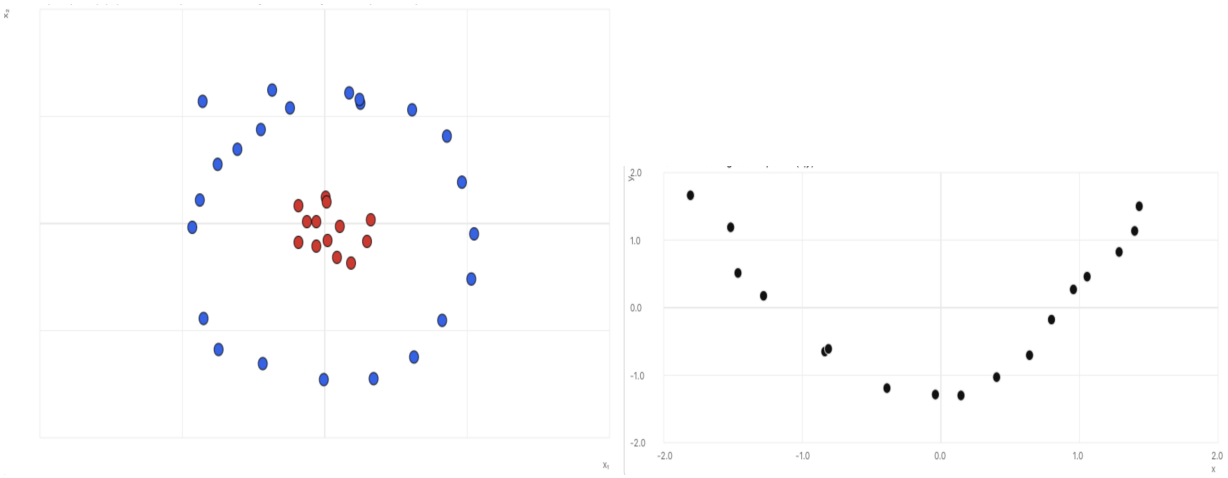
Kernel Method (CS3780/5780, Sp26)

1 Linear Methods and Feature Space

In class we have considered a large number of linear methods. Perceptron, Logistic regression, SVM are all linear classification algorithms and ordinary least squares and ridge regression are linear regression methods. In fact, most of these methods can be written as solution to optimization problems of the form:

$$\mathbf{w}, b = \underset{\mathbf{w}\in\mathbb{R}^d, b\in\mathbb{R}}{\arg\min} \sum_{i=1}^{n} \ell(\mathbf{w}^\top \mathbf{x}_i + b, y_i) + \lambda \mathbf{w}^\top \mathbf{w} \tag{1}$$

with different choices of loss function ℓ and λ. However, all these methods only produce a linear relationship between input and output variables. Like linear classification boundary in classification problems and prediction for y as a linear function of x in regression problems. However, linear methods can only be so expressive. Consider the following classification and regression problems:



(a) Classification (b) regression

Figure 1: Need Non-linearity

Tho effectively solve these problems well we need to consider methods that are not necessarily only linear in inout vectors $\mathbf{x}$.

An elegant idea that lets one port all the nice linear methods we have worked hard to build to obtain more richer methods is the so called kernel method or kernel trick. We will put together the kernel magic in a few steps. But let us start first with the idea of a feature map. Given an input vector $\mathbf{x} \in \mathbb{R}^d$ the feature map $\phi(x)$ is a mapping that maps vector x to another vector $\phi(x)$ which is typically much higher dimensional vector in the so called feature space. In these notes we

will eventually see that this feature space can technically even be infinite dimensional. But first let us consider the regression problem shown in the second figure above. Clearly a linear regression solution will have a large error since no line can approximate the shown curve well. But consider the following feature map:

$$\phi(x) = \begin{bmatrix} x \\ x^2 \end{bmatrix}$$

Now if one considers linear regression not on the one dimensional data x but on the two dimensional input feature $\phi(x)$ then we can see that the resulting solution is a pretty good fit. See the figure below.

