Lecture topics:

- Kernel form of linear regression
- Kernels, examples, construction, properties

Linear regression and kernels

Consider a slightly simpler model where we omit the offset parameter θ_0, reducing the model to $y = \theta^T \phi(\mathbf{x}) + \epsilon$ where $\phi(\mathbf{x})$ is a particular feature expansion (e.g., polynomial). Our goal here is to turn both the estimation problem and the subsequent prediction task into forms that involve only inner products between the feature vectors.

We have already emphasized that regularization is necessary in conjunction with mapping examples to higher dimensional feature vectors. The regularized least squares objective to be minimized, with parameter λ, is given by

$$J(\theta) = \sum_{t=1}^{n} \big(y_t - \theta^T \phi(\mathbf{x}_t)\big)^2 + \lambda \|\theta\|^2 \tag{1}$$

This form can be derived from penalized log-likelihood estimation (see previous lecture notes). The effect of the regularization penalty is to pull all the parameters towards zero. So any linear dimensions in the parameters that the training feature vectors do not pertain to are set explicitly to zero. We would therefore expect the optimal parameters to lie in the span of the feature vectors corresponding to the training examples. This is indeed the case.

As before, the optimality condition for θ follows from setting the gradient to zero:

$$\frac{dJ(\theta)}{d\theta} \;=\; -2\sum_{t=1}^{n} \overbrace{\big(y_t - \theta^T \phi(\mathbf{x}_t)\big)}^{\alpha_t} \phi(\mathbf{x}_t) + 2\lambda\theta = 0 \tag{2}$$

We can therefore construct the optimal θ in terms of prediction differences α_t and the feature vectors:

$$\theta = \frac{1}{\lambda} \sum_{t=1}^{n} \alpha_t \phi(\mathbf{x}_t) \tag{3}$$

The implication is that the optimal θ (however high dimensional) will lie in the span of the feature vectors corresponding to the training examples. This is due to the regularization penalty we added. But how do we set α_t? The values for α_t can be found by insisting that they indeed can be interpreted as prediction differences:

$$\alpha_t = y_t - \theta^T\phi(\mathbf{x}_t) = y_t - \frac{1}{\lambda}\sum_{t'=1}^{n}\alpha_{t'}\phi(\mathbf{x}_{t'})^T\phi(\mathbf{x}_t) \tag{4}$$

Thus α_t depends only on the actual responses y_t and the inner products between the training examples, the *Gram matrix*:

$$\mathbf{K} \;=\; \begin{bmatrix} \phi(\mathbf{x}_1)^T\phi(\mathbf{x}_1) & \cdots & \phi(\mathbf{x}_1)^T\phi(\mathbf{x}_n) \\ \cdots & \cdots & \cdots \\ \phi(\mathbf{x}_n)^T\phi(\mathbf{x}_1) & \ldots & \phi(\mathbf{x}_n)^T\phi(\mathbf{x}_n) \end{bmatrix} \tag{5}$$

In a vector form,

$$\begin{aligned} \mathbf{a} &= [\alpha_1, \ldots, \alpha_n]^T, & (6)\\ \mathbf{y} &= [y_1, \ldots, y_n]^T, & (7)\\ \mathbf{a} &= \mathbf{y} - \frac{1}{\lambda}\mathbf{K}\mathbf{a} & (8) \end{aligned}$$

the solution is

$$\hat{\mathbf{a}} = \lambda\Big(\lambda\mathbf{I} + \mathbf{K}\Big)^{-1}\mathbf{y} \tag{9}$$

Note that finding the estimates $\hat{\alpha}_t$ requires inverting a $n \times n$ matrix. This is the cost of dealing with inner products as opposed to handing feature vectors directly. In some cases, the benefit is substantial since the feature vectors in the inner products may be infinite dimensional but never needed explicitly.

As a result of finding $\hat{\alpha}_t$ we can cast the predictions for new examples also in terms of inner products:

$$y = \hat{\theta}^T\phi(\mathbf{x}) = \sum_{t=1}^{n}(\hat{\alpha}_t/\lambda)\phi(\mathbf{x}_{t'})^T\phi(\mathbf{x}) = \sum_{t=1}^{n}\hat{\alpha}_t K(\mathbf{x}_{t'}, \mathbf{x}) \tag{10}$$

where we view $K(\mathbf{x}_{t'}, \mathbf{x})$ as a *kernel function*, a function of two arguments $\mathbf{x}_{t'}$ and $\mathbf{x}$.

