Lecture topics: model selection criteria

- Minimum description length (MDL)
- Feature (subset) selection

Model selection criteria: Minimum description length (MDL)

The minimum description length criterion (MDL) turns the model selection problem into a communication problem. The basic idea is that the best model should lead to the best way of compressing the available data. This view equates learning with the ability to compress data. This is indeed a sensible view since learning has to do with the ability to make predictions and effective compression is precisely based on such ability. So, the more we have learned, the better we can compress the observed data.

In a classification context the communication problem is defined as follows. We have a sender and a receiver. The sender has access to the training data, both examples $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and labels $y_1, \ldots, y_n$, and needs to communicate the labels to the receiver. The receiver already has the examples but not the labels.

Any classifier that perfectly classifies the training examples would permit the receiver to reproduce the labels for the training examples. But the sender (say we) would have to communicate the classifier to the receiver before they can run it on the training examples. It is this part that pertains to the complexity of the set of classifiers we are fitting to the data. The more choices we have, the more bits it takes to communicate the specific choice that perfectly classified the training examples. We can also consider classifiers that are not perfectly accurate on the training examples by communicating the errors that they make. As a result, the communication cost, the number of bits we have to send to the receiver, depends on two things: the cost of errors on the training examples (description length of the data given the model), and the cost of describing the selected classifier (description length of the model), both measured in bits.

Simple two-part MDL. Let's start by defining the basic concepts in a simpler context where we just have a sequence of labels. You could view the examples $\mathbf{x}_i$ in this case just specifying the indexes of the corresponding labels. Consider the following sequence of mostly 1's:

$$\overbrace{1 \;\; 1 \;\; 1 \; -1 \;\; 1 \;\; 1 \;\; 1 \;\; 1 \;\; 1 \; -1 \;\; 1 \ldots}^{n \text{ labels}} \tag{1}$$

If we assume that each label in the sequence is an independent sample from the same distribution, then each $y_t \in \{-1, 1\}$ is a sample from

$$P(y_t|\theta) = \theta^{\delta(y_t,1)}(1-\theta)^{\delta(y_t,-1)} \tag{2}$$

for some θ. Here $\delta(y_1, 1)$ is an indicator such that $\delta(y_t, 1) = 1$ if $y_t = 1$ and zero otherwise. In our model $P(y_t = 1|\theta) = \theta$. We don't necessarily know that this is the correct model for the data. But whatever model (here a set of distributions) we choose, we can evaluate the resulting communication cost.

Now, the question is: given this model for the data, how many bits do we need to send the observed sequence of n labels to the receiver? This can be answered generally. Any distribution over the data can be used as a basis for a coding sceme. Moreover, the optimal coding scheme, assuming the data really did follow our distribution, would require $-\log_2 P(y_1, \ldots, y_n|\theta)$ bits (base two logarithm) or $-\log P(y_1, \ldots, y_n|\theta)$ nats (natural logarithm). We will use the natural logarithm for consistency with other material. You can always convert the answer to bits by multiplying nats by $\log(2)$.

Since our model assumes that the labels are independent of each other, we get

$$-\log P(y_1, \ldots, y_n|\theta) = \sum_{t=1}^{n} -\log P(y_t|\theta) \tag{3}$$

It is now tempting to minimize this encoding cost with respect to θ (maximizing the log-likelihood):

$$\begin{aligned} \min_{\theta} \left\{ -\sum_{t=1}^{n} \log P(y_t|\theta) \right\} &= -\sum_{t=1}^{n} \log P(y_t|\hat{\theta}) && (4) \\ &= -l(y_1, \ldots, y_n; \hat{\theta}) && (5) \end{aligned}$$

where $l(D; \theta)$ is again the log-likelihood of data D and $\hat{\theta}$ is the maximum likelihood setting of the parameters. This corresponds to finding the distribution in our model (one corresponding to $\hat{\theta}$) that minimizes the encoding length of the data. The problem is that the receiver needs to be able to *decode* what we send. This is only possible if they have access to the same distribution (be able to recreate the code we use). So the receiver would have to know $\hat{\theta}$ (in addition to the model we are considering but we will omit this part, assuming that the receiver already knows the type of distributions we are have).

The solution is to encode the choice of $\hat{\theta}$ as well. This way the receiver can recreate our steps to decode the bits we have sent. So how do we encode $\hat{\theta}$? It is a continuous parameter and it might seem that we need an infinite number of bits to encode a real number. But not all θ can arise as ML estimates of the parameters. In our case, the ML estimates of the parameters θ have the following form

$$\hat{\theta} = \frac{\hat{n}(1)}{n} \tag{6}$$

where $\hat{n}(1)$ is the number of 1's in the observed sequence. There are thus only $n+1$ possible values of $\hat{\theta}$ corresponding to $\hat{n}(1) \in \{0, \ldots, n\}$. Since the receiver already knows that we are encoding a sequence of length n, then we can just send θ by defining a distribution over its possible discrete values, say $Q(\theta = k/n) = \frac{1}{n+1}$ for simplicity (i.e., a uniform distribution over possible discrete values). As before the cost in bits (nats) is $-\log Q(\hat{\theta}) = \log(n+1)$.