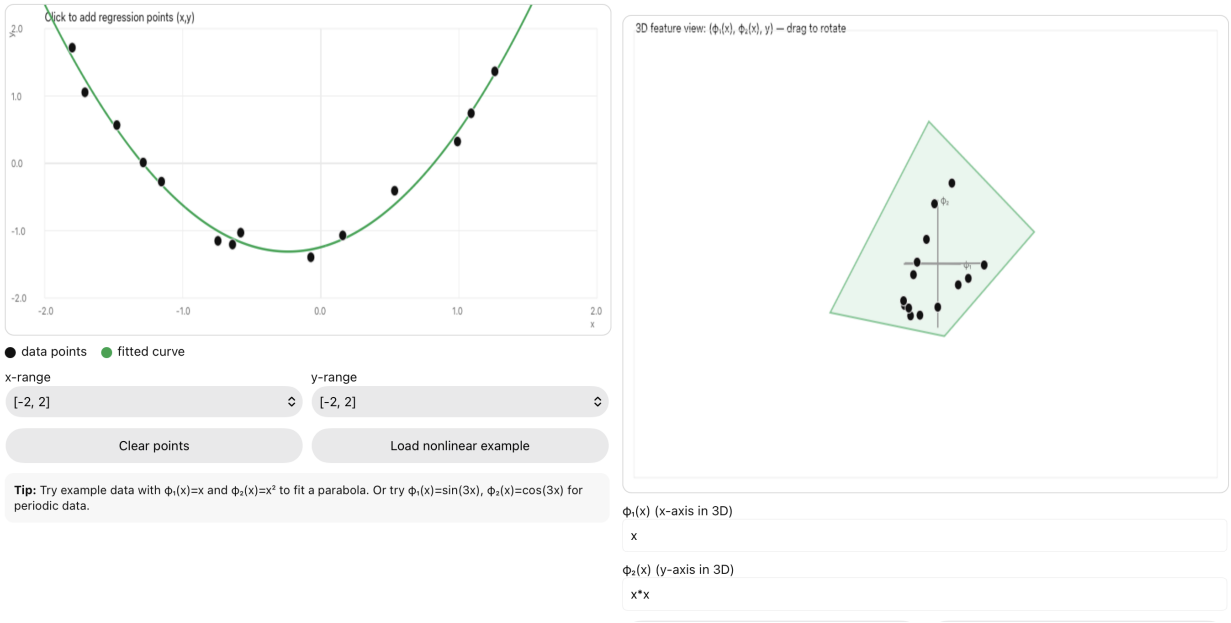


Figure 2: Linear in feature space is non-linear in input space

Magic 1: Linear in $\phi(x)$ leads to non-linearity in x space.

This means that the more expressive feature mapping $\phi(x)$ we take, the more complex functions of x we can express.

Quiz 1: Suggest a feature mapping for the classification example above so that once we apply linear method on feature space, like SVM for instance, we get good classification error on the example.

What was your suggestion above? For the above example we could visually see the example, but in general, we cant have a very accurate guess. So one option is to try and enumerate as many features as we can. For instance, if $\mathbf{x} = [\mathbf{x}[1], \mathbf{x}[2]]^\top$ and we wanted to obtain a method that can capture all functions of $\mathbf{x}$ that are polynomials of up to degree 2, we could use:

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

Quiz 2: If $\mathbf{x}$ were d dimensional, and:

1. we wanted to capture all functions of $\mathbf{x}$ that are polynomials of up to degree 2, how many dimensions would $\phi(\mathbf{x})$ be?
2. we wanted to capture all functions of $\mathbf{x}$ that are polynomials of up to degree p, how many dimensions would $\phi(\mathbf{x})$ be?

The answer above indicates a problem that fairly quickly, we will have too many features (ie. $\phi(\mathbf{x})$ is very large dimensional) to enumerate to get even a modest degree polynomial of $\mathbf{x}$. This is where the second part of the kernel magic comes in. The magic is that we will never need to explicitly enumerate $\phi(\mathbf{x})$.

Magic 2: Never explicitly enumerate $\phi(x)$, only ever evaluate $k(x, x') = \phi(x)^\top \phi(x')$.