Kernels

So we have now successfully turned a regularized linear regression problem into a kernel form. This means that we can simply substitute different kernel functions $K(\mathbf{x}, \mathbf{x}')$ into the estimation/prediction equations. This gives us an easy access to a wide range of possible regression functions. Here are a couple of standard examples of kernels:

- *Polynomial kernel*

$$K(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^T \mathbf{x}')^p, \ \ p = 1, 2, \ldots \tag{11}$$

- *Radial basis kernel*

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\beta}{2} \|\mathbf{x} - \mathbf{x}'\|^2 \right), \ \ \beta > 0 \tag{12}$$

We have already discussed the feature vectors corresponding to the polynomial kernel. The components of these feature vectors were polynomial terms up to degree p with specifically chosen coefficients. The restricted choice of coefficients was necessary in order to collapse the inner product calculations.

The feature "vectors" corresponding to the radial basis kernel are infinite dimensional! The components of these "vectors" are indexed by $\mathbf{z} \in \mathcal{R}^d$ where d is the dimension of the original input $\mathbf{x}$. More precisely, the feature vectors are functions:

$$\phi_{\mathbf{z}}(\mathbf{x}) = c(\beta, d)\, N(\mathbf{z};\, \mathbf{x},\, 1/2\beta) \tag{13}$$

where $N(\mathbf{z}; \mathbf{x}, (1/\beta))$ is a normal pdf over $\mathbf{z}$ and $c(\beta, d)$ is a constant. Roughly speaking, the radial basis kernel measures the probability that you would get the same sample $\mathbf{z}$ (in the same small region) from two normal distributions with means $\mathbf{x}$ and $\mathbf{x}'$ and a common variance $1/2\beta$. This is a reasonable measure of "similarity" between $\mathbf{x}$ and $\mathbf{x}'$ and kernels are often defined from this perspective. The inner product giving rise to the radial basis kernel is defined through integration

$$K(\mathbf{x}, \mathbf{x}') = \int \phi_{\mathbf{z}}(\mathbf{x}) \phi_{\mathbf{z}}(\mathbf{x}') d\mathbf{z} \tag{14}$$

We can also construct various types of kernels from simpler ones. Here are a few rules to guide us. Assume $K_1(\mathbf{x}, \mathbf{x}')$ and $K_2(\mathbf{x}, \mathbf{x}')$ are valid kernels (correspond to inner products of some feature vectors), then

1. $K(\mathbf{x}, \mathbf{x}') = f(\mathbf{x}) K_1(\mathbf{x}, \mathbf{x}') f(\mathbf{x}')$ for any function $f(\mathbf{x})$,
2. $K(\mathbf{x}, \mathbf{x}') = K_1(\mathbf{x}, \mathbf{x}') + K_2(\mathbf{x}, \mathbf{x}')$,
3. $K(\mathbf{x}, \mathbf{x}') = K_1(\mathbf{x}, \mathbf{x}') K_2(\mathbf{x}, \mathbf{x}')$

are all valid kernels. While simple, these rules are quite powerful. Let's first understand these rules from the point of view of the implicit feature vectors. For each rule, let $\phi(\mathbf{x})$ be the feature vector corresponding to K and $\phi^{(1)}(\mathbf{x})$ and $\phi^{(2)}(\mathbf{x})$ the feature vectors associated with K_1 and K_2, respectively. The feature mapping for the first rule is given simply by multiplying with the scalar function $f(\mathbf{x})$:

$$\phi(\mathbf{x}) = f(\mathbf{x})\phi^{(1)}(\mathbf{x}) \tag{15}$$

so that $\phi(\mathbf{x})^T\phi(\mathbf{x}') = f(\mathbf{x})\phi^{(1)}(\mathbf{x})^T\phi^{(1)}(\mathbf{x}')f(\mathbf{x}') = f(\mathbf{x})K_1(\mathbf{x},\mathbf{x}')f(\mathbf{x}')$. The second rule, adding kernels, corresponds to just concatenating the feature vectors

$$\phi(\mathbf{x}) = \begin{bmatrix} \phi^{(1)}(\mathbf{x}) \\ \phi^{(2)}(\mathbf{x}) \end{bmatrix} \tag{16}$$

