

## Over-fitting and model selection

We have so far paid relatively little attention to the complexity of the set of classifiers or regression methods we are fitting to finite data. We have therefore not directly addressed the problem of *over-fitting* or fitting too complex a model to too few data points. Avoiding over-fitting lies at the core of machine learning.

Consider a simple two dimensional classification problem in Figure 1. We can use increasingly high order polynomial kernels to fit to the same data points. For example, we need an 8<sup>th</sup> order polynomial kernel to perfectly separate the data points in Figure 1. The classifiers corresponding to high order polynomial kernels are not likely to generalize well, however. The correct answer for the figure is a linear kernel (we know how the data points were generated). How can we identify the right degree of polynomial kernel to use? The typical solution in practice is to select the classifier (kernel) that leads to the lowest 10-fold or leave-one-out cross-validation error. This would actually work fine for the data in Figure 1. While it often suffices in practice, the cross-validation heuristic does not help us understand the trade-offs between the number of training examples and the complexity of the model to use. We need a bit more machinery to understand such issues.

## Model / kernel selection

Here we will cast the problem of selecting a kernel as a *model selection* problem. By choosing a kernel we specify the feature vectors on the basis of which linear predictions are made. Each model<sup>1</sup> (class) refers to a set of linear functions (classifiers) based on the chosen feature representation. In many cases the models are nested in the sense that the more “complex” model contains the “simpler” one. Consider, for example, solving a classification problem with either

$$K_1(\underline{x}, \underline{x}') = (1 + \underline{x}^T \underline{x}') \text{ or} \quad (1)$$

$$K_2(\underline{x}, \underline{x}') = (1 + \underline{x}^T \underline{x}')^2 \quad (2)$$

Classifiers making use of the quadratic polynomial kernel can in principle reproduce the classifiers based on the linear kernel. As a model, i.e., as a set of linear classifiers based on the quadratic kernel, the quadratic model contains the simpler linear one. We can state this a bit more formally in terms of discriminant functions. For example, based on the linear kernel  $K_1$ , our discriminant functions are of the form

$$f_1(\underline{x}; \underline{\theta}, \theta_0) = \underline{\theta}^T \underline{\phi}^{(1)}(\underline{x}) + \theta_0 \quad (3)$$

---

<sup>1</sup>In statistics, a model is a family/set of distributions or a family/set of linear separators.