To see this magic, will take us a couple steps. First, we make the following claim.
Claim: w that is a solution to the optimization problem in Eq. 1 takes the form:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i$$

We will see why the above claim is true in a bit, but before we do that, let us see the implications of the claim and how we can use it to obtain the kernel magic. To this end, note that if the above were true, then if one replaces $\mathbf{x}_i$'s instead by feature mappings $\phi(\mathbf{x}_i)$'s then we have that the solution for optimization problem takes the form $\mathbf{w} = \sum_{i=1}^{n} \alpha_i \phi(\mathbf{x}_i)$. In this case, note that if for a new point $\mathbf{x}$ we wanted to predict $\mathbf{w}^\top \phi(\mathbf{x}) + b$, this would take the form:

$$h(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}) + b = \sum_{i=1}^{n} \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}) + b$$

Now as we already saw, ϕ can be very large dimensional (possibly even infinite). But say we had a function, the kernel function such that for any $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$,

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top \phi(\mathbf{x}')$$

While enumerating $\phi(\mathbf{x})$ might be computationally prohibitive or even impossible (if feature space in infinite dimensional), it could still be the case that kernel function $k(\mathbf{x}, \mathbf{x}')$ which is the inner product in the feature space can be computed efficiently.

Quiz 3: What is $\phi(\mathbf{x})$ for the following kernel function: $k(x, x') = (1 + x \cdot x')^p$?

Given the claim, if we consider the hypothesis in the feature space, we could make our prediction easily as:

$$h(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}) + b = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b$$

Note that inner products measure similarity between points and so kernel functions which are inner products in feature space measure similarity between the arguments. From this perspective, if we consider the prediction for a new $\mathbf{x}$ as $h(\mathbf{x}) = \sum_{i=1}^{n} k(\mathbf{x}_i, \mathbf{x}) + b$ then for a second ignoring the bias term b, the prediction can be seen as a weighted sum of similarities between each training point to the test point. From this perspective if we think of the kernel function $k(\mathbf{x}, \mathbf{x}')$ a being 1 on closest point and 0 on other points, then the resulting classifier is a one nearest neighbor algorithm. Here are some other popular kernels:

- Linear kernel: $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$
- Polynomial kernel: $k(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^\top \mathbf{x}')^p$
- Radial Basis Fucntion (RBF) kernel: $k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x} - \mathbf{x}'\|^2\right)$

Why the claim is true: We will show this by induction. Say the optimization problem in Eq. 1 can be solved using gradeint descent. So say $\mathbf{w}_0 = 0$ and we perform gradient descent steps till convergence to the minima. Note that by gradient descent update:

$$\mathbf{w}_1 = \mathbf{w}_0 - \eta \sum_{i=1}^{n} \ell'(\mathbf{w}_0^\top \mathbf{x}_i, y_i)\mathbf{x}_i - 2\eta\lambda\mathbf{w}_0 = -\sum_{i=1}^{n} \alpha_i^1 \mathbf{x}_i$$

where $\alpha_i^1 = \eta\ell'(\mathbf{w}_0^\top \mathbf{x}_i, y_i)$. Hence, $\mathbf{w}_1 = \sum_{i=1}^n \alpha_i^1 \mathbf{x}_i$. This forms the base case for our induction. Now if we assume the statement for a given step t that $\mathbf{w}_t = \sum_{i=1}^n \alpha_i^t \mathbf{x}_i$ then we will see that by gradient descent update:

$$\begin{aligned}
\mathbf{w}_{t+1} &= \mathbf{w}_t - \eta \sum_{i=1}^{n} \ell'(\mathbf{w}_t^\top \mathbf{x}_i, y_i)\mathbf{x}_i - 2\eta\lambda\mathbf{w}_t \\
&= (1-\lambda)\mathbf{w}_t - \eta \sum_{i=1}^{n} \ell'(\mathbf{w}_t^\top \mathbf{x}_i, y_i)\mathbf{x}_i \\
&= (1-\lambda)\sum_{i=1}^{n} \alpha_i^t \mathbf{x}_i - \eta \sum_{i=1}^{n} \ell'(\mathbf{w}_t^\top \mathbf{x}_i, y_i)\mathbf{x}_i
\end{aligned}$$

Thus if we define $\alpha_i^{t+1} = (1-\lambda)\alpha_i^t - \eta\ell'(\mathbf{w}_t^\top \mathbf{x}_i, y_i)$ then we have the inductive step that

$$\mathbf{w}_{t+1} = \sum_{i=1}^{n} \alpha_i^{t+1} \mathbf{x}_i$$

Hence by induction we have shown that every $\mathbf{w}_t$ can be written as a linear combination of data points $\mathbf{x}_i$'s. Since gradient descent is assumed to converge to the minimizer, we have that the minimizer satisfies the claimed form.