The third and the last rule is a little more complicated but not much. Suppose we use a double index i, j to index the components of $\phi(\mathbf{x})$ where i ranges over the components of $\phi^{(1)}(\mathbf{x})$ and j refers to the components of $\phi^{(2)}(\mathbf{x})$. Then

$$\phi_{i,j}(\mathbf{x}) = \phi_i^{(1)}(\mathbf{x})\phi_j^{(2)}(\mathbf{x}) \tag{17}$$

It is now easy to see that

$$\begin{aligned}
K(\mathbf{x},\mathbf{x}') &= \phi(\mathbf{x})^T\phi(\mathbf{x}') && (18)\\
&= \sum_{i,j}\phi_{i,j}(\mathbf{x})\phi_{i,j}(\mathbf{x}') && (19)\\
&= \sum_{i,j}\phi_i^{(1)}(\mathbf{x})\phi_j^{(2)}(\mathbf{x})\phi_i^{(1)}(\mathbf{x}')\phi_j^{(2)}(\mathbf{x}') && (20)\\
&= [\sum_i \phi_i^{(1)}(\mathbf{x})\phi_i^{(1)}(\mathbf{x}')][\sum_j \phi_j^{(2)}(\mathbf{x})\phi_j^{(2)}(\mathbf{x}')] && (21)\\
&= [\phi^{(1)}(\mathbf{x})^T\phi^{(1)}(\mathbf{x}')][\phi^{(2)}(\mathbf{x})^T\phi^{(2)}(\mathbf{x}')] && (22)\\
&= K_1(\mathbf{x},\mathbf{x}')K_2(\mathbf{x},\mathbf{x}') && (23)
\end{aligned}$$

These construction rules can also be used to verify that something is a valid kernel. As an example, let's figure out why a radial basis kernel

$$K(\mathbf{x},\mathbf{x}') = \exp\{-\frac{1}{2}\|\mathbf{x}-\mathbf{x}'\|^2\} \tag{24}$$

is a valid kernel.

$$\begin{aligned}\exp\{-\frac{1}{2}\|\mathbf{x}-\mathbf{x}'\|^2\} &= \exp\{-\frac{1}{2}\mathbf{x}^T\mathbf{x}+\mathbf{x}^T\mathbf{x}'-\frac{1}{2}\mathbf{x}'^T\mathbf{x}'\} &(25)\\ &= \overbrace{\exp\{-\frac{1}{2}\mathbf{x}^T\mathbf{x}\}}^{f(\mathbf{x})}\cdot\exp\{\mathbf{x}^T\mathbf{x}'\}\cdot\overbrace{\exp\{-\frac{1}{2}\mathbf{x}'^T\mathbf{x}'\}}^{f(\mathbf{x}')} &(26)\end{aligned}$$

Here $\exp\{\mathbf{x}^T\mathbf{x}'\}$ is a sum of simple products $\mathbf{x}^T\mathbf{x}'$ and is therefore a kernel based on the second and third rules; the first rule allows us to incorporate $f(\mathbf{x})$ and $f(\mathbf{x}')$.

String kernels. It is often necessary to make predictions (classify, assess risk, determine user ratings) on the basis of more complex objects such as variable length sequences or graphs that do not necessarily permit a simple description as points in $\mathcal{R}^d$. The idea of kernels extends to such objects as well. Consider, for example, the case where the inputs $\mathbf{x}$ are variable length sequences (e.g., documents or biosequences) with elements from some common alphabet $\mathcal{A}$ (e.g., letters or protein residues). One way to compare such sequences is to consider subsequences that they may share. Let $\mathbf{u} \in \mathcal{A}^k$ denote a length k sequence from this alphabet and $\mathbf{i}$ a sequence of k indexes. So, for example, we can say that $\mathbf{u} = \mathbf{x}[\mathbf{i}]$ if $u_1 = x_{i_1}$, $u_2 = x_{i_2}$, $\ldots$, $u_k = x_{i_k}$. In other words, $\mathbf{x}$ contains the elements of $\mathbf{u}$ in positions $i_1 < i_2 < \cdots < i_k$. If the elements of $\mathbf{u}$ are found in successive positions in $\mathbf{x}$, then $i_k - i_1 = k - 1$. A simple string kernel corresponds to feature vectors with counts of occurences of length k subsequences:

$$\phi_{\mathbf{u}}(\mathbf{x}) = \sum_{\mathbf{i}:\mathbf{u}=\mathbf{x}[\mathbf{i}]} \delta(i_k - i_1, k-1) \tag{27}$$

In other words, the components are indexed by subsequences $\mathbf{u}$ and the value of $\mathbf{u}$-component is the number of times $\mathbf{x}$ contains $\mathbf{u}$ as a contiguous subsequence. For example,

$$\phi_{\texttt{on}}(\texttt{the common construct}) = 2 \tag{28}$$

The number of components in such feature vectors is very large (exponential in k). Yet, the inner product

$$\sum_{\mathbf{u}\in\mathcal{A}^k} \phi_{\mathbf{u}}(\mathbf{x})\phi_{\mathbf{u}}(\mathbf{x}') \tag{29}$$

can be computed efficiently (there are only a limited number of possible contiguous subsequences in $\mathbf{x}$ and $\mathbf{x}'$). The reason for this difference, and the argument in favor of kernels more generally, is that the feature vectors have to aggregate the information necessary to compare any two sequences while the inner product is evaluated for two *specific* sequences.

We can also relax the requirement that matches must be contiguous. To this end, we define the length of the window of $\mathbf{x}$ where $\mathbf{u}$ appears as $l(\mathbf{i}) = i_k - i_1$. The feature vectors in a *weighted gapped substring kernel* are given by

$$\phi_{\mathbf{u}}(\mathbf{x}) = \sum_{\mathbf{i}:\mathbf{u}=\mathbf{x}[\mathbf{i}]} \lambda^{l(\mathbf{i})} \tag{30}$$

where the parameter $\lambda \in (0, 1)$ specifies the penalty for non-contiguous matches to $\mathbf{u}$. The resulting kernel

$$K(\mathbf{x}, \mathbf{x}') = \sum_{\mathbf{u}\in\mathcal{A}^k} \phi_{\mathbf{u}}(\mathbf{x})\phi_{\mathbf{u}}(\mathbf{x}') = \sum_{\mathbf{u}\in\mathcal{A}^k} \left(\sum_{\mathbf{i}:\mathbf{u}=\mathbf{x}[\mathbf{i}]} \lambda^{l(\mathbf{i})} \right) \left(\sum_{\mathbf{i}:\mathbf{u}=\mathbf{x}'[\mathbf{i}]} \lambda^{l(\mathbf{i})} \right) \tag{31}$$

can be computed recursively. It is often useful to normalize such a kernel so as to remove any immediate effect from the sequence length:

$$\tilde{K}(\mathbf{x}, \mathbf{x}') = \frac{K(\mathbf{x}, \mathbf{x}')}{\sqrt{K(\mathbf{x}, \mathbf{x})}\sqrt{K(\mathbf{x}', \mathbf{x}')}} \tag{32}$$

Appendix (optional): Kernel linear regression with offset

Given a feature expansion specified by $\phi(\mathbf{x})$ we try to minimize

$$J(\theta, \theta_0) = \sum_{t=1}^{n} \left(y_t - \theta^T\phi(\mathbf{x}_t) - \theta_0 \right)^2 + \lambda\|\theta\|^2 \tag{33}$$

where we have chosen *not* to regularize θ_0 to preserve the similarity to classification discussed later on. Not regularizing θ_0 means, e.g., that we do not care whether all the responses have a constant added to them; the value of the objective, after optimizing θ_0, would remain the same with or without such constant.

