Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

6.S064 Introduction to Machine Learning

Phase 3: model selection, generalization (lecture 14)

Model selection for classifiers

In the last lecture, we discussed the problem selecting the number of components to use in a mixture model, introducing the BIC-score as an approximate selection criterion. The BIC score was composed of two parts, 1) the log-likelihood of the training data, maximized with respect to the available parameters, and 2) a penalty term that offsets the likelihood gain from having many parameters. We will do the same here for classification problems, replacing log-likelihood with a classification error. However, we go a step further, casting the model selection problem in terms of finding a model with the best guarantees of generalization.

To fix ideas, let's define our classification task a bit more precisely. We assume that training and test examples are drawn independently at random from some fixed but unknown distribution P^* over (x, y). So, for example, each $(x^{(t)}, y^{(t)})$ in the training set $S_n = \{(x^{(t)}, y^{(t)}), t = 1, \ldots, n\}$, is sampled from P^*. You can view $P*$ as a large (infinite) pool of (x, y) pairs and each (x, y), whether training or test, is simply drawn at random from this pool. We do not know P^* (though could, and will, try to estimate it later). The major assumption is that the training and test examples and labels come from the *same* underlying distribution P^*.

The training error of any classifier $h(x) \in \{-1, 1\}$ is measured, as before, as counting the number of mistakes on the training set S_n

$$\mathcal{E}_n(h) = \frac{1}{n} \sum_{t=1}^{n} [\![\, y^{(t)} h(x^{(t)}) \leq 0 \,]\!] \tag{1}$$

The test or generalization error is defined as the expected value

$$\mathcal{E}(h) = E_{(x,y)\sim P^*} [\![\, yh(x) \leq 0 \,]\!] \tag{2}$$

where you can imagine drawing (x, y) from the large "pool" P^* and averaging the results. Note that the training error will change if we measure it based on another set of n examples S'_n drawn from P^*. The generalization error does not vary for any fixed $h(x)$, however, as this error is measured across the whole pool of examples already. Also note that the generalization error $\mathcal{E}(h)$ is also the probability that $h(x)$ would misclassify a randomly drawn example from P^*.

Given a set of classifiers $\mathcal{H}$, we would like to choose h^* that minimizes the generalization error, i.e., $\mathcal{E}(h^*) = \min_{h\in\mathcal{H}} \mathcal{E}(h)$. If we had access to $\mathcal{E}(h)$, there would be no

model selection problem either. We would simply select the largest set $\mathcal{H}$ so as to find a classifier that minimizes the error. But we only have access to $\hat{h}$ that minimizes the training error, $\hat{h} \in \arg\min_{h \in H} \mathcal{E}_n(h)$, and still wish to guarantee that the generalization error $\mathcal{E}(\hat{h})$ is low. A large $\mathcal{H}$ can hurt us in this setup. The basic intuition is that if $\mathcal{H}$ involves too many choices, we may select $\hat{h}$ that fits the noise rather than the signal in the training set. Any characteristics of $\hat{h}$ that are based on noise will not generalize well. We must select the appropriate "model" $\mathcal{H}$.

Suppose we have sets of classifiers $\mathcal{H}_i$, $i = 1, 2, \ldots,$, ordered from simpler to complex. We assume that these sets are nested in the sense that the more complex ones always include the simpler ones, i.e., $H_1 \subseteq H_2 \subseteq H_3 \subseteq \ldots$. So, for example, if $h \in H_1$, then $h \in H_2$ as well. The nested sets ensure, for example, that the training error will go down if we adopt a more complex set of classifiers. In terms of linear classifiers,

$$H = \left\{ h : \ s.t.\ h(x) = \text{sign}(\phi(x) \cdot \theta + \theta_0), \text{ for some } \ \theta \in \mathcal{R}^p, \ \theta_0 \in \mathcal{R} \right\}, \tag{3}$$

the sets H_i could correspond to the degree of polynomial features in $\phi(x)$. For example, in two dimensions,

$$H_1 : \ \phi(x) = [x_1, x_2]^T \tag{4}$$
$$H_2 : \ \phi(x) = [x_1, x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2]^T \tag{5}$$
$$\ldots$$

Note that H_2 contains the first order features as well as the additional 2nd order ones. So, any $h \in H_1$ has the equivalent classifier in H_2 simply by setting the parameters corresponding to the 2nd order features to zero.