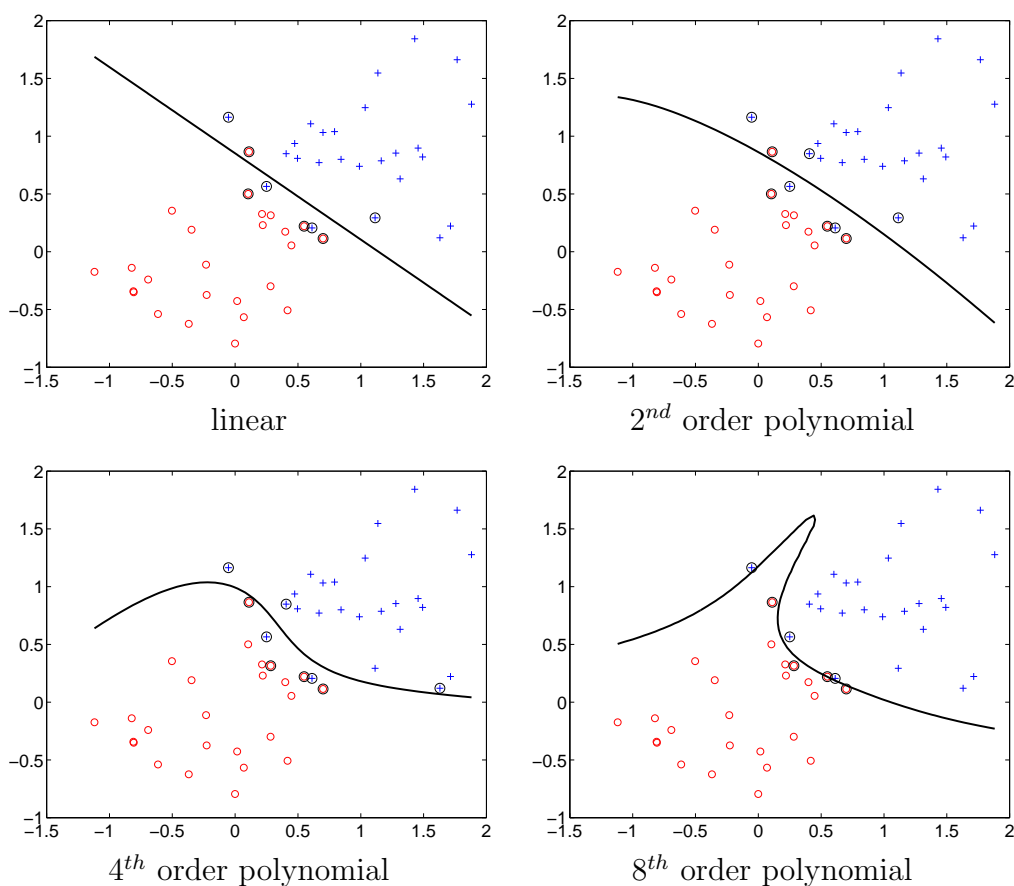


Figure 1: Classifiers with different polynomial kernels

where  $\phi^{(1)}(\underline{x})$  is the feature representation corresponding to  $K_1$  such that  $K_1(\underline{x}, \underline{x}') = \phi^{(1)}(\underline{x}) \cdot \phi^{(1)}(\underline{x}')$ . By varying the parameters  $\underline{\theta}$  and  $\theta_0$  we can generate the set of possible discriminant functions corresponding to this kernel:

$$\mathcal{F}_1 = \{f_1(\cdot; \underline{\theta}, \theta_0) : \underline{\theta} \in \mathcal{R}^{d_1}, \theta_0 \in \mathcal{R}\} \quad (4)$$

$\mathcal{F}_2$  is defined analogously for the quadratic kernel. The fact that the two models are nested means that  $\mathcal{F}_1 \subseteq \mathcal{F}_2$ . For purposes of classification, we wouldn't actually have to assert that the families of discriminant functions are nested, only that the discriminant functions in  $\mathcal{F}_2$  can produce the signs of those in  $\mathcal{F}_1$ .

The formal problem for us to solve is then to select a kernel  $K_i$  from a set of possible kernels

$K_1, K_2, \dots$ , where the models associated with the kernels are nested  $\mathcal{F}_1 \subseteq \mathcal{F}_2 \subseteq \dots$ . This is a model selection problem in a standard nested form.

From here on we will be referring to discriminant functions rather than kernels so as to emphasize the point that the discussion applies to other types of classifiers as well.

### Model selection preliminaries

Before getting into specific selection criteria let's understand a bit better what exactly we are doing here. Recall that our goal is to accurately classify new test examples. Model selection is intended to facilitate this process. In other words, we switch from one model (kernel) to another so as to generalize better. The model we select will define how we will respond to any training data, i.e., which classifier we choose to make predictions on new examples. Model selection cannot therefore be decoupled from how we find the "best fitting" classifier from a given model. After all, it is that best fitting classifier that will determine how well we generalize.

Let  $S_n = \{(\underline{x}_1, y_1), \dots, (\underline{x}_n, y_n)\}$  denote a training set of  $n$  examples and labels. If we chose model  $\mathcal{F}_i$  then we would find the best fitting discriminant function  $\hat{f}_i \in \mathcal{F}_i$  by minimizing

$$J(\underline{\theta}, \theta_0) = \sum_{t=1}^n \text{Loss}\left(y_t, f(\underline{x}_t; \underline{\theta}, \theta_0)\right) + \lambda_n \|\underline{\theta}\|^2 \quad (5)$$

where the loss could be the hinge loss (SVM), logistic, or other. The regularization parameter  $\lambda_n$  would in general depend on the number of training examples. We are interested in how the classifier  $\hat{f}_i(\underline{x}) = f(\underline{x}; \hat{\underline{\theta}}, \hat{\theta}_0)$  resulting from our estimation procedure generalizes to new examples.