Setting the derivatives with respect to θ_0 and θ to zero gives the following optimality conditions:

$$\frac{dJ(\theta, \theta_0)}{d\theta_0} = -2\sum_{t=1}^{n} \left(y_t - \theta^T\phi(\mathbf{x}_t) - \theta_0 \right) = 0 \tag{34}$$

$$\frac{dJ(\theta, \theta_0)}{d\theta} = 2\lambda\theta - 2\sum_{t=1}^{n} \overbrace{\left(y_t - \theta^T\phi(\mathbf{x}_t) - \theta_0 \right)}^{\alpha_t} \phi(\mathbf{x}_t) = 0 \tag{35}$$

We can therefore construct the optimal θ in terms of prediction differences α_t and the feature vectors as before:

$$\theta = \frac{1}{\lambda}\sum_{t=1}^{n} \alpha_t \phi(\mathbf{x}_t) \tag{36}$$

Using this form of the solution for θ and Eq.(34) we can also express the optimal θ_0 as a function of the prediction differences α_t:

$$\theta_0 = \frac{1}{n}\sum_{t=1}^{n}\big(y_t - \theta^T\phi(\mathbf{x}_t)\big) = \frac{1}{n}\sum_{t=1}^{n}\left(y_t - \frac{1}{\lambda}\sum_{t'=1}^{n}\alpha_{t'}\phi(\mathbf{x}_{t'})^T\phi(\mathbf{x}_t)\right) \tag{37}$$

We can now constrain α_t to take on values that can indeed be interpreted as prediction differences:

$$\begin{aligned}
\alpha_i &= y_i - \theta^T\phi(\mathbf{x}_i) - \theta_0 & (38)\\
&= y_i - \frac{1}{\lambda}\sum_{t'=1}^{n}\alpha_{t'}\phi(\mathbf{x}_{t'})^T\phi(\mathbf{x}_i) - \theta_0 & (39)\\
&= y_i - \frac{1}{\lambda}\sum_{t'=1}^{n}\alpha_{t'}\phi(\mathbf{x}_{t'})^T\phi(\mathbf{x}_i) - \frac{1}{n}\sum_{t=1}^{n}\left(y_t - \frac{1}{\lambda}\sum_{t'=1}^{n}\alpha_{t'}\phi(\mathbf{x}_{t'})^T\phi(\mathbf{x}_t)\right) & (40)\\
&= y_i - \frac{1}{n}\sum_{t=1}^{n}y_t - \frac{1}{\lambda}\sum_{t'=1}^{n}\alpha_{t'}\left(\phi(\mathbf{x}_{t'})^T\phi(\mathbf{x}_i) - \frac{1}{n}\sum_{t=1}^{n}\phi(\mathbf{x}_{t'})^T\phi(\mathbf{x}_t)\right) & (41)
\end{aligned}$$

With the same matrix notation as before, and letting $\mathbf{1} = [1, \ldots, 1]^T$, we can rewrite the above condition as

$$\mathbf{a} = \overbrace{(\mathbf{I} - \mathbf{1}\mathbf{1}^T/n)}^{C}\,\mathbf{y} - \frac{1}{\lambda}(\mathbf{I} - \mathbf{1}\mathbf{1}^T/n)\mathbf{K}\mathbf{a} \tag{42}$$

where $C = \mathbf{I} - \mathbf{1}\mathbf{1}^T/n$ is a *centering* matrix. Any solution to the above equation has to satisfy $\mathbf{1}^T\mathbf{a} = 0$ (just left multiply the equation with $\mathbf{1}^T$). Note that this is exactly the optimality condition for θ_0 in Eq.(34). Using this "summing to zero" property of the solution we can rewrite the above equation as

$$\mathbf{a} = C\mathbf{y} - \frac{1}{\lambda}C\mathbf{K}C\mathbf{a} \tag{43}$$

where we have introduced an additional centering operation on the right hand side. This cannot change the solution since $C\mathbf{a} = \mathbf{a}$ whenever $\mathbf{1}^T\mathbf{a} = 0$. The solution $\hat{\mathbf{a}}$ is then

$$\hat{\mathbf{a}} = \lambda \left(\lambda \mathbf{I} + C\mathbf{K}C\right)^{-1} C\mathbf{y} \tag{44}$$

Once we have $\hat{\mathbf{a}}$ we can reconstruct $\hat{\theta}_0$ from Eq.(37). $\hat{\theta}^T\phi(\mathbf{x})$ reduces to the kernel form as before.