Kernel Methods

He He

Slides based on Lecture 4d from David Rosenberg's course material.

CDS, NYU

February 22, 2022

Logistics:

- Please fill out the course survey so that we can better help you
- Feel free to ask questions during the lab

Today's lecture:

- Wrap-up SVM
- Motivation of kernel methods: Our data is typically not linearly separable, but we like to work with linear models.
- Adding features (going to high-dimensional space) allow us to use linear models for complex data.
- Kernels allow us to think about similarities rather than feature engineering.

Feature Maps

The Input Space $\mathcal{X}$

- Our general learning theory setup: no assumptions about $\mathcal{X}$
- But $\mathcal{X} = \mathrm{R}^d$ for the specific methods we've developed:
  - Ridge regression
  - Lasso regression
  - Support Vector Machines
- Our hypothesis space for these was all affine functions on R^d:

$$\mathcal{F} = \left\{ x \mapsto w^T x + b \mid w \in \mathrm{R}^d, b \in \mathrm{R} \right\}.$$

- What if we want to do prediction on inputs not natively in R^d?

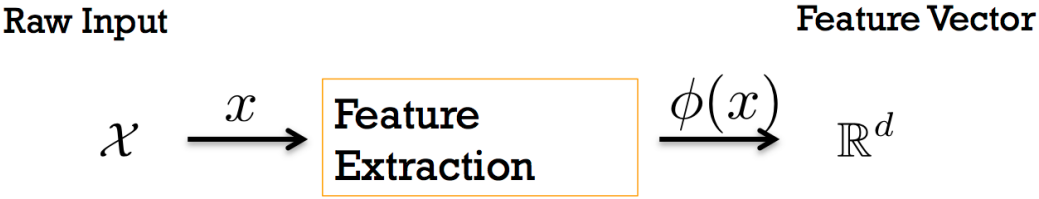
The Input Space $\mathcal{X}$

- Often want to use inputs not natively in R^d:
  - Text documents
  - Image files
  - Sound recordings
  - DNA sequences
- But everything in a computer is a sequence of numbers
  - The ith entry of each sequence should have the same "meaning"
  - All the sequences should have the same length

Feature Extraction

Definition

Mapping an input from $\mathcal{X}$ to a vector in R^d is called **feature extraction** or **featurization**.



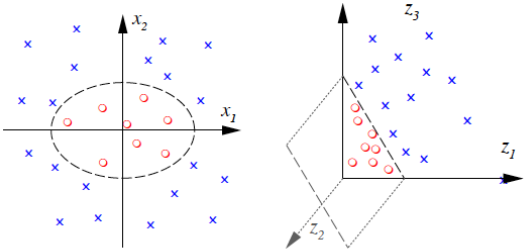
Linear Models with Explicit Feature Map

- Input space: $\mathcal{X}$ (no assumptions)
- Introduce **feature map** $\phi : \mathcal{X} \to \mathrm{R}^d$
- The feature map maps into the **feature space** R^d.
- Hypothesis space of affine functions on feature space:

$$\mathcal{F} = \left\{ x \mapsto w^T \phi(x) + b \mid w \in \mathrm{R}^d, b \in \mathrm{R} \right\}.$$

Geometric Example: Two class problem, nonlinear boundary

$$\Phi : R^2 \to R^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{(2)} x_1 x_2, x_2^2)$$



- With identity feature map $\phi(x) = (x_1, x_2)$ and linear models, can't separate regions
- With appropriate featurization $\phi(x) = (x_1, x_2, x_1^2 + x_2^2)$, becomes linearly separable .
- Video: http://youtu.be/3liCbRZPrZA

Expressivity of Hypothesis Space

- For linear models, to grow the hypothesis spaces, we must add features.
- Sometimes we say a larger hypothesis is more expressive.
  - (can fit more relationships between input and action)
- Many ways to create new features.

Handling Nonlinearity with Linear Methods

Example Task: Predicting Health

- General Philosophy: Extract every feature that might be relevant
- Features for medical diagnosis
  - height
  - weight
  - body temperature
  - blood pressure
  - etc...

From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.

Feature Issues for Linear Predictors

- For linear predictors, it's important **how** features are added
  - The relation between a feature and the label may not be linear
  - There may be complex dependence among features
- Three types of nonlinearities can cause problems:
  - Non-monotonicity
  - Saturation
  - Interactions between features

From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.