Our goal here is to find the set of classifiers $\mathcal{H}_i$ such that, if we choose $\hat{h}_i \in \mathcal{H}_i$ by minimizing the training error, $\mathcal{E}_n(\hat{h}_i)$, we obtain the best guarantee of generalization error $\mathcal{E}(\hat{h}_i)$. In other words, we select the model (set of classifiers) for which we can guarantee the best generalization error. The remaining problem is to find such generalization guarantees.

Generalization guarantees

Our goal here is to provide some generalization guarantees for classifiers chosen based on the training set S_n. We assume, for simplicity, that the training algorithm finds a classifier $\hat{h} \in \mathcal{H}$ that minimizes the training error $\mathcal{E}_n(\hat{h})$. What can we say about the resulting generalization error $\mathcal{E}(\hat{h})$? Well, since the training set S_n is a random draw from P^*, our trained classifier $\hat{h}$ also varies according to this random draw. So, the generalization error $\mathcal{E}(\hat{h})$ also varies through $\hat{h}$. We cannot simply give a absolute bound on the generalization error. We can only say that with high probability, where the probability refers to the choice of the training examples in S_n, we find $\hat{h}$ that would

generalize well. After all, we might have been very unlucky with the training set, and therefore obtain a bad $\hat{h}$ that generalizes poorly. But we want the procedure to work most of the time, in all "typical" cases.

More formally, we are looking for PAC (Probably Approximately Correct) guarantees of the form: *With probability at least* $1-\delta$ *over the choice of the training set* S_n *from* P^**, any classifier* $\hat{h}$ *that minimizes the training error* $\mathcal{E}_n(\hat{h})$ *has generalization error* $\mathcal{E}(\hat{h}) \leq \epsilon$. Note that the statement is always true if we set $\epsilon = 1$ (since generalization error is bounded by one). Moreover, it is easy to satisfy the statement if we increase δ, i.e., require that good classifiers are found only in response to a small fraction $1-\delta$ of possible training sets. A key task here is to find the smallest ϵ for a given δ, n, and $\mathcal{H}$. In other words, we wish to find the best guarantee of generalization (i.e., ϵ) that we can give with confidence $1-\delta$ (fraction of training sets for which the guarantee is true), the number of training examples n, and the set of classifiers $\mathcal{H}$ (in particular, some measure of the size of this set). In looking for such guarantees, we will start with a simple finite case.

A finite set of classifiers

Suppose $\mathcal{H} = \{h_1, \ldots, h_K\}$, i.e., there are at most K distinct classifiers in our set. We'd like to understand, in particular, how $|\mathcal{H}| = K$ relates to ϵ, the guarantee of generalization. To make our derivations simpler, we assume, in addition, that there exists some $h^* \in \mathcal{H}$ with zero generalization error, i.e., $\mathcal{E}(h^*) = 0$. This additional assumption is known as the *realizable* case: there exists a perfect classifier in our set, we just don't know which one.

Our derivation proceeds as follows. We will fix ϵ, $\mathcal{H}$, and n, and try to obtain the smallest δ so that the guarantee holds (recall, δ is the probability over the choice of the training set that our guarantee fails). To this end, let $h \in \mathcal{H}$ be any classifier that generalizes poorly, i.e., $\mathcal{E}(h) > \epsilon$. What is the probability that we would consider it after seeing the training set? It is the probability that this classifier makes no errors on the training set. This is at most $(1-\epsilon)^n$ since the probability that it makes an error on any example drawn from P^* is at least ϵ. But there may be many such "offending" classifiers that have high generalization error (above ϵ), yet appear good on the training set (zero training error). Clearly, there cannot be more than $|\mathcal{H}|$ of such classifiers. Taken together, we clearly have that $\delta \leq |\mathcal{H}|(1-\epsilon)^n$. So,

$$\log \delta \leq \log |\mathcal{H}| + n \log(1-\epsilon) \leq \log |\mathcal{H}| - n\epsilon \tag{6}$$

where we used the fact that $\log(1-\epsilon) \leq -\epsilon$. Solving for ϵ, we get

$$\epsilon \leq \frac{\log |\mathcal{H}| + \log(1/\delta)}{n} \tag{7}$$

In other words, the generalization error of any classifier $\hat{h}$ that achieves zero training error (under our two assumptions) is bounded by the right hand side in the above expression. Note that

