{x}, \mathcal{D}) = \arg\max_{y \in \mathcal{Y}} \left\{ p(y|\mathbf{x}, \mathcal{D}) \right\},$$

which readily leads to the following formulation

$$\begin{aligned} f_{\text{MAP}}(\mathbf{x}, \mathcal{D}) &= \arg\max_{y \in \mathcal{Y}} \left\{ \int_{\mathcal{F}} p(y|\mathbf{x}, f, \mathcal{D}) p(f|\mathcal{D}) df \right\} \\ &= \arg\max_{y \in \mathcal{Y}} \left\{ \int_{\mathcal{F}} p(y|\mathbf{x}, f) p(\mathcal{D}|f) p(f) df \right\}. \end{aligned}$$

It can be shown that no classifier can outperform the *Bayes optimal classifier*. Interestingly, the Bayes optimal model also hints that a better prediction performance can be achieved by combining multiple models and averaging their

outputs. This provides theoretical support for ensemble learning and methods such as bagging and boosting.

One problem in Bayes optimal classification is efficient calculation of $f_{\mathrm{MAP}}(\mathbf{x}, \mathcal{D})$, given that the function (hypothesis) space $\mathcal{F}$ is generally uncountable. One approach to this is sampling of functions from $\mathcal{F}$ according to $p(f)$ and then calculating $p(f|\mathcal{D})$ or $p(\mathcal{D}|f)$. This can be computed until $p(y|\mathbf{x}, \mathcal{D})$ converges.