Each parameter setting  $(\underline{\theta}, \theta_0)$ , i.e., each discriminant function in our set, has an associated expected loss or *risk*

$$R(\underline{\theta}, \theta_0) = E_{(\underline{x}, y) \sim P} \left\{ \text{Loss}_{0-1}\left(y, f(\underline{x}; \underline{\theta}, \theta_0)\right) \right\} \quad (6)$$

where the new test example and label,  $(\underline{x}, y)$ , is sampled from an underlying distribution  $P$  which is typically unknown to us. This is the generalization error we would like to minimize. Note that we have used  $\text{Loss}_{0-1}(\cdot, \cdot)$  above rather than the loss used in training. These need not be the same and often they are not. For example, our goal may be to minimize classification error so that  $\text{Loss}_{0-1}(y, f(\underline{x})) = 1$  if  $yf(\underline{x}) \leq 0$  and zero otherwise (zero-one loss). We could still estimate the SVM classifier from the training set in the usual

way, optimizing the hinge loss. The hinge loss can be viewed as a convex surrogate for the zero-one loss and it behaves much better in terms of the resulting optimization problem we have to solve during training (quadratic rather than integer programming problem).

The quantity of interest to us is the generalization error  $R(\hat{\theta}, \hat{\theta}_0)$ , or  $R(\hat{f}_i)$  for short, corresponding to the classifier or discriminant function we would choose from  $\mathcal{F}_i$  in response to the training data  $S_n$ . Ideally, we would then select the model  $\mathcal{F}_i$  that leads to the smallest generalization error, minimizing

$$R(\hat{f}_i) = E_{(\underline{x}, y) \sim P} \left\{ \text{Loss}_{0-1} \left( y, \hat{f}_i(\underline{x}) \right) \right\} \quad (7)$$

Note that the risk  $R(\hat{f}_i)$  is still a random variable as  $\hat{f}_i$  depends on the training data that we assume was also sampled from the same underlying distribution  $P$ . If the training data were sampled from a different distribution, how could we expect to generalize? Actually, the only thing we really need is that the relationship between the labels and examples is the same for the training and test samples, along with some guarantee that the training examples cover the areas of input space that we will be tested on. In theoretical analysis it is nevertheless much more convenient to assume that the distributions are the same.

Now, we clearly do not have access to the underlying distribution and therefore cannot evaluate  $R(\hat{f}_i)$ . In fact, the whole model selection problem would go away if had access to the underlying distribution  $P(\underline{x}, y)$ . To classify new instances, we would simply forget about the training set and use the minimum probability of error classifier  $\hat{y}(\underline{x}) = \arg \max_y P(y|\underline{x})$ . No classifier could lead to a lower probability of error (see the appendix). Our task is much more difficult since we have to select  $\hat{f}_i \in \mathcal{F}_i$  as well as the model  $\mathcal{F}_i$  on the basis of the training data alone, without access to  $P$ .

Let's try to understand first intuitively what the model selection criterion has to be able to do. To make this a bit more concrete, consider just choosing between  $\mathcal{F}_1$  and  $\mathcal{F}_2$  corresponding to linear or quadratic feature vectors. Since the models are nested,  $\mathcal{F}_1 \subseteq \mathcal{F}_2$ , we can always achieve lower classification error on the training set by adopting  $\mathcal{F}_2$ . This is regardless of whether the true underlying model is linear. So, by choosing  $\mathcal{F}_2$ , we may be *over-fitting*. If the true relationship between the labels and examples were linear (the minimum probability of error classifier is linear), then the quadratic nature of the resulting decision boundary would simply be due to noise and couldn't generalize very well. So we should be able to see an increasing gap between the training and test errors as a function of the model complexity as in Figure 2 below. Clearly, all things being equal, we should select  $\mathcal{F}_1$  as it is a simpler model. The real question is how to balance the "complexity" of the model, some measure of size or power of  $\mathcal{F}_i$ , against their fit to the training data. There are a number of answers to this question depending on your perspective. We will

briefly go over a few possibilities and return to them later on.

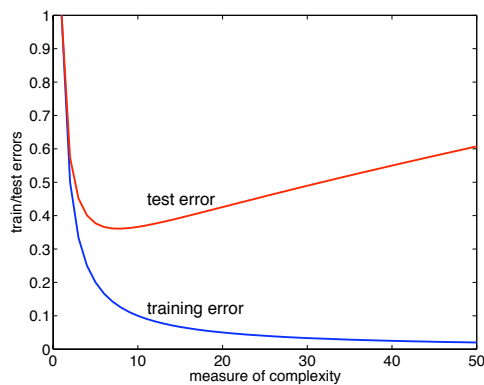


Figure 2: Training and test errors as a function of model order (e.g., degree of polynomial kernel).

