36-462: Data Mining **Spring 2019**

Lecture 2: January 17

Lecturer: Siva Balakrishnan

Our broad goal today will be to think more deeply about the problem of classification. This material is in Sections 4.1 and 4.2 of the ISL book.

2.1 Supervised Learning Setup

Recall the canonical supervised learning setup. We observe *training examples*:

$$\{(x_1, y_1), \ldots, (x_n, y_n)\} \sim \mathcal{P}_{xy}.$$

Each feature vector $x_i \in \mathbb{R}^p$, i.e. the feature vectors are p-dimensional. There are a few symbols in the above expression that need some clarification:

1. The $\sim$ is used to mean that each of the training samples is an *independent* draw (or sample) from the distribution $\mathcal{P}_{xy}$.

2. The distribution $\mathcal{P}_{xy}$ is the joint distribution on (x, y) pairs.

The broad goal of learning is to use the training data to construct procedures that can predict the y for new, previously unseen xs.

2.2 A Thought Experiment

Before we start investigating different ways to construct learning procedures we should begin with a thought experiment. Suppose that we were handed the distribution $\mathcal{P}_{xy}$, instead of just the training samples (this is roughly equivalent to having infinitely many training samples from the distribution $\mathcal{P}_{xy}$).

Here are two questions to think about – suppose we knew the distribution $\mathcal{P}_{xy}$:

1. What is the best predictor of y from x that we can construct?

2. How good is this predictor?

The answer – as we discussed last time depends on the *loss function* – which roughly captures the sense in which we want our predictions to be "good". The loss function is an important part of the learning process and requires careful thought. For today however, we will consider two popular standard choices – the squared loss for regression and the 0/1 loss for classification.

2.2.1 Regression

You have likely seen this before (those of you taking Cosma/Ann's class have seen this in your first lecture) but hopefully it will help see the parallels in classification. If we care about the squared loss then we essentially want to find a regression function (a function of the inputs X) that has low *mean squared error*/risk:

$$R(f) = \text{MSE}(f) = \mathbb{E}[(Y - f(X))^2].$$

To find the best function f we use the law of total expectation which tells us that for any random variables U and V we have that:

$$\mathbb{E}[U] = \mathbb{E}[\mathbb{E}[U|V]],$$

i.e. to calculate the average of U we can calculate the average of U for each fixed value V and average those averages. Using the law of total expectation we can calculate the loss for each X and average those:

$$\begin{aligned} R(f) &= \mathbb{E}[\mathbb{E}[(Y - f(X))^2|X]] = \mathbb{E}[\mathbb{E}[(Y^2 + f(X)^2 - 2f(X)Y)|X]] \\ &= \mathbb{E}[\mathbb{E}[Y^2|X]] + \mathbb{E}[f(X)^2 - 2f(X)\mathbb{E}[Y|X]], \end{aligned}$$

and finding the value $f(X)$ that minimizes the second term (the first one does not depend on X) yields the familiar expression that we should choose:

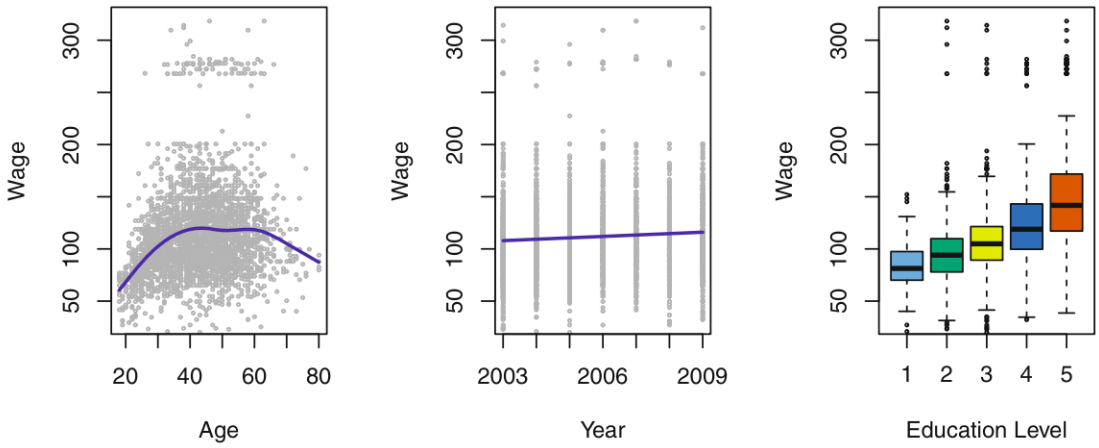
$$f(X) = \mathbb{E}[Y|X].$$

This is the optimal regression function we could construct if we knew the distribution $\mathcal{P}_{xy}$. It is the best predictor of y in the squared loss sense given x.

So how good is this predictor? The answer is given by:

$$R(\mathbb{E}[Y|X]) = \mathbb{E}[\mathbb{E}[(Y - \mathbb{E}[Y|X])^2|X]].$$

Which is the average of the conditional variance of the quantity we are trying to predict Y. This quantity is known as the *unpredictable error* and captures the statistical fluctuation around even the best predictor. Just so that you have a picture in mind – consider the following example from the ISL book:



The task is to predict Wage given some covariates (Age, the year in which we are predicting and the Education Level). Suppose we considered predicting just using Age, then the blue line in the left panel gives us roughly the best predictor – however, you can see that it is not very good – there is a lot of spread around the blue line (the unpredictable error if Age was the only covariate). However, we might be able to improve our predictor by using other/additional covariates. This leads to two important things:

1. In contrast to things like bias and variance, the unpredictable error cannot be lowered by fitting a more flexible model or by collecting more samples.
2. You can try to lower the unpredictable error by collecting more/better features, i.e. changing the inputs can change the unpredictable error. This is one of the reasons why *feature engineering* or collecting/creating the right set of features is an important part of machine learning practice. We should always ask ourselves, are there different inputs that we could collect/create that would help us predict better?

The main question in regression is how do we estimate $\mathbb{E}[Y|X]$? You have either seen/will see in Cosma/Ann's class several different methods to estimate this conditional expectation.

2.2.2 Binary Classification

In binary classification, the outputs Y take on only two values, i.e. there are only two classes. We will denote the two classes by $Y = 0$ (Class 0) and $Y = 1$ (Class 1) respectively. In some cases, (in the distant future) it will be more convenient to label the two classes by $Y \in \{-1, +1\}$ instead.

In binary classification we are often interested in the 0/1 loss and we can once again ask – suppose we had access to the distribution $\mathcal{P}_{xy}$ – what is the best possible classifier and how good is it? We will call a function f a classifier if it maps inputs to 0/1, i.e. $f(X) \in \{0, 1\}$.

Once again, we use the law of total expectation and notice that the risk or the average 0/1 loss of our classifier (this is also called the expected prediction error/expected mis-classification error) is given by:

$$\begin{aligned} R(f) &= \mathbb{E}[\mathbb{E}[\mathbb{I}(Y \neq f(X))|X]] \\ &= \mathbb{E}\left[\sum_{k\in\{0,1\}} \mathbb{I}(f(X) \neq k)\mathbb{P}(Y = k|X)\right], \end{aligned}$$

where we (on the inside) take the average over Y with X held fixed, and then average over the X (the outer expectation). Now, once again we can ask what classifier has smallest risk/loss. For each fixed X we can minimize the expression, and see that the best classifier is given by:

$$f(X) = \begin{cases} 0 & \text{if} \quad \mathbb{P}(Y = 1|X) < \mathbb{P}(Y = 0|X) \\ 1 & \text{otherwise.} \end{cases}$$

We can also rewrite this classifier as:

$$f(X) = \begin{cases} 0 & \text{if} \quad \mathbb{P}(Y = 1|X) < 1/2 \\ 1 & \text{otherwise.} \end{cases}$$