Non-monotonicity: The Issue

- Feature Map: $\phi(x) = [1, \text{temperature}(x)]$
- Action: Predict health score $y \in \mathbb{R}$ (positive is good)
- Hypothesis Space $\mathcal{F}$={affine functions of temperature}
- Issue:
  - Health is not an affine function of temperature.
  - Affine function can either say
    - Very high is bad and very low is good, or
    - Very low is bad and very high is good,
    - But here, both extremes are bad.

From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.

Non-monotonicity: Solution 1

- Transform the input:

$$\phi(x) = \left[1, \{\text{temperature(x)-37}\}^2\right],$$

  where 37 is "normal" temperature in Celsius.

- Ok, but requires manually-specified domain knowledge
  - Do we really need that?
  - What does $w^T \phi(x)$ look like?

From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.

Non-monotonicity: Solution 2

- Think less, put in more:

$$\phi(x) = \left[1, \text{temperature}(x), \{\text{temperature}(x)\}^2\right].$$

- More expressive than Solution 1.

General Rule

Features should be simple building blocks that can be pieced together.

From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.

Saturation: The Issue

- Setting: Find products relevant to user's query
- Input: Product x
- Action: Score the relevance of x to user's query
- Feature Map:

$$\phi(x) = [1, N(x)],$$

  where $N(x) =$ number of people who bought x.

- We expect a monotonic relationship between $N(x)$ and relevance, but also expect diminishing return.

From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.

Saturation: Solve with nonlinear transform

- Smooth nonlinear transformation:

$$\phi(x) = [1, \log\{1 + N(x)\}]$$

  - $\log(\cdot)$ good for values with large dynamic ranges

- Discretization (a discontinuous transformation):

$$\phi(x) = (1(0 \leqslant N(x) < 10), 1(10 \leqslant N(x) < 100), \ldots)$$

  - Small buckets allow quite flexible relationship

From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.

Interactions: The Issue

- Input: Patient information x
- Action: Health score $y \in \mathbb{R}$ (higher is better)
- Feature Map

$$\phi(x) = [\text{height}(x), \text{weight}(x)]$$

- Issue: It's the weight *relative* to the height that's important.
- Impossible to get with these features and a linear classifier.
- Need some **interaction** between height and weight.

From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.

Interactions: Approach 1

- Google "ideal weight from height"
- J. D. Robinson's "ideal weight" formula (for a male):

$$\text{weight(kg)} = 52 + 1.9\,[\text{height(in)} - 60]$$

- Make score square deviation between height(h) and ideal weight(w)

$$f(x) = (52 + 1.9\,[h(x) - 60] - w(x))^2$$

- WolframAlpha for complicated Mathematics:

$$f(x) = 3.61h(x)^2 - 3.8h(x)w(x) - 235.6h(x) + w(x)^2 + 124w(x) + 3844$$

From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.

Interactions: Approach 2

- Just include all second order features:

$$\phi(x) = \left[1, h(x), w(x), h(x)^2, w(x)^2, \underbrace{h(x)w(x)}_{\text{cross term}}\right]$$

- More flexible, no Google, no WolframAlpha.

General Principle

Simpler building blocks replace a single "smart" feature.

From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.

Monomial Interaction Terms

Interaction terms are useful building blocks to model non-linearities in features.

- Suppose we start with $x = (1, x_1, \ldots, x_d) \in \mathrm{R}^{d+1} = \mathcal{X}$.
- Consider adding all **monomials** of degree M: $x_1^{p_1} \cdots x_d^{p_d}$, with $p_1 + \cdots + p_d = M$.
  - Monomials with degree 2 in 2D space: x_1^2, x_2^2, $x_1 x_2$
- How many features will we end up with? $\binom{M+d-1}{M}$ ("stars and bars")
- This leads to extremely large data matrices
  - For $d = 40$ and $M = 8$, we get 314457495 features.

Big Feature Spaces

Very large feature spaces have two potential issues:

- Overfitting
- Memory and computational costs

Solutions:

- Overfitting we handle with regularization.
- **Kernel methods** can help with memory and computational costs when we go to high (or infinite) dimensional spaces.

