CSE 6363 - Machine Learning

Kernel Functions

Alex Dillhoff

University of Texas at Arlington

What is covered?

1. Definition
2. Dual Representation
3. Common kernel functions

Kernel Functions

Parametric models use training data to estimate a set of parameters that can then be used to perform inference on new data.

An alternative approach uses **nonparametric methods**: the function is estimated directly from the data.

Kernel Functions

One possible downside to such an approach is that it becomes less efficient as the amount of training data increases.

Explicitly transforming into a feature space such that the data becomes linearly separable may be intractable.

Kernel Functions

Consider sequential data such as text or audio.

If each sample has a variable number of features, how do we account for this using standard linear models with a fixed number of parameters?

Kernel Functions

These situations can be overcome through the use of the **kernel trick**.

Computing a measure of similarity between samples in the feature space obviates the need to directly transform each individual sample to that space.

Kernel Functions

A kernel function is defined as

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}'),$$

where ϕ is some function which transforms the input to a feature space.

Kernel Functions

Methods that require part or all of the training data to make prediction will benefit from using kernel representations, especially when using high dimensional data.

Instead of transforming the data into a high dimensional space which may be computationally intractable, a measure of similarity via the *inner product* is used.

Kernel Functions

The inner product is not the projection into some space.

It represents the outcome of that projection.

Polynomial Kernels

To gain a visual understanding of the usefulness of kernel functions, let's look at a simple example using a polynomial kernel.

$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + c)^d.$$

Polynomial Kernels

This is a common choice for solving problems akin to polynomial regression.

We can use this kernel to present a visual explanation of kernel functions.

Consider the following dataset.

Polynomial Kernels

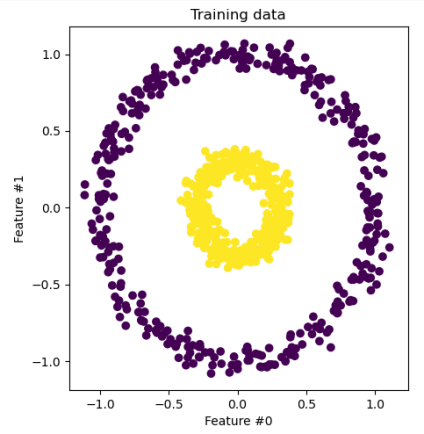

Figure 1: Binary classification dataset that is not linearly separable.

Polynomial Kernels

It is easy enough to see that this dataset could not be separated using a hyperplane in 2D.

We could separate the two using some nonlinear decision boundary like a circle.

Polynomial Kernels

Can we change perspective such that we see a linear decision boundary?

Polynomial Kernels

If we could transform this into 3D space, we could come up with some features such that it is linearly separable in 3D.

For example, let $\phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$.

Polynomial Kernels

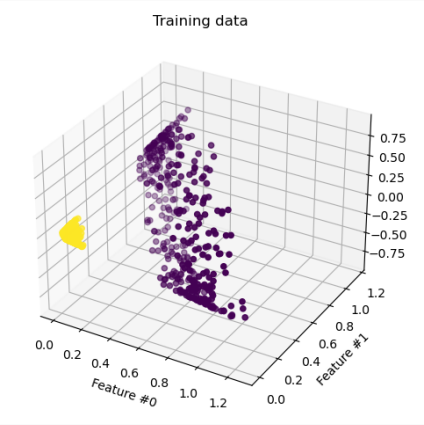

Figure 2: Dataset transformed into a 3D feature space.

Polynomial Kernels

From this perspective, we can clearly see that the data is linearly separable.

The question remains: if we only have the original 2D features, how do we compare points in this 3D features space without explicitly transforming each point?