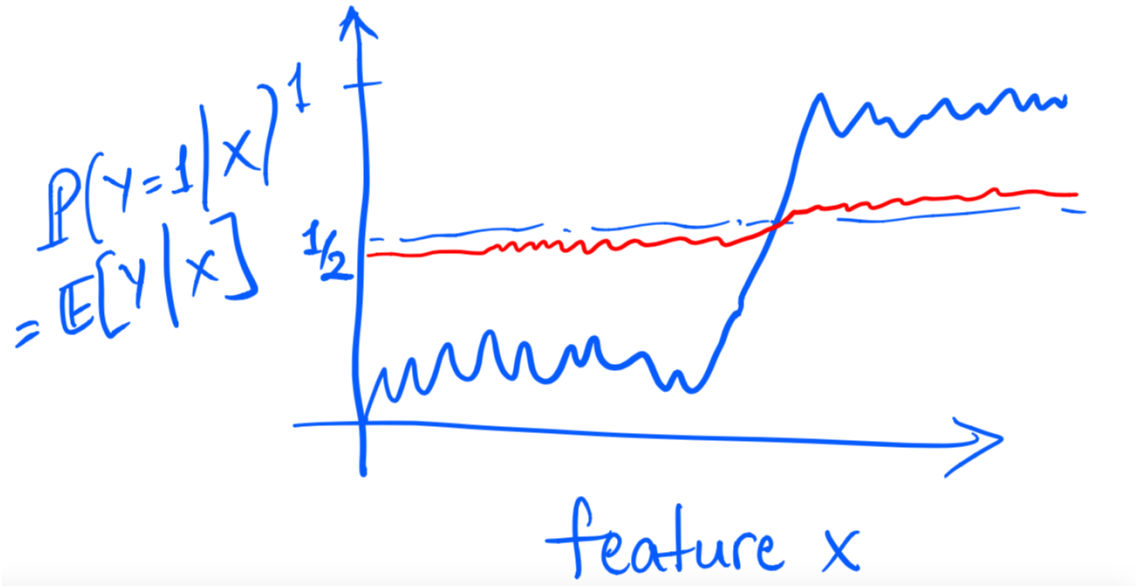
(It does not matter what class we label points for which $\mathbb{P}(Y = 1|X) = \mathbb{P}(Y = 0|X)$ or equivalently points for which $\mathbb{P}(Y = 1|X) = 0.5$.) This leads to the natural rule for classification – for each X we pick the most likely class, i.e. the class $k \in \{0, 1\}$ for which $\mathbb{P}(Y = k|X)$ is larger.

This classifier, which is the best possible classifier in our setup, is called the *Bayes Classifier*. There is an interesting fact to note – in our setting:

$$\mathbb{P}(Y = 1|X) = \mathbb{E}[Y|X],$$

since Y is binary. So binary classification is not that different from regression, we are trying to estimate the same regression function $\mathbb{E}[Y|X]$ – except we don't really care much about what its value is – just if it is above a half or below it.

Just for intuition, here is a picture to keep in mind:



Where for illustration I have shown two one-dimensional classification problems and the corresponding regression function in red and blue. Notice, that in both cases, actually estimating the curve (either the red or blue curve) might be very difficult and might need a lot of data (they are quite wiggly). However, for the blue curve – maybe even a coarse approximation to the regression function will lead to a good classifier – in some sense this is an "easy classification problem" because the probability of being 0/1 is always close to 0/1.

This is why we often, at least from a statistical perspective, think of classification as "easier" than regression. In many cases, even a rough approximation to the regression function will lead to a good classifier.