The Kernel Trick

SVM with Explicit Feature Map

- Let $\psi : \mathcal{X} \to \mathbb{R}^d$ be a feature map.
- The SVM objective (with explicit feature map):

$$\min_{w \in \mathbb{R}^d} \frac{1}{2}||w||^2 + \frac{c}{n} \sum_{i=1}^{n} \max\left(0, 1 - y_i w^T \psi(x_i)\right).$$

- Computation is costly if d is large (e.g. with high-degree monomials)
- Last time we mentioned an equivalent optimization problem from Lagrangian duality.

SVM Dual Problem

- By Lagrangian duality, it is equivalent to solve the following dual problem:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \psi(x_j)^T \psi(x_i) \\ \text{s.t.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \quad \text{and} \quad \alpha_i \in \left[0, \frac{c}{n}\right] \quad \forall i. \end{aligned}$$

- If α^* is an optimal value, then

$$w^* = \sum_{i=1}^{n} \alpha_i^* y_i \psi(x_i) \quad \text{and} \quad \hat{f}(x) = \sum_{i=1}^{n} \alpha_i^* y_i \psi(x_i)^T \psi(x).$$

- Key observation: $\psi(x)$ only shows up in inner products with another $\psi(x')$ *for both training and inference*.

Compute the Inner Products

Consider 2D data. Let's introduce degree-2 monomials using $\psi : \mathbb{R}^2 \to \mathbb{R}^3$.

$$(x_1, x_2) \mapsto (x_1^2, \sqrt{2}x_1x_2, x_2^2).$$

The inner product is

$$\begin{aligned}
\psi(x)^T\psi(x') &= x_1^2 x_1'^2 + (\sqrt{2}x_1x_2)(\sqrt{2}x_1'x_2') + x_2^2 x_2'^2 \\
&= (x_1x_1')^2 + 2(x_1x_1')(x_2x_2') + (x_2x_2')^2 \\
&= (x_1x_1' + x_2x_2')^2 \\
&= (x^Tx')^2
\end{aligned}$$

We can calculate the inner product $\psi(x)^T\psi(x')$ in the original input space without accessing the features $\psi(x)$!

Compute the Inner Products

Now, consider monomials up to degree-2:

$$(x_1, x_2) \mapsto (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2).$$

The inner product can be computed by

$$\psi(x)^T\psi(x') = (1 + x^T x')^2 \quad \text{(check)}.$$

More generally, for features maps producing monomials up to degree-p, we have

$$\psi(x)^T\psi(x') = (1 + x^T x')^p.$$

(Note that the coefficients of each monomial in ψ may not be 1)

Kernel trick: we do not need explicit features to calculate inner products.

- Using explicit features: $O(d^p)$
- Using implicit computation: $O(d)$

Kernel Function

The Kernel Function

- **Input space**: $\mathcal{X}$
- **Feature space**: $\mathcal{H}$ (a Hilbert space, e.g. R^d)
- **Feature map**: $\psi : \mathcal{X} \to \mathcal{H}$
- The **kernel function** corresponding to ψ is

$$k(x, x') = \left\langle \psi(x), \psi(x') \right\rangle,$$

  where $\langle \cdot, \cdot \rangle$ is the inner product associated with $\mathcal{H}$.

Why introduce this new notation $k(x, x')$?

- We can often evaluate $k(x, x')$ without explicitly computing $\psi(x)$ and $\psi(x')$.

When can we use the kernel trick?

Some Methods Can Be "Kernelized"

Definition

A method is **kernelized** if every feature vector $\psi(x)$ only appears inside an inner product with another feature vector $\psi(x')$. This applies to both the optimization problem and the prediction function.

The SVM Dual is a kernelization of the original SVM formulation.

Optimization:

$$\text{maximize} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \psi(x_j)^T \psi(x_i)$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \qquad \text{and} \qquad \alpha_i \in \left[0, \frac{C}{n}\right] \quad \forall i.$$

Prediction:

$$\hat{f}(x) = \sum_{i=1}^{n} \alpha_i^* y_i \psi(x_i)^T \psi(x).$$

The Kernel Matrix

Definition

The **kernel matrix** for a kernel k on $x_1, \ldots, x_n \in \mathcal{X}$ is

$$K = \big(k(x_i, x_j)\big)_{i,j} = \begin{pmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \vdots & \ddots & \ldots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{pmatrix} \in \mathrm{R}^{n \times n}.$$

- In ML this is also called a **Gram matrix**, but traditionally (in linear algebra), Gram matrices are defined without reference to a kernel or feature map.

