A Probabilistic View of Supervised and Unsupervised Learning

Unsupervised learning can be seen as estimating the *joint probability distribution* of all the feature variables, $x_1, \ldots, x_p$. I'll write such joint probabilities (or for continuous features, probability densities) as $P(x_1, \ldots, x_p)$.

If we know the joint probabilities, we can get conditional probabilities for any feature given any others. For example,

$$P(x_1 \,|, \mathbf{x}_2, \ldots, x_p) \quad = \quad \frac{P(x_1, x_2, \ldots, x_p)}{P(x_2, \ldots, x_p)} \quad = \quad \frac{P(x_1, x_2, \ldots, x_p)}{\sum\limits_{x_1} P(x_1, x_2, \ldots, x_p)}$$

Supervised learning focuses on predicting one particular variable, y. We can view it as estimating the *conditional distribution* for y given the other variables, $P(y \,|\, x_1, \ldots, x_p)$, without bothering to estimate $P(x_1, \ldots, x_p)$.

Supervised learning focuses effort on what we're interested in, but at the cost of not being able to predict other variables. Supervised learning also can't easily handle situations where some of $x_1, \ldots, x_p$ were not measured (easy to handle if you have joint probabilities — just condition on fewer variables).

Getting Class Probabilities From Nearest Neighbors

For classification problems, it is easy to get estimates of class probabilities using the k-NN method.

We can just count how many of the k training cases nearest to the test case are in each class, and estimate the probability of a class by its count divided by k.

Sometimes avoiding zero probabilities is desirable, in which case we might add a constant α to each count, and then divide by $k + \alpha C$, if there are C classes.

Using Probabilities to Classify

Consider a classification problem with C classes, labeled 0 to $C-1$.

Suppose that, using the training data, we can estimate the conditional probabilities for the class, y^*, in a test case, given the inputs, x^*, for that test case. In other words, we can compute $P(y^* = c \,|\, x^*)$ for each $c = 0, \ldots, C-1$.

How should we use these probabilities to predict y^*?

The probabilities themselves are the most informative prediction. But suppose we need to make a single guess for y^*.

If we guess that $y^* = c$, the probability of being wrong (if we trust our estimates) is $P(y^* \neq c \,|\, x^*) \;=\; 1 \,-\, P(y^* = c \,|\, x^*)$. This is minimized by choosing c to be the class with the largest probability.

Minimizing the probability of being wrong isn't always the best thing to do, however...

Loss Functions for Classification

Suppose that we're diagnosing patients, with class 0 being "no disease" and each of classes 1 through $C-1$ being different diseases.

How bad is it to diagnose a healthy person as having some disease?
How bad is it to diagnose someone with a disease as being healthy?
How bad is it to diagnose someone with with disease c as having disease c'?

The answers depend on how serious the diseases are, how effectively each disease can be treated, and how bad the side effects of these treatments are.

All this can be summarized in a *loss function*, $L(c, c')$, which measures how bad it is to classify a case as being in class c' when really it is in class c. We assume that $L(c, c') \geq 0$, and $L(c, c) = 0$.

Note: The idea of a loss function can be extended to situations where we can take actions other than just guessing the class. When diagnosing a patient, one action might be to do more tests.

Classifying to Minimize Loss

The loss function and the class probabilities together determine the best guess at the class for a test case. The best guess minimizes the *expected loss*, which is

$$E(L(c,c')) \;\;=\;\; \sum_{c=0}^{C-1} L(c,c')\, P(y^* = c \,|\, x^*)$$

The value of c' that minimizes this is the optimal guess.

When all errors are equally bad, we can use *0–1 loss*, in which $L(c,c') = 1$ when $c \neq c'$. With this loss, $E(L(c,c')) \;=\; \sum_{c \neq c'} P(y^* = c \,|\, x^*) \;=\; 1 - P(y^* = c' \,|\, x^*)$, so the optimal guess is the class with highest probability.

When there are only two classes, all that matters is $L(0,1)/L(1,0)$. It's best to guess class 1 if $P(y^* = 1 \,|\, x^*)$ is greater than some threshold that depends on this ratio. The threshold goes to one as the ratio goes to infinity, and goes to zero as the ratio goes to zero.

Exercise: Find the exact formula for the threshold.

Loss Functions for Regression

If make a guess for a continuous-valued target value, y, we can again measure how bad our guess by means of a loss function. The loss might be particular to the problem, but more often a somewhat conventional loss function is used.

Squared error loss, sometimes written as L_2, is defined by $L_2(y, y') \;=\; (y - y')^2$.

The guess for a test case that minimizes the expected squared error can be found by setting the derivative to zero:

$$\begin{aligned} 0 \;=\; \frac{d}{dy'} E(L(y, y')) \;&=\; \frac{d}{dy'} \int (y - y')^2 \, P(y \,|\, x^*) \, dy \;=\; \int \frac{d}{dy'} \Big[(y - y')^2 \Big] \, P(y \,|\, x^*) \, dy \\ &=\; 2 \int (y' - y) \, P(y \,|\, x^*) \, dy \;=\; 2 \Big(y' - E(y|x^*) \Big) \end{aligned}$$

So the optimal guess will be $E(y|x^*)$, the mean of the conditional distribution of the target given the inputs for the test case.

Absolute error loss, $L_1 = |y - y'|$, puts less emphasis on large errors. The optimal guess for this loss is the median of the conditional distribution for y given x^*.

Exercise: Prove that the median is the optimal guess with L_1 loss.

Models for Residuals in Regression

Finding the conditional distribution, $P(y \,|\, x)$, for a regression model is often split into two sub-problems:

- Finding the *regression function*, $E(y \,|\, x)$.
- Finding the distribution of the *residual* for a case with inputs x, defined as $y - E(y \,|\, x)$. (This distribution always has mean zero.)

With the *maximum likelihood* method, these problems are linked — the best regression function maximizes the probability of the residuals in the training set.

Residuals are often modeled as having a Gaussian (normal) distribution, with the distribution being the same for all cases. We then only need to estimate the standard deviation of this Gaussian distribution.

We might use a non-Gaussian distribution, however. This might be particularly beneficial if there are occasional large residuals.

We might also think that the residuals have different standard deviation for different x — in other words, predicting y is more difficult for some values of x than for others. This is called *heteroskedasticity* (opposite of *homoskedasticity*).

Models for Residuals Vs. Loss Functions

If we model residuals using the Gaussian probability density function,

$$f(z) \;=\; (2\pi\sigma^2)^{-1/2} \exp(-z^2/2\sigma^2)$$

training by maximum likelihood will end up minimizing the squared error on the training cases, since $\log f(z) \;=\; -z^2/2\sigma^2 + \text{constant}$.

The optimal prediction for a test case when we use Gaussian residuals will be the value of the regression function, $E(y|x^*)$, for either squared or absolute error loss. (The mean and median of a Gaussian are the same.)

But in general, maximum likelihood training isn't the same as minimizing loss on training cases. We might use some non-Gaussian model for residuals, even if we use squared error loss. Would this be good? Two views are current:

- We should train using a model of the distribution of residuals that is as good as we can build, then find the appropriate prediction for our loss function.
- We shouldn't use maximum likelihood, or any explicit model of residuals, and instead just minimize the average loss on the training cases. This avoids disasters that might occur if we used a model that was wrong.