- For good generalization (small ϵ), we must ensure that $\log|\mathcal{H}|$ is small compared to n. In other words, the "size" of the set of classifiers cannot be too large. What matters is the logarithm of the size.
- The more confident we wish to be about our guarantee (the smaller δ is), the more training examples we need. Clearly, the more training examples we have, the more confident we will be that a classifier which achieves zero training error will also generalize well.

The analysis is slightly more complicated if we remove the assumption that there has to be one perfect classifier in our set. The resulting guarantee is weaker but with similar qualitative dependence on the relevant quantities: with probability at least $1-\delta$ over the choice of the training set,

$$\mathcal{E}(\hat{h}) \leq \mathcal{E}_n(\hat{h}) + \sqrt{\frac{\log|\mathcal{H}| + \log(2/\delta)}{2n}} \tag{8}$$

where $\hat{h}$ is the classifier that minimizes the training error. In fact, in this case, the guarantee holds for all $\hat{h} \in \mathcal{H}$, not only for the classifier that we would choose based on the training set. The result merely shows how many examples we would need in relation to the size of the set of classifiers so that the generalization error is close to the training error *for all* classifiers in our set.

What happens to our analysis if $\mathcal{H}$ is not finite? $\log|\mathcal{H}|$ is infinite, and the results are meaningless. We must count the size of the set of classifiers differently when there are continuous parameters as in linear classifiers.

Growth function and the VC-dimension

The set of linear classifiers is an uncountably infinite set. How do we possibly count them? We will try to understand this set in terms of how classifiers from this set label training examples. In other words, we can think of creating a matrix where each row corresponds to a classifier and each column corresponds to a training example. Each entry of the matrix tells us how a particular classifier labels a specific training example. Note that there are infinite rows in this matrix and exactly n columns.

$$\begin{array}{rccccc} & x^{(1)} & x^{(2)} & \cdots & x^{(n)} \\ h \in \mathcal{H}: & +1 & -1 & \cdots & -1 \\ h' \in \mathcal{H}: & +1 & -1 & \cdots & -1 \\ h'' \in \mathcal{H}: & +1 & +1 & \cdots & -1 \\ \cdots & \cdots & \cdots & \cdots & \cdots \end{array} \tag{9}$$

Not all the rows in this matrix are distinct. In fact, there can be only at most 2^n distinct rows. But our set of classifiers may not be able to generate all the 2^n possible labelings. Let $N_{\mathcal{H}}(x^{(1)}, \ldots, x^{(n)})$ be the number of distinct rows in the matrix if we choose classifiers

from $\mathcal{H}$. Since this depends on the specific choice of the training examples, we look at instead the maximum number of labelings (distinct rows) that can be obtained with the same number of points:

$$N_{\mathcal{H}}(n) = \max_{x^{(1)},\ldots,x^{(n)}} N_{\mathcal{H}}(x^{(1)},\ldots,x^{(n)}) \tag{10}$$

This is known as the *growth function* and measures how powerful the set of classifiers is. The relevant measure of the size of the set of classifiers is now $\log N(n)$ (again, the logarithm of the "number"). Indeed, using this as a measure of size already gives us a generalization guarantee similar to the case of finite number of classifiers: with probability at least $1-\delta$ over the choice of the training set,

$$\mathcal{E}(\hat{h}) \leq \mathcal{E}_n(\hat{h}) + \sqrt{\frac{\log N_{\mathcal{H}}(2n) + \log(4/\delta)}{n}}, \quad \text{for all } \hat{h} \in \mathcal{H} \tag{11}$$

The fact that the guarantee uses $\log N_{\mathcal{H}}(2n)$ rather than $\log N_{\mathcal{H}}(n)$ comes from a particular technical argument (symmetrization) used to derive the result. The key question here is how the new measure of size, i.e., $\log N_{\mathcal{H}}(n)$, grows relative to n. When $N_{\mathcal{H}}(n) = 2^n$, we have $\log N_{\mathcal{H}}(n) = n\log(2)$ and the guarantee remains vacuous. Indeed, our guarantee becomes interesting only when $\log N_{\mathcal{H}}(n)$ grows much slower than n. When does this happen?