### Structural risk minimization

One perspective to model selection is to find the model (set of discriminant functions) that has the best *guarantee of generalization*. To obtain such guarantees we have to relate the *empirical risk*  $R_n(\hat{f}_i)$ , i.e., classification error on the training set,

$$R_n(\hat{f}_i) = \frac{1}{n} \sum_{t=1}^n \text{Loss}_{0-1}(y_t, \hat{f}_i(\underline{x}_t)) \quad (8)$$

to the (expected) risk  $R(\hat{f}_i)$

$$R(\hat{f}_i) = E_{(\underline{x}, y) \sim P} \left\{ \text{Loss}_{0-1}(y, \hat{f}_i(\underline{x})) \right\} \quad (9)$$

that we would like to have access to. In fact, we would like to keep these somewhat close so that the empirical risk (training error) still reflects how well the method will generalize. The empirical risk is computed on the basis of the available training set  $S_n = \{(\underline{x}_1, y_1), \dots, (\underline{x}_n, y_n)\}$  and the loss function  $\text{Loss}_{0-1}(\cdot, \cdot)$  rather than say the hinge loss. For our purposes  $f_i \in \mathcal{F}_i$  could be any estimate derived from the training set that approximately tries to minimizing the empirical risk. In our analysis we will assume that  $\text{Loss}_{0-1}(\cdot, \cdot)$  is the zero-one loss (classification error).

We'd like to quantify how much  $R(\hat{f}_i)$  can deviate from  $R_n(\hat{f}_i)$ . The more powerful our set of classifiers is the more we would expect them to deviate from one another. In other words, the more choices we have in terms of discriminant functions, the less representative the training error of the minimizing classifier is about its generalization error. So, our goal is to show that

$$R(\hat{f}_i) \leq R_n(\hat{f}_i) + \epsilon(n, \mathcal{F}_i, \delta) \quad (10)$$

where the *complexity penalty*  $\epsilon(n, \mathcal{F}_i, \delta)$  depends on the model  $\mathcal{F}_i$ , the number of training instances, and a parameter  $\delta$ . The penalty does *not* depend on the actual training data (otherwise it would be a random variable). We will discuss the parameter  $\delta$  below in more detail. For now, it suffices to say that  $1 - \delta$  specifies the probability that the bound holds. We can only give a probabilistic guarantee in this sense since the empirical risk (training error) is a random quantity that depends on the specific instantiation of the data.

For nested models,  $\mathcal{F}_1 \subseteq \mathcal{F}_2 \subseteq \dots$ , the penalty is necessarily an increasing function of  $i$ , the model order (e.g., the degree of polynomial kernel). Moreover, the penalty should go down as a function  $n$ . In other words, the more data we have, the more complex models we expect to be able to fit and still have the training error close to the generalization error.

The type of result in Eq.(10) gives us an *upper bound guarantee of generalization error*. We can then select the model with the best guarantee, i.e., the one with the lowest bound. Figure 3 shows how we would expect the upper bound to behave as a function of increasingly complex models in our nested "hierarchy" of models.

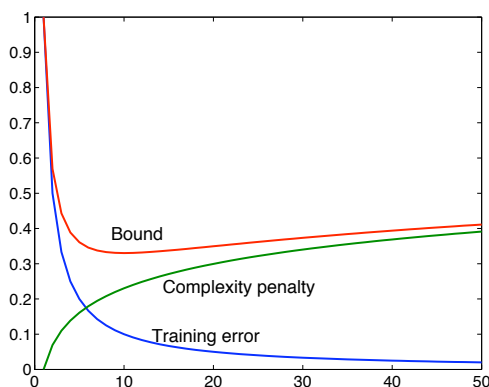


Figure 3: Bound on the generalization error as a function of model order (e.g., degree of polynomial kernel).