Now that we have answered what the best binary classifier is – we can also ask how good is it exactly? As we saw before:

$$\begin{aligned} R(f_{\text{Bayes}}) &= \mathbb{E}\left[\sum_{k\in\{0,1\}} \mathbb{I}(f_{\text{Bayes}}(X) \neq k)\mathbb{P}(Y=k|X)\right] \\ &= \mathbb{E}\left[\mathbb{P}(Y=1|X) \times \mathbb{I}(f_{\text{Bayes}}(X) \neq 1) + \mathbb{P}(Y=0|X) \times \mathbb{I}(f_{\text{Bayes}}(X) \neq 0)\right] \\ &= \mathbb{E}[\min\{\mathbb{P}(Y=0|X), \mathbb{P}(Y=1|X)\}]. \end{aligned}$$

This quantity is called the *Bayes error* – i.e. it is analogous to the unpredictable error in the regression problem – it is the unavoidable error of the best classifier. In some sense, the Bayes error tells us how hard our classification problem really is – and the only way to make it smaller is to collect better/more features/predictors.

Notice that the Bayes error can be written in terms of the regression function:

$$\begin{aligned} R(f_{\text{Bayes}}) &= \mathbb{E}[\min\{\mathbb{P}(Y=0|X), \mathbb{P}(Y=1|X)\}] \\ &= \mathbb{E}[\min\{1 - \mathbb{E}[Y|X], \mathbb{E}[Y|X]\}], \end{aligned}$$

and is large when the regression function hovers near $1/2$ (matching our previous figure and intuition about what are difficult regression problems).

Generative and Discriminative Approaches to Classification: Now, that we have seen the form of the Bayes classifier we can think generally about two ways of trying to actually do classification:

1. **Discriminative Classifiers:** We model $\mathbb{P}(Y = 1|X)$ in some way (maybe by a linear-ish model) and then try to estimate this function using the training data. To classify a new point we simply check if $\widehat{\mathbb{P}}(Y = 1|X) > 1/2$ (in which case, we label the point 1).

2. **Generative Classifiers:** We use Bayes' rule to observe that:

   $$\mathbb{P}(Y=1|X) = \frac{\mathbb{P}(X|Y=1) \times \mathbb{P}(Y=1)}{\mathbb{P}(X)}, \mathbb{P}(Y=0|X) = \frac{\mathbb{P}(X|Y=0) \times \mathbb{P}(Y=0)}{\mathbb{P}(X)},$$

   so in order to decide which of these two probabilities is higher we can instead compare:

   $$\mathbb{P}(X|Y=1) \times \mathbb{P}(Y=1) \stackrel{?}{>} \mathbb{P}(X|Y=0) \times \mathbb{P}(Y=0).$$

   So in order to construct our classifier we model the distribution of the features X for the class with $Y = 1$, and the distribution of the features for the class with $Y = 0$, and use our training data to fit these models. To classify we check if:

   $$\widehat{\mathbb{P}}(X|Y=1) \times \widehat{\mathbb{P}}(Y=1) \stackrel{?}{>} \widehat{\mathbb{P}}(X|Y=0) \times \widehat{\mathbb{P}}(Y=0).$$

We will discuss the pros and cons of generative versus discriminative modeling in detail as we learn about more classifiers.

2.2.3 General Classification

In the general case, we have K different classes – so $Y \in \{0, \ldots, K-1\}$. Again we might ask our two favorite questions – if we had access to the distribution $\mathcal{P}_{xy}$ then what is the best classifier and how good is it?

If we use the 0/1 loss we can once again go through the same argument as before and see that the *Bayes Classifier* computes $\mathbb{P}(Y = k|X)$ for $k \in \{0, \ldots, K-1\}$ and picks the most likely class, i.e.:

$$f_{\text{Bayes}}(x) = \arg \max_{k \in \{0,\ldots,K-1\}} \mathbb{P}(Y = k|X = x).$$