The total cost we have to send to the receiver is therefore

$$\text{DL} = \overbrace{-l(y_1, \ldots, y_n; \hat{\theta})}^{\text{DL of data given model}} + \overbrace{\log(n+1)}^{\text{DL of model}} \tag{7}$$

This is known as *the description length* of the sequence. The minimum description length criterion simple finds the model that leads to the shortest description length. Asymptotically, the simple two-part MDL principle is essentially the same as BIC.

Universal coding, normalized maximum likelihood. The problem with the simple two part coding is that the description length we associate with the second part, the model, may seem a bit arbitrary. We can correct this by finding a distribution that is in some sense universally closest to just encoding the data with the best fitting distribution in our model:

$$-\log P_{NML}(y_1, \ldots, y_n) \approx \min_{\theta} -l(y_1, \ldots, y_n; \theta) = -\max_{\theta} l(y_1, \ldots, y_n; \theta) \tag{8}$$

In other words, we don't wish to define a prior over the parameters in our model so as to encode the parameter values. While it won't be possible to achieve the above approximate equality with equality since the right hand side, if exponentiated, wouldn't define a valid distribution (because of the ML fitting). The distribution that minimizes the maximum deviation in Eq.(8) is given by

$$P_{NML}(y_1, \ldots, y_n) = \frac{\exp\left(\max_{\theta} l(y_1, \ldots, y_n; \theta)\right)}{\sum_{y_1', \ldots, y_n'} \exp\left(\max_{\theta} l(y_1', \ldots, y_n'; \theta)\right)} \tag{9}$$

and is known as the *normalized maximum likelihood distribution*. Using this distribution to encode the sequence leads to a minimax optimal coding length

$$-\log P_{NML}(y_1,\ldots,y_n) = -\max_{\theta} l(y_1,\ldots,y_n;\theta) + \overbrace{\log \sum_{y_1',\ldots,y_n'} \exp\left(\max_{\theta} l(y_1',\ldots,y_n';\theta)\right)}^{\text{Parametric complexity of model}} \quad (10)$$

Note that the result is again in the same form, the negative log-likelihood with a ML parameter setting and a complexity penalty. The new parametric complexity penalty depends on the length of the sequence as well as the model we use. While the criterion is theoretically nice, it is hard to evaluate the parametric complexity penalty in practice.

Feature subset selection

Feature selection is a problem where we wish to identify components of feature vectors most useful for classification. For example, some components may be very noisy and therefore unreliable. Depending on how such features[1] would be treated in the classification method, it may be best not to include them. Here we assume that we have no prior knowledge of which features might be useful for solving the classification task and instead have to provide an automated solution. This is a type of model selection problem, where the models are identified with subsets of the features we choose to include (or the number of features we wish to include). Note that kernels do not solve nor avoid this problem. By taking an inner product between feature vectors that may contain many noisy or irrelevant features results in an unnecessarily varying kernel value.

Instead of selecting a subset of the features we could also try to weight them according to how useful they are. This is a related *feature weighting* problem and we will get to this later on. This is also the method typically used with kernels (cf. kernel optimization).

Let's address the feature selection problem a bit more concretely. Suppose our input features are $d-$dimensional binary vectors $\mathbf{x} = [x_1, ..., x_d]^d$ where $x_i \in \{-1, 1\}$ and labels are binary $y \in \{-1, 1\}$ as before. We will begin by using a particular type of probability model known as *Naive Bayes* model to solve the classification problem. This will help tie our MDL discussion to the feature selection problem. In the Naive Bayes model, we assume that the features are conditionally independent given the label so that

$$P(\mathbf{x}, y) = \left[\prod_{i=1}^{d} P(x_i|y)\right] P(y) \quad (11)$$

[1]We will refer to the components of the feature/input vectors as "features".

Put another way, in this model, the features are correlated only through the labels. We will contrast this model with a "null model" that assumes that none of the features are relevant for the label:

$$P_0(\mathbf{x}, y) = \left[\prod_{i=1}^{d} P(x_i)\right] P(y) \tag{12}$$

In other words, in this model we assume that the features and the label are all independent of each other.