So far we have shown that the claim holds that the solution to the optimization problem that SVM, logistic regression, linear regression etc all take the form of linear combination of data points. That is in feature space case $\mathbf{w} = \sum_{i=1}^n \alpha_i \phi(\mathbf{x}_i)$ for some values of α_i. We also showed that with this claim, and the fact that one can compute the inner product $\phi(\mathbf{x})^\top \phi(\mathbf{x}')$ directly using the kernel function without enumerating ϕ for computing predictions. We are left with two things now. **A. how do we obtain the α_i's, and B. what functions constitute of kernel functions.**

To answer the first question, note that minimizing $\mathbf{w}, b$ can now equivalently be written as $\mathbf{w} =$ where α's are solution to optimization:

$$\begin{aligned}
\alpha_1, \ldots, \alpha_n, b &= \underset{\alpha_1,\ldots,\alpha_n \in \mathbb{R}, b \in \mathbb{R}}{\arg\min} \sum_{j=1}^{n} \ell(\sum_{i=1}^{n} \alpha_i \phi(\mathbf{x}_i)^\top \phi(x_j) + b, y_j) + \lambda \left(\sum_{i=1}^{n} \alpha_i \phi(\mathbf{x}_i)\right)^\top \left(\sum_{i=1}^{n} \alpha_i \phi(\mathbf{x}_i)\right) \\
&= \underset{\alpha_1,\ldots,\alpha_n \in \mathbb{R}, b \in \mathbb{R}}{\arg\min} \sum_{j=1}^{n} \ell(\sum_{i=1}^{n} \alpha_i \phi(\mathbf{x}_i)^\top \phi(x_j) + b, y_j) + \lambda \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j) \\
&= \underset{\alpha_1,\ldots,\alpha_n \in \mathbb{R}, b \in \mathbb{R}}{\arg\min} \sum_{j=1}^{n} \ell(\sum_{i=1}^{n} \alpha_i k(x_i, x_j) + b, y_j) + \lambda \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j k(x_i, x_j)
\end{aligned}$$

In fact, if we write down an $n \times n$ kernel matrix K as

$$K_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$$

then we can write the optimization as:

$$\alpha_1, \ldots, \alpha_n, b = \underset{\alpha \in \mathbb{R}^n, b \in \mathbb{R}}{\arg\min} \sum_{j=1}^{n} \ell(K\alpha + b, y_j) + \lambda \alpha K \alpha$$

One can use gradient descent on α's this time and perform the above optimization. Now the final thing we are left with is the question of how do we find kernel functions. To this end:

A function $k : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$ corresponds to inner product w.r.t. some feature map ϕ if and only if the function is positive semi-definite. That is, for any set of points, $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and any n, the kernel matrix given by $K_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$ is positive semi-definite. However this property is hard to verify. Below is a set of rules that help in practice. The following are all valid kernel functions:

1. $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$
2. $k(\mathbf{x}, \mathbf{x}') = ck_1(\mathbf{x}, \mathbf{x}')$
3. $k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}')$
4. $k(\mathbf{x}, \mathbf{x}') = g(k(\mathbf{x}, \mathbf{x}'))$
5. $k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}')$
6. $k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})k_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}')$
7. $k(\mathbf{x}, \mathbf{x}') = \exp(k_1(\mathbf{x}, \mathbf{x}'))$
8. $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top A \mathbf{x}'$

where k_1, k_2 are valid kernels, $c > 0$, g is a polynomial function with positive coefficients, f is any function and A is a positive definite matrix.

Quiz 4: Using the properties above show that the following functions are valid kernels:

$$k(x, x') = \exp\left(-\frac{(x - x')^2}{2\sigma^2}\right) \qquad \text{and} \qquad k(S, S') = \exp(|S \cap S'|)$$

where in the above $S_1, S_2 \subset \Omega$ where you can think of Ω as a finite set and for a set A, $|A|$ denotes the cardinality of A.

Claim: Any w that is solution to optimization problem

$$\mathbf{w}, b = \arg\min_{\mathbf{w},b} \sum_{i=1}^{n} \ell(\mathbf{w}^\top \Phi(\mathbf{x}_i) + b, y_i) + \lambda \mathbf{w}^\top \mathbf{w}$$

takes the form $\mathbf{w} = \sum_{i=1}^{n} \alpha_i \Phi(\mathbf{x}_i)$