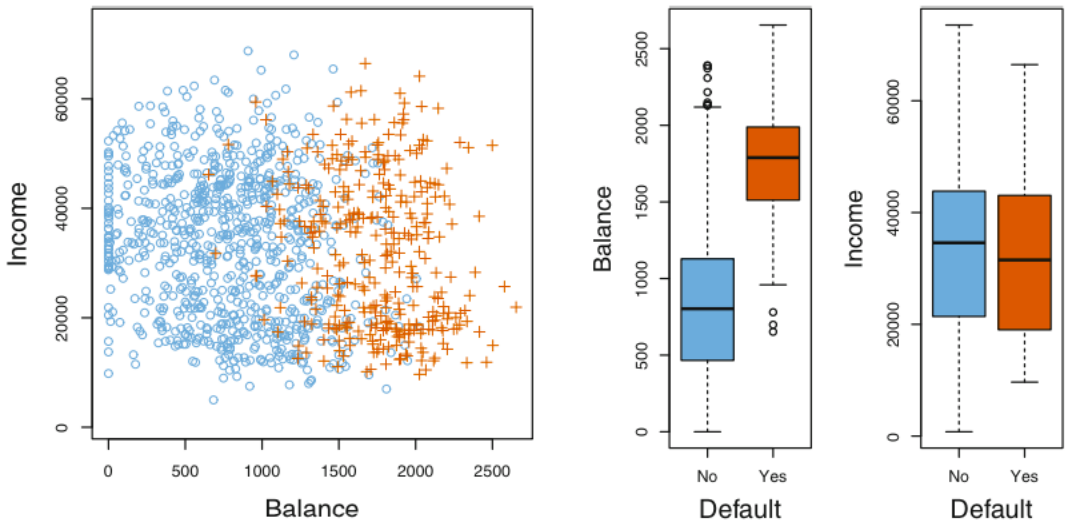
Once again this should be intuitive – if we knew the probability of each class we would pick the most likely class. Similarly, we can evaluate the 0/1 loss of the Bayes classifier and see that:

$$\begin{aligned} R(f_{\text{Bayes}}) &= \mathbb{E}\left[\sum_{k\in\{0,\dots,K-1\}} \mathbb{I}(f_{\text{Bayes}}(X) \neq k)\mathbb{P}(Y=k|X)\right] \\ &= 1 - \mathbb{E}[\max_k \mathbb{P}(Y=k|X)]. \end{aligned}$$

Notice, that in this case the analogy with regression breaks down, i.e. unlike in the binary case we cannot re-write these probabilities as a conditional expectations (i.e. as regressions). We discuss this in more detail in a later section.

2.3 An example

To have an example in mind consider the following classification problem. This is the `default` dataset in ISL. We have two covariates (people's current annual income and their current credit card balances). We would like to predict the customers who will default – the blue circles are the ones who pay back their loans and the red crosses are the ones who default.



We can see that the balance covariate seems quite discriminative – i.e. seems helpful in distinguishing blue circles from red crosses in the picture.

2.4 Classification via Regression

It is first worth understanding in more detail why the relationship between classification and regression breaks down in the multi-class setting.

Here is an example from ISL: Suppose that we are trying to predict the medical condition of a patient in the emergency room on the basis of her symptoms. In this simplified example, there are three possible diagnoses: stroke, drug overdose, and epileptic seizure. We could consider encoding these values as a quantitative response variable, Y as follows:

$$Y = \begin{cases} 1 \text{ if stroke,} \\ 2 \text{ if drug overdose,} \\ 3 \text{ if epileptic seizure.} \end{cases}$$

Now, we can attempt to regress Y on our covariates X. However, notice that this encoding of a categorical variable implies orderings on the outcomes (that weren't present in the original categorical encoding). In particular, we are somehow putting drug overdose in between stroke and epileptic seizure, and insisting that the difference between stroke and drug overdose is the same as the difference between drug overdose and epileptic seizure.

