Lecture topics:

- Kernel optimization
- Model (kernel) selection

Kernel optimization

Whether we are interested in (linear) classification or regression we are faced with the problem of selecting an appropriate kernel function. A step in this direction might be to tailor a particular kernel a bit better to the available data. We could, for example, introduce additional parameters in the kernel and optimize those parameters so as to improve the performance. These parameters could be simple as the β parameter in the radial basis kernel, weight each dimension of the input vectors, or more flexible as finding the best convex combination of basic (fixed) kernels. Key to such an approach is the measure we would optimize. Ideally, this measure would be the generalization error but we obviously have to settle for a surrogate measure. The surrogate measure could be cross-validation or an alternative criterion related to the generalization error such as the geometric margin.

We need additional safeguards if we are to use the geometric margin. For example, simply multiplying the feature vectors by two would double the geometric margin. So, without normalization, the margin cannot serve as an appropriate criterion. The simplest way to normalize the feature vectors prior to estimation would be to require that $\|\phi(\mathbf{x})\| = 1$ for all $\mathbf{x}$ regardless of the kernel. This normalization can be done directly in the kernel representation as follows

$$\tilde{K}(\mathbf{x}, \mathbf{x}') = \frac{K(\mathbf{x}, \mathbf{x}')}{\sqrt{K(\mathbf{x}, \mathbf{x}) K(\mathbf{x}', \mathbf{x}')}} \tag{1}$$

Another approach to optimizing the kernel function is *kernel alignment*. In other words, we would adjust the kernel parameters so as to make it, or its Gram matrix, more towards an ideal target kernel. For example, in a classification setting, we could use

$$K^*_{ij} = y_i y_j \tag{2}$$

as the Gram matrix of the target kernel. One argument for selecting this as the target is that if we set $\alpha_j = 1/n$ then

$$\sum_{j=1}^{n} \alpha_j y_j K^*_{ij} = y_i \tag{3}$$

and all the training examples are classified correctly with the same margin. (You could argue that another target should be used instead). Let's see how we can align the kernel towards this target. Suppose our parameterized kernel is a convex combination of kernels (e.g., constructed on the basis of different sources of input data)

$$K(\mathbf{x}, \mathbf{x}'; \theta) = \sum_{i=1}^{m} \theta_i K_i(\mathbf{x}, \mathbf{x}') \tag{4}$$

where $\theta_i \geq 0$ and $\sum_{i=1}^{m} \theta_i = 1$. These are the parameters we can adjust. We can now set θ so as to make the Gram matrix of this kernel, $K_{ij}(\theta)$, more similar to the Gram matrix of the target kernel, K^*_{ij}. To do this we view the Gram matrices as vectors and define their inner product in the usual way

$$\langle K^*, K_\theta \rangle = \sum_{i,j=1}^{n} K^*_{ij} K_{ij}(\theta) \tag{5}$$

The parameters θ can be now set so as to maximize the cosine of the angle between the Gram matrices:

$$\frac{\langle K^*, K_\theta \rangle}{\sqrt{\langle K^*, K^* \rangle \langle K_\theta, K_\theta \rangle}} \tag{6}$$

Model (kernel) selection

Optimizing the kernel in a parameterized form involved little consideration for the complexity of the set of classifiers we are fitting to finite data. It therefore did not address the problem of *over-fitting* or fitting too complex a model to too few data points. In many cases it makes sense to explicitly 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[1] (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

$$K_1(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^T \mathbf{x}') \text{ or} \tag{7}$$

$$K_2(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^T \mathbf{x}')^2 \tag{8}$$

[1] In statistics, a model is a family/set of distributions or a family/set of linear separators.

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, it therefore 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

$$f_1(\mathbf{x}; \theta, \theta_0) \quad = \quad \theta^T \phi^{(1)}(\mathbf{x}) + \theta_0 \tag{9}$$

where $\phi^{(1)}(\mathbf{x})$ is the feature representation corresponding to K_1 such that $K_1(\mathbf{x}, \mathbf{x}') = \phi^{(1)}(\mathbf{x})^T \phi^{(1)}(\mathbf{x}')$. By varying the parameters θ and θ_0 we can generate the set of possible discriminant functions corresponding to this kernel:

$$\mathcal{F}_1 = \{ f_1(\cdot; \theta, \theta_0) : \theta \in \mathcal{R}^{d_1}, \theta_0 \in \mathcal{R} \} \tag{10}$$

$\mathcal{F}_2$ 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 $K_1, K_2, \ldots$, where the models associated with the kernels are nested $\mathcal{F}_1 \subseteq \mathcal{F}_2 \subseteq \ldots$. 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 $S_n = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ 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

$$J(\theta, \theta_0) = \sum_{t=1}^{n} \text{Loss}\Big(y_t, f(\mathbf{x}_t; \theta, \theta_0) \Big) + \lambda_n \|\theta\|^2 \tag{11}$$