Let's derive a result of this type in the simple context where  $\mathcal{F}_i$  only contains a finite number of classifiers  $|\mathcal{F}_i| < \infty$ . We will get to the general theory later on but this simple setting is helpful in understanding how such results come about. To avoid the question of how exactly we estimate  $\hat{f}_i$ , we will require a stronger guarantee: the bound should hold for *all* the classifiers in our set. Specifically, we will fix  $1 - \delta$ , the confidence that the bound holds, and find the smallest  $\epsilon$  such that

$$P\left(\max_{f \in \mathcal{F}_i} |R(f) - R_n(f)| \leq \epsilon\right) \geq 1 - \delta \quad \Leftrightarrow \quad P\left(\max_{f \in \mathcal{F}_i} |R(f) - R_n(f)| > \epsilon\right) \leq \delta \quad (11)$$

We will use the expression on the right to relate the penalty  $\epsilon$  to  $n$ ,  $|\mathcal{F}_i|$ , and  $\delta$ . The expression is the probability that at least one classifier in our set deviates by more than  $\epsilon$  from its training error. The probability is taken over the choice of the training data. Note that  $\delta$  is the probability, over the choice of the training set, that our bound fails. Put another way, if we used  $\epsilon$  to claim that

$$R(f) \leq R_n(f) + \epsilon \quad \text{for all } f \in \mathcal{F}_i \quad (12)$$

then this expression would fail with probability

$$\delta = P\left(\max_{f \in \mathcal{F}_i} |R(f) - R_n(f)| > \epsilon\right) \quad (13)$$

(or holds with probability  $1 - \delta$  over the choice of the training data.) If we fix  $\delta$ , then the smallest  $\epsilon = \epsilon(n, \mathcal{F}_i, \delta)$  that satisfies Eq.(13) is the complexity penalty we are after. Note that since the expression holds for all  $f \in \mathcal{F}_i$  it necessarily also holds for  $\hat{f}_i$ .

In most cases we cannot compute  $\delta$  exactly from Eq.(13) but we can derive an upper bound. This upper bound will lead to a larger than necessary complexity penalty but at least we will get a closed form expression (the utility of the model selection criterion will indeed depend on how tight a bound we can obtain). We will proceed as follows:

$$P\left(\max_{f \in \mathcal{F}_i} |R(f) - R_n(f)| > \epsilon\right) = P(\exists f \in \mathcal{F}_i \text{ s.t. } |R(f) - R_n(f)| > \epsilon) \quad (14)$$

$$\leq \sum_{f \in \mathcal{F}_i} P(|R(f) - R_n(f)| > \epsilon) \quad (15)$$

where we have used the *union bound*  $P(A_1 \cup A_2 \cup \dots) \leq P(A_1) + P(A_2) + \dots$  for any set of events  $A_1, A_2, \dots$  (not necessarily disjoint). In other words, we bound the probability that there are functions in our set with larger than  $\epsilon$  deviation by a sum that each function individually has more than  $\epsilon$  deviation between training and generalization errors.

Now, the individual terms in the sum, i.e.,

$$P(|R(f) - R_n(f)| > \epsilon) \quad (16)$$

deal with a *fixed* discriminant function (it won't change as a function of the training data). We can then associate with each *i.i.d.* training sample  $(x_t, y_t)$ , an indicator  $s_t \in \{0, 1\}$  of whether the sample disagrees with  $f$ :  $s_t = 1$  iff  $y_t f(x_t) \leq 0$ . The empirical error  $R_n(f)$  is therefore just an average of independent random variables (indicators)  $s_t$ :

$$R_n(f) = \frac{1}{n} \sum_{t=1}^n s_t \quad (17)$$

What is the expected value of each  $s_t$  when the expectation is taken over the choice of the training data? It's just  $R(f)$ , the expected risk. So, we can rewrite

$$P(|R(f) - R_n(f)| > \epsilon) \quad (18)$$

as

$$P\left(\left|q - \frac{1}{n} \sum_{t=1}^n s_t\right| > \epsilon\right) \quad (19)$$

where  $q$  equals  $R(f)$  and the probability is now over  $n$  independent binary random variables  $s_1, \dots, s_n$  for which  $P(s_t = 1) = q$ . There are now standard results for evaluating a bound on how much an average of binary random variables deviates from its expectation (Hoeffding's inequality):

$$P\left(\left|q - \frac{1}{n} \sum_{t=1}^n s_t\right| > \epsilon\right) \leq 2 \exp(-2n\epsilon^2) \quad (20)$$