This key question motivates us to define a measure of complexity of a set of classifiers, the *Vapnik-Chervonenkis* dimension or VC-dimension for short. It is the largest number of points for which $N_{\mathcal{H}}(n) = 2^n$, i.e., the largest number of points that can be labeled in all possible ways by choosing classifiers from $\mathcal{H}$. Let $d_{\mathcal{H}}$ be the VC-dimension. It turns out that $N_{\mathcal{H}}(n)$ grows much slower after n is larger than the VC-dimension $d_{\mathcal{H}}$. Indeed, when $n > d_{\mathcal{H}}$,

$$\log N_{\mathcal{H}}(2n) \leq d_{\mathcal{H}}(\log(2n/d_{\mathcal{H}}) + 1) \tag{12}$$

In other words, the number of labelings grows only logarithmically. As a result, the VC-dimension $d_{\mathcal{H}}$ captures a clear threshold for learning: when we can and cannot guarantee generalization.

So, what is the VC-dimension of a set of linear classifiers? It is exactly $d+1$ (the number of parameters in d-dimensions). This relation to the number of parameters is often but not always true. The figure below illustrates that the VC-dimension of the set of linear classifiers in two dimensions is exactly 3.

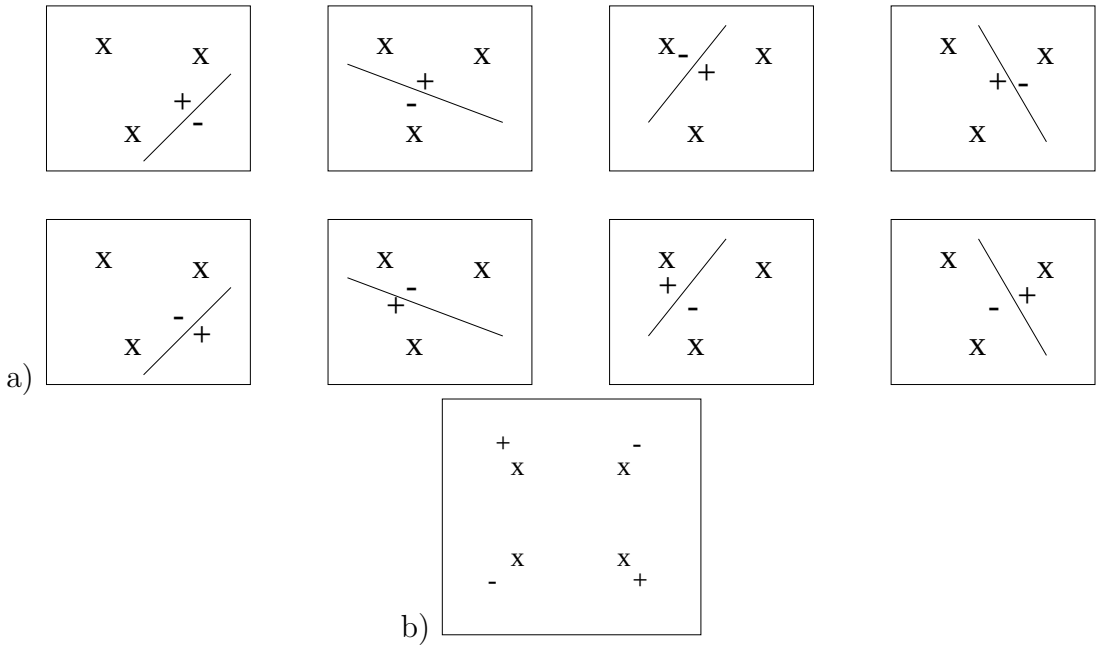

Figure 1: a) The set of linear classifiers in 2d can label three points in all possible ways; b) a labeling of four points that cannot be obtained with a linear classifier.