Polynomial Kernels

The kernel function corresponding to the feature transform above is

$$
\begin{aligned}
k(\mathbf{x}, \mathbf{x}') &= (\mathbf{x}^T \mathbf{x}')^2 \\
&= (x_1 x_1' + x_2 x_2')^2 \\
&= 2 x_1 x_1' x_2 x_2' + (x_1 x_1')^2 + (x_2 x_2')^2 \\
&= \phi(\mathbf{x})^T \phi(\mathbf{x}')
\end{aligned}
$$

Polynomial Kernels

where

$$\phi(\mathbf{x}) = \begin{bmatrix} \sqrt{2}x_1x_2 \\ x_1^2 \\ x_2^2 \end{bmatrix}.$$

Dual Representation

How do we utilize the kernel trick for our problem?

The key to taking advantage of the kernel trick relies on reformulating our linear model into a dual representation.

Dual Representation

Consider the least squares loss with $L2$ regularization.

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}(\mathbf{w}^T\phi(\mathbf{x}_i) - y_i)^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

Dual Representation

ϕ is a basis function that transforms the input.

This could also be a simple identity function in which $\phi(\mathbf{x}) = \mathbf{x}$.

To solve for $\mathbf{w}$, we take the gradient of $J(\mathbf{w})$ with respect to $\mathbf{w}$ and set it to 0.

Dual Representation

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \sum_{i=1}^{n} (\mathbf{w}^T \phi(\mathbf{x}_i) - y_i)\phi(\mathbf{x}_i) + \lambda \mathbf{w}$$

$$\implies \mathbf{w} = -\frac{1}{\lambda} \sum_{i=1}^{n} (\mathbf{w}^T \phi(\mathbf{x}_i) - y_i)\phi(\mathbf{x}_i)$$

Dual Representation

We can formulate this as a matrix-vector product by letting

$$\mathbf{\Phi} = \begin{bmatrix} \phi(\mathbf{x}_1)^T \\ \vdots \\ \phi(\mathbf{x}_n)^T \end{bmatrix} \text{ and } a_i = -\frac{1}{\lambda}(\mathbf{w}^T\phi(\mathbf{x}_i) - y_i).$$

Dual Representation

$\mathbf{w} = \mathbf{\Phi}^T \mathbf{a}$, where $\mathbf{a} = [a_1, \ldots, a_n]^T$.

The dual representation is derived by reformulating $J(\mathbf{w})$ in terms of $\mathbf{a}$.

Dual Representation

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^T\mathbf{\Phi}\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{\Phi}^T\mathbf{a} - \mathbf{a}^T\mathbf{\Phi}\mathbf{\Phi}^T\mathbf{y} + \frac{1}{2}\mathbf{y}^T\mathbf{y} + \frac{\lambda}{2}\mathbf{a}^T\mathbf{\Phi}\mathbf{\Phi}^T\mathbf{a},$$

where $\mathbf{y} = [y_1, \ldots, y_n]$.

Dual Representation

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}(\mathbf{w}^T\phi(\mathbf{x}_i) - y_i)^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^T\mathbf{\Phi}\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{\Phi}^T\mathbf{a} - \mathbf{a}^T\mathbf{\Phi}\mathbf{\Phi}^T\mathbf{y} + \frac{1}{2}\mathbf{y}^T\mathbf{y} + \frac{\lambda}{2}\mathbf{a}^T\mathbf{\Phi}\mathbf{\Phi}^T\mathbf{a}$$

$$\mathbf{\Phi} = \begin{bmatrix} \phi(\mathbf{x}_1)^T \\ \vdots \\ \phi(\mathbf{x}_n)^T \end{bmatrix}, a_i = -\frac{1}{\lambda}(\mathbf{w}^T\phi(\mathbf{x}_i) - y_i),$$

$\mathbf{w} = \mathbf{\Phi}^T\mathbf{a}$, and $\mathbf{a} = [a_1, \ldots, a_n]^T$.

Dual Representation

$\mathbf{\Phi}\mathbf{\Phi}^T$ relate to our original kernel form: $\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$.

This product defines a **Gram matrix** $\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}^T$ whose elements are $k(\mathbf{x}_i, \mathbf{x}_j)$.