Note that the bound does not depend on  $q$  (or  $R(f)$ ) and therefore not on which  $f$  we chose. Using this result in Eq.(15), gives

$$P\left(\max_{f \in \mathcal{F}_i} |R(f) - R_n(f)| > \epsilon\right) \leq 2|\mathcal{F}_i| \exp(-2n\epsilon^2) = \delta \quad (21)$$

The last equality relates  $\delta$ ,  $|\mathcal{F}_i|$ ,  $n$ , and  $\epsilon$ , as desired. By solving for  $\epsilon$  we get

$$\epsilon = \epsilon(n, \mathcal{F}_i, \delta) = \sqrt{\frac{\log |\mathcal{F}_i| + \log(2/\delta)}{2n}} \quad (22)$$

This is the complexity penalty we were after in this simple case with only a finite number of classifiers in our set.

We have now showed that with probability at least  $1 - \delta$  over the choice of the training set,

$$R(f) \leq R_n(f) + \sqrt{\frac{\log |\mathcal{F}_i| + \log(2/\delta)}{2n}}, \quad \text{uniformly for all } f \in \mathcal{F}_i \quad (23)$$

So, for model selection, we would then estimate  $\hat{f}_i \in \mathcal{F}_i$  for each model, plug the resulting  $\hat{f}_i$  and  $|\mathcal{F}_i|$  on the right hand side of the above equation, and choose the model with the lowest bound (right hand side).  $n$  and  $\delta$  would be the same for all models under consideration.

As an example of another way of using the result, suppose we set  $\delta = 0.05$  and would like any classifier that achieves zero training error to have at most 10% generalization error. Let's solve for the number of training examples we would need for such a guarantee within model  $\mathcal{F}_i$ . We want

$$R(f) \leq 0 + \sqrt{\frac{\log |\mathcal{F}_i| + \log(2/0.05)}{2n}} \leq 0.10 \quad (24)$$

Solving for  $n$  gives

$$n = \frac{\log |\mathcal{F}_i| + \log(2/0.05)}{2(0.10)^2} \quad (25)$$

training examples.

## Appendix: minimum probability of error classifier

Suppose we know the distribution  $P(\underline{x}, y)$  from which training and test examples are sampled from. In this case, we would not need to train any classifier or even look at the training set. We would simply classify each test point according to  $\hat{y}(\underline{x}) = \arg \max_y P(y|\underline{x})$ , i.e., select the most likely label corresponding to  $\underline{x}$ . This classifier has the lowest possible probability of error. Let's see why that is.

Select any classifier  $y(\underline{x}) \in \{-1, 1\}$ . Let  $I(y \neq y(\underline{x})) = 1 - I(y = y(\underline{x}))$  be the indicator function for the case that label  $y$  disagrees with the classifier prediction  $y(\underline{x})$ . Now, to evaluate the probability of error of this classifier, we can sample  $(\underline{x}, y) \sim P$ , check whether

$y$  agrees with  $y(\underline{x})$  and if not, we would add one to the average. More formally,

$$\sum_{\underline{x}, y} P(\underline{x}, y) I(y \neq y(\underline{x})) = \sum_{\underline{x}} P(\underline{x}) \sum_y P(y|\underline{x}) I(y \neq y(\underline{x})) \quad (26)$$

$$= \sum_{\underline{x}} P(\underline{x}) \sum_y P(y|\underline{x}) [1 - I(y = y(\underline{x}))] \quad (27)$$

$$= \sum_{\underline{x}} P(\underline{x}) [\sum_y P(y|\underline{x}) - \sum_y P(y|\underline{x}) I(y = y(\underline{x}))] \quad (28)$$

$$= \sum_{\underline{x}} P(\underline{x}) [1 - P(y(\underline{x})|\underline{x})] \quad (29)$$

$$\geq \sum_{\underline{x}} P(\underline{x}) [1 - \max_y P(y|\underline{x})] \quad (30)$$

$$= \sum_{\underline{x}} P(\underline{x}) [1 - P(\hat{y}(\underline{x})|\underline{x})] \quad (31)$$

where the last equality comes from the fact that  $\hat{y}(\underline{x}) = \arg \max_y P(y|\underline{x})$  selects a label that attains the maximum value  $\max_y P(y|\underline{x}) = P(\hat{y}(\underline{x})|\underline{x})$ .