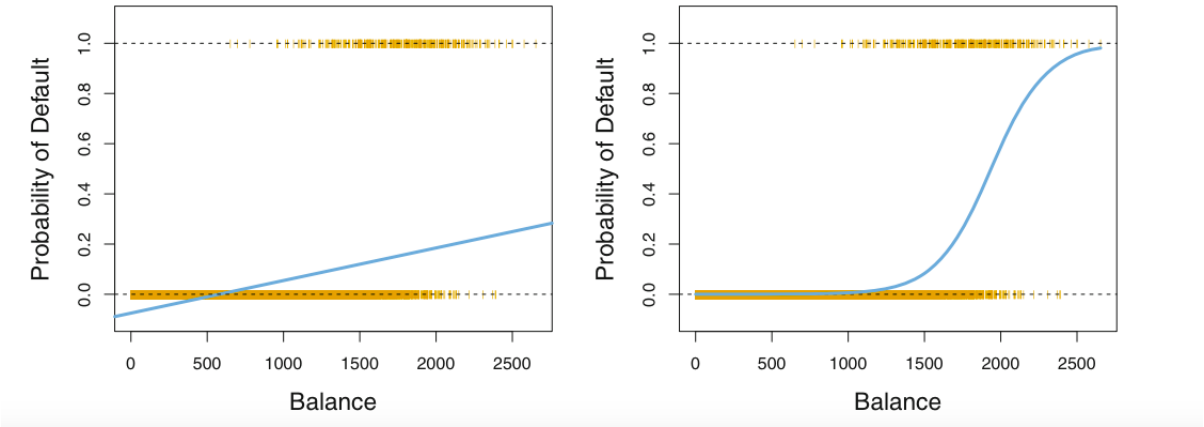
We could equally well use a different encoding:

$$Y = \begin{cases} 1 \text{ if drug overdose,} \\ 2 \text{ if stroke,} \\ 3 \text{ if epileptic seizure,} \end{cases}$$

and our regression would yield a completely different linear model. To summarize – pretending a multivariate categorical variable is a continuous variable is not a good idea.

This actually does not happen for binary classification. As we have seen already binary classification is indeed very similar to regression. In particular, for binary classification our final classifier will not be different if we flip which class we decide to call 0/1 (i.e. it does not depend on the encoding).

Given the similarity between (binary) classification and regression a natural idea would be to use *linear regression* (or to model the regression function by a linear function). This however, is not a very good idea and the following picture illustrates what might go wrong (using the default data).



The figure on the left shows what would happen if we fit a linear regressor to the training data. Notice that it's not a great idea to estimate probabilities using regression – our estimates are often outside the range $[0, 1]$ and they just don't look right. Moreover, we motivated linear regression using the *squared loss* which usually does not make as much sense for classification (especially if we care about the 0/1 loss at the end of the day). The figure on the right shows the fit of a logistic regression classifier – which we can see makes more intuitive sense. We will return to logistic regression in the next lecture.

2.5 Thinking about loss functions

Suppose that we have K classes, then there are $K(K-1)$ *different types of mistakes* that we can make. Most generally, we can thinking of specifying a loss for each of these types of mistakes (and also for getting the answer correct). We can tabulate our guesses and the right answer and the price we pay (i.e. our loss) into a matrix – called the cost matrix or the *confusion matrix*:

$$C = \begin{bmatrix} L_{00} & L_{01} & L_{02} & \dots & L_{0(K-1)} \\ L_{10} & L_{11} & L_{12} & \dots & L_{1(K-1)} \\ \vdots & & & & \\ L_{(K-1)0} & L_{(K-1)1} & L_{(K-1)2} & \dots & L_{(K-1)(K-1)} \end{bmatrix},$$

where L_{ij} is the loss for guessing the label i when the true label is j. When we think of the 0/1 loss we are thinking of the case when the confusion matrix is 0 on the diagonals and 1 everywhere else, i.e. we pay nothing for getting the answer correct and pay 1 for any mistake.

On the other hand, in practice, in most cases we have very asymmetric costs. For instance, suppose we take tumor biopsies from patients and would like to predict whether they have cancer ($Y = 1$) or not ($Y = 0$).

In this case, you might imagine the consequence of mistakenly declaring a person with cancer to be cancer free is much worse than mistakenly declaring a cancer free person to have cancer (assuming that they just go through more rigorous tests). Formally, we would like to enforce that $L_{01} \gg L_{10}$.

In many cases, you can suitably modify the training and evaluation procedures of common classification algorithms to account for these asymmetric costs (and existing software will have this capability built-in). It is however, worth thinking deeply about the loss function for your problem.