Dual Representation

Thus, we can rewrite $J(\mathbf{a})$ as

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^T\mathbf{K}\mathbf{K}\mathbf{a} - \mathbf{a}^T\mathbf{K}\mathbf{y} + \frac{1}{2}\mathbf{y}^T\mathbf{y} + \frac{\lambda}{2}\mathbf{a}^T\mathbf{K}\mathbf{a}.$$

Dual Representation

Solving for $\mathbf{a}$ can be done by computing the gradient of $J(\mathbf{a})$ with respect to $\mathbf{a}$ and setting the result to 0.

$$\begin{aligned}
\nabla_{\mathbf{a}} J(\mathbf{a}) = \mathbf{K}\mathbf{K}\mathbf{a} - \mathbf{K}\mathbf{y} + \lambda\mathbf{K}\mathbf{a} &= 0 \\
\mathbf{K}\mathbf{a} + \lambda I\mathbf{a} - \mathbf{y} &= 0 \\
(\mathbf{K} + \lambda I)\mathbf{a} &= \mathbf{y} \\
\mathbf{a} &= (\mathbf{K} + \lambda I)^{-1}\mathbf{y}.
\end{aligned}$$

Dual Representation

With $\mathbf{a}$ solved, we can complete the dual representation of our original linear regression model.

Dual Representation

Recall that

$$h(\mathbf{x}; \mathbf{w}) = \mathbf{w}^T \phi(\mathbf{x}).$$

If we substitute $\mathbf{w} = \mathbf{\Phi}^T \mathbf{a}$, we get

$$\begin{aligned} f(\mathbf{x}; \mathbf{a}) &= \mathbf{a}^T \mathbf{\Phi} \phi(\mathbf{x}) \\ &= \left[(\mathbf{K} + \lambda I)^{-1} \mathbf{y}) \right]^T \mathbf{\Phi} \phi(\mathbf{x}). \end{aligned}$$

Dual Representation

The kernel form is apparent in the product $\mathbf{\Phi}\phi(\mathbf{x})$.

Dual Representation

If we let $k_i(\mathbf{x}) = k(\mathbf{x}_i, \mathbf{x})$ and

$$\mathbf{k}(\mathbf{x}) = \begin{bmatrix} k_1(\mathbf{x}) \\ \vdots \\ k_n(\mathbf{x}) \end{bmatrix},$$

we can write the dual representation of our linear regression model as

$$f(\mathbf{x}) = \mathbf{k}(\mathbf{x})^T(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y}.$$

Back to the Original Formulation

In this dual formulation, the solution for $\mathbf{a}$ can be expressed as a linear combination of elements $\phi(\mathbf{x})$.

Back to the Original Formulation

We start with

$$a_i = -\frac{1}{\lambda}\left(\mathbf{w}^T \phi(\mathbf{x}_i) - y_i\right).$$

Expanding this into individual coefficients yields

$$\begin{aligned} a_i &= -\frac{1}{\lambda}\left(w_1\phi_1(\mathbf{x}_i) + \cdots + w_m\phi_m(\mathbf{x}_i) - y_i\right) \\ &= -\frac{w_1}{\lambda}\phi_1(\mathbf{x}_i) - \cdots - \frac{w_m}{\lambda}\phi_m(\mathbf{x}_i) + \frac{y_i}{\lambda}. \end{aligned}$$

Back to the Original Formulation

We still need to do something about the term $\frac{y_i}{\lambda}$.

For this, we can multiply both sides of our equation by a convenient 1.