The Naive Bayes model (as well as the null model) are *generative models* in the sense that they can generate all the data, not just the labels. These models are typically trained using the log-likelihood of all the data as the estimation criterion. Let us follow this here as well. We will discuss later on how this affects the results. So, we aim to maximize

$$\begin{aligned}
\sum_{t=1}^{n} \log P(\mathbf{x}_t, y_t) &= \sum_{t=1}^{n} \left[\sum_{i=1}^{d} \log P(x_{ti}|y_t) + \log P(y_t)\right] &(13)\\
&= \sum_{i=1}^{d} \sum_{t=1}^{n} \log P(x_{ti}|y_t) + \sum_{t=1}^{n} \log P(y_t) &(14)\\
&= \sum_{i=1}^{d} \left(\sum_{x_i,y} \hat{n}_{iy}(x_i,y) \log P(x_i|y)\right) + \sum_{y} \hat{n}_y(y) \log P(y_t) &(15)
\end{aligned}$$

where we have used counts $\hat{n}_{iy}(x_i, y)$ and $\hat{n}_y(y)$ to summarize the available data for the purpose of estimating the model parameters. $\hat{n}_{iy}(x_i, y)$ indicates the number of training instances that have value x_i for the i^{th} feature and y for the label. These are called *sufficient statistics* as they are the only things from the data that the model cares about (the only numbers computed from the data that are required to estimate the model parameters). The ML parameter estimates, values that maximize the above expression, are simply fractions of these counts:

$$\begin{aligned}
\hat{P}(y) &= \frac{\hat{n}_y(y)}{n} &(16)\\
\hat{P}(x_i|y) &= \frac{\hat{n}_{iy}(x_i,y)}{\hat{n}_y(y)} &(17)
\end{aligned}$$

As a result, we also have $\hat{P}(x_i, y) = \hat{P}(x_i|y)\hat{P}(y) = \hat{n}_{iy}(x_i, y)/n$. If we plug these back into the expression for the log-likelihood, we get

$$\begin{aligned}
\hat{l}(S_n) &= \sum_{t=1}^{n} \log \hat{P}(\mathbf{x}_t, y_t) && (18)\\
&= n \left[\sum_{i=1}^{d} \left(\sum_{x_i, y} \frac{\hat{n}_{iy}(x_i, y)}{n} \log \hat{P}(x_i|y) \right) + \sum_{y} \frac{\hat{n}_y(y)}{n} \log \hat{P}(y_t) \right] && (19)\\
&= n \left[\sum_{i=1}^{d} \overbrace{\sum_{x_i, y} \hat{P}(x_i, y) \log \hat{P}(x_i|y)}^{-\hat{H}(X_i|Y)} + \overbrace{\sum_{y} \hat{P}(y) \log \hat{P}(y_t)}^{-\hat{H}(Y)} \right] && (20)\\
&= n \left[-\sum_{i=1}^{d} \hat{H}(X_i|Y) - \hat{H}(Y) \right] && (21)
\end{aligned}$$

where $\hat{H}(Y)$ is the *Shannon entropy* (uncertainty) of y relative to distribution $\hat{P}(y)$ and $\hat{H}(X_i|Y)$ is the *conditional entropy* of X_i given Y. Entropy $H(Y)$ measures how many bits (here nats) we would need on average to encode a value y chosen at random from $\hat{P}(y)$. It is zero when y is deterministic and takes the largest value (one bit) when $\hat{P}(y) = 1/2$ in case of binary labels. Similarly, the conditional entropy $\hat{H}(X_i|Y)$ measures how uncertain we would be about X_i if we already knew the value for y and the values for these variables were sampled from $\hat{P}(x_i, y)$. If x_i is perfectly predictable from y according to $\hat{P}(x_i, y)$, then $\hat{H}(X_i|Y) = 0$.

We can perform a similar calculation for the null model and obtain

$$\hat{l}_0(S_n) = \sum_{t=1}^{n} \log \hat{P}_0(\mathbf{x}_t, y_t) = n \left[-\sum_{i=1}^{d} \hat{H}(X_i) - \hat{H}(Y) \right] \qquad (22)$$

Note that the conditioning on y is gone since the features were assumed to be independent of the label. By taking a difference of these two, we can gauge how much we gain in terms of the resulting log-likelihood if we assume the features to depend on the label:

$$\hat{l}(S_n) - \hat{l}_0(S_n) = n \sum_{i=1}^{d} \left(\hat{H}(X_i) - \hat{H}(X_i|Y) \right) \qquad (23)$$

The expression

$$\hat{I}(X_i; Y) = \hat{H}(X_i) - \hat{H}(X_i|Y) \qquad (24)$$

is known as the *mutual information* between x_i and y, relative to distribution $\hat{P}(x_i, y)$. Mutual information is *non-negative* and *symmetric*. In other words,

$$\hat{I}(X_i; Y) = \hat{H}(X_i) - \hat{H}(X_i|Y) = \hat{H}(Y) - \hat{H}(Y|X_i) \tag{25}$$

To understand mutual information in more detail, please consult, e.g., Cover and Tomas listed on the course website.

So, as far as feature selection is concerned, our calculation so far suggests that we should select the features to include in the decreasing order of $\hat{I}(X_i; Y)$ (the higher the mutual information, the more we gain in terms of log-likelihood). It may seem a bit odd, however, that we can select the features individually, i.e., not consider them in specific combinations. This is due to the simple Naive Bayes assumption, and our use of the log-likelihood of all the data to derive the selection criterion. The mutual information criterion is nevertheless very common and it is useful to understand how it comes about. We will now turn to improving this a bit with MDL.