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

Each parameter setting (θ, θ_0), i.e., each discriminant function in our set, has an associated expected loss or *risk*

$$R(\theta, \theta_0) = E_{(\mathbf{x},y)\sim P}\left\{ \text{Loss}^*\Big(y, f(\mathbf{x}; \theta, \theta_0)\Big) \right\} \tag{12}$$

where the new test example and label, $(\mathbf{x}, y)$, 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 $\text{Loss}^*(\cdot, \cdot)$ 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 $\text{Loss}^*(y, f(\mathbf{x})) = 1 - \delta(y, \text{sign}(f(\mathbf{x})))$, i.e., the 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

$$R(\hat{f}_i) = E_{(\mathbf{x},y)\sim P}\Big\{ \text{Loss}^*\Big(y, \hat{f}_i(\mathbf{x})\Big) \Big\} \tag{13}$$

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 $P(\mathbf{x}, y)$. To classify new instances, we would simply forget about the training set and use the minimum probability of error classifier $\hat{y}(\mathbf{x}) = \arg\max_y P(y|\mathbf{x})$ (see the appendix). No classifier could lead to a lower probability of error. 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 1 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.

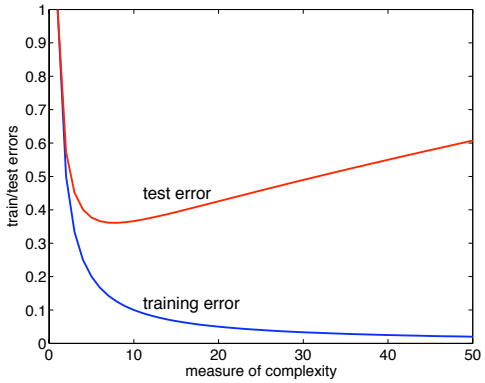

Figure 1: Training and test errors as a function of model order (e.g., degree of polynomial kernel).

Model selection criteria: structural risk minimization

One approach to model selection is to try to directly relate the (expected) risk $R(\hat{f}_i)$

$$R(\hat{f}_i) \quad = \quad E_{(\mathbf{x},y)\sim P}\Big\{ \text{Loss}^*\Big(y, \hat{f}_i(\mathbf{x})\Big) \Big\} \tag{14}$$

that we would like to have and the *empirical risk* $R_n(\hat{f}_i)$

$$R_n(\hat{f}_i) \;=\; \frac{1}{n}\sum_{t=1}^{n} \mathrm{Loss}^*\Big(y_t, \hat{f}_i(\mathbf{x}_t)\Big) \tag{15}$$

that we can compute. If we can do this, then we have a partial access to $R(\hat{f}_i)$ through its empirical counterpart $R_n(\hat{f}_i)$. Note that the empirical risk here is computed on the basis of the available training set $S_n = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ and $\mathrm{Loss}^*(\cdot,\cdot)$ rather than say the hinge loss. For our purposes here, $\hat{f}_i \in \mathcal{F}_i$ could be any estimate derived from the training set that approximately tries to minimizing the empirical risk.

We are interested in quantifying how much $R(\hat{f}_i)$ can deviate from $R_n(\hat{f}_i)$. The larger the deviation the less representative the training error is about the generalization error. This happens with more complex models $\mathcal{F}_i$. Indeed, we aim to show that

$$R(\hat{f}_i) \leq R_n(\hat{f}_i) + C(n, \mathcal{F}_i, \delta) \tag{16}$$

where the *complexity penalty* $C(n, \mathcal{F}_i)$ only depends on the model $\mathcal{F}_i$, the number of training instances, and a parameter δ. The peanalty does *not* depend on the actual training data. We will discuss the parameter δ 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, $\mathcal{F}_1 \subseteq \mathcal{F}_2 \subseteq \ldots$, 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.(16) 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 2 shows how we would expect the upper bound to behave as a function of increasingly complex models in our nested "hierarchy" of models.

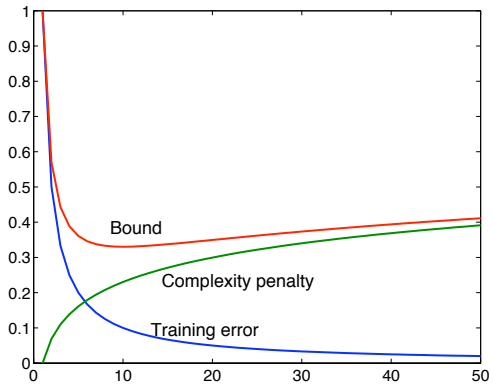

Figure 2: Bound on the generalization error as a function of model order (e.g., degree of polynomial kernel).