Back to the Original Formulation

That convenient 1 is

$$\frac{\phi_1(\mathbf{x}_i) + \cdots + \phi_m(\mathbf{x}_i)}{\phi_1(\mathbf{x}_i) + \cdots + \phi_m(\mathbf{x}_i)}.$$

Back to the Original Formulation

By doing this and grouping the ϕ_j terms, we get

$$\left(\frac{y_i}{\lambda} \cdot \frac{1}{\phi_1(\mathbf{x}_i) + \cdots + \phi_m(\mathbf{x}_i)} - \frac{w_1}{\lambda}\right)\phi_1(\mathbf{x}_i) + \cdots$$
$$+ \left(\frac{y_i}{\lambda} \cdot \frac{1}{\phi_1(\mathbf{x}_i) + \cdots + \phi_m(\mathbf{x}_i)} - \frac{w_m}{\lambda}\right)\phi_m(\mathbf{x}_i).$$

Back to the Original Formulation

We can simplify this by introducing a term

$$c_i = \frac{y_i}{\lambda} \cdot \frac{1}{\phi_1(\mathbf{x}_i) + \cdots + \phi_m(\mathbf{x}_i)}.$$

Back to the Original Formulation

The solution can be rewritten as

$$\left(c_i - \frac{w_1}{\lambda}\right)\phi_1(\mathbf{x}_i) + \cdots + \left(c_i - \frac{w_m}{\lambda}\right)\phi_m(\mathbf{x}_i).$$

Back to the Original Formulation

We can step backwards using intermediate results in the previous section to get back to the original formulation of our linear regression model.

In the next lecture we will learn about Support Vector Machines, which use the kernel trick to produce linear decision boundaries in higher dimensional spaces.

Constructing Kernels

A valid kernel function must satisfy the following conditions:

- Symmetry: $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$
- Positive semi-definite Gram Matrix: $\mathbf{K} \geq 0$

Constructing Kernels

If the feature space can be represented as a dot product, then it will satisfy the first condition by definition.

The second condition can be shown by constructing a Gram matrix $\mathbf{K}$ and showing that it is positive semi-definite.

A matrix $\mathbf{K}$ is positive semi-definite if and only if $\mathbf{v}^T\mathbf{K}\mathbf{v} \geq 0$ for all $\mathbf{v} \in \mathbb{R}^n$.

Direct Construction of a Kernel

In this approach, we define a feature space $\phi(\mathbf{x})$ and then compute the kernel function as

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}').$$

Direct Construction of a Kernel

This is the approach used in the example from above.

In that example, we used the kernel function $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}')^2$.

For our 2D input, the feature space is $\phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2} x_1 x_2)$.

Construction from valid kernels

As a more convenient approach, it is possible to construct complex kernels from known kernels.

Given valid kernels $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$, we can construct a new kernel $k(\mathbf{x}, \mathbf{x}')$ using operations on the next slide.

Construction from valid kernels

- $k(\mathbf{x}, \mathbf{x}') = ck_1(\mathbf{x}, \mathbf{x}')$ for $c > 0$
- $k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})k_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}')$ for $f(\mathbf{x})$
- $k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}')$
- $k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}')$
- $k(\mathbf{x}, \mathbf{x}') = \exp(k_1(\mathbf{x}, \mathbf{x}'))$
- $k(\mathbf{x}, \mathbf{x}') = \tanh(k_1(\mathbf{x}, \mathbf{x}'))$

RBF maps to infinite-dimensional space

RBF maps to ∞ dimensional space

It can be shown that the RBF kernel maps the input to an infinite-dimensional space.

This is a result of the Taylor series expansion of the exponential function.

RBF maps to ∞ dimensional space

The RBF kernel is defined as

$$k(\mathbf{x}, \mathbf{x}') = \exp\Big(- \frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2} \Big).$$

RBF maps to ∞ dimensional space

The Taylor series expansion of the exponential function is

$$\exp(x) = \sum_{n=0}^{\infty} \frac{x^n}{n!}.$$

RBF maps to ∞ dimensional space

Substituting the RBF kernel into the Taylor series expansion yields

$$\exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}\right) = \sum_{n=0}^{\infty} \frac{\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}\right)^n}{n!}.$$

RBF maps to ∞ dimensional space

The benefit of this result is that it allows us to work in a high-dimensional space without explicitly transforming the input.

This is especially useful when the input space is infinite-dimensional, such as with text data.

It is also used to compare the similarity of documents without explicitly transforming the input into a high-dimensional space.