The Kernel Matrix

- The kernel matrix summarizes all the information we need about the training inputs $x_1, \ldots, x_n$ to solve a kernelized optimization problem.
- In the kernelized SVM, we can replace $\psi(x_i)^T \psi(x_j)$ with K_{ij}:

$$\begin{aligned} \text{maximize}_\alpha \quad & \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j K_{ij} \\ \text{s.t.} \quad & \sum_{i=1}^n \alpha_i y_i = 0 \qquad \text{and} \qquad \alpha_i \in \left[0, \frac{C}{n}\right] \; i = 1, \ldots, n. \end{aligned}$$

Kernel Methods

Given a kernelized ML algorithm (i.e. all $\psi(x)$'s show up as $\langle \psi(x), \psi(x') \rangle$),

- Can swap out the inner product for a new kernel function.
- New kernel may correspond to a very high-dimensional feature space.
- Once the kernel matrix is computed, the computational cost depends on number of data points n, rather than the dimension of feature space d.
- Useful when $d >> n$.
- Computing the kernel matrix may still depend on d and the essence of the **trick** is getting around this $O(d)$ dependence.

Example Kernels

Kernels as Similarity Scores

- Often useful to think of the $k(x, x')$ as a **similarity score** for x and x'.
- We can design similarity functions without thinking about the explicit feature map, e.g. "string kernels", "graph kerners".
- How do we know that our kernel functions actually correspond to inner products in some feature space?

How to Get Kernels?

- Explicitly construct $\psi(x) : \mathcal{X} \to \mathbb{R}^d$ (e.g. monomials) and define $k(x, x') = \psi(x)^T \psi(x')$.
- Directly define the kernel function $k(x, x')$ ("similarity score"), and verify it corresponds to $\langle \psi(x), \psi(x') \rangle$ for some ψ.

There are many theorems to help us with the second approach.

Linear Algebra Review: Positive Semidefinite Matrices

Definition

A real, symmetric matrix $M \in \mathrm{R}^{n \times n}$ is **positive semidefinite (psd)** if for any $x \in \mathrm{R}^n$,

$$x^T M x \geqslant 0.$$

Theorem

The following conditions are each necessary and sufficient for a symmetric matrix M to be positive semidefinite:

- *M can be factorized as $M = R^T R$, for some matrix R.*
- *All eigenvalues of M are greater than or equal to 0.*

Positive Definite Kernel

Definition

A symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathrm{R}$ is a **positive definite (pd)** kernel on $\mathcal{X}$ if for any finite set $\{x_1, \ldots, x_n\} \in \mathcal{X}$ ($n \in \mathbb{N}$), the kernel matrix on this set

$$K = \big(k(x_i, x_j)\big)_{i,j} = \begin{pmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \vdots & \ddots & \cdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{pmatrix}$$

is a positive semidefinite matrix.

- Symmetric: $k(x, x') = k(x', x)$
- The kernel matrix needs to be positive semidefinite for any finite set of points.
- Equivalent definition: $\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j k(x_i, x_j) \geqslant 0$ given $\alpha_i \in \mathrm{R}\, \forall i$.

Mercer's Theorem

Theorem

A symmetric function $k(x, x')$ can be expressed as an inner product

$$k(x, x') = \langle \psi(x), \psi(x') \rangle$$

*for some ψ if and only if $k(x, x')$ is **positive definite**.*

- Proving a kernel function is positive definite is typically not easy.
- But we can construct new kernels from valid kernels.

Generating New Kernels from Old

- Suppose $k, k_1, k_2 : \mathcal{X} \times \mathcal{X} \to \mathrm{R}$ are pd kernels. Then so are the following:

$$
\begin{aligned}
k_{\text{new}}(x, x') &= \alpha k(x, x') \quad \text{for } \alpha \geqslant 0 \quad \text{(non-negative scaling)} \\
k_{\text{new}}(x, x') &= k_1(x, x') + k_2(x, x') \quad \text{(sum)} \\
k_{\text{new}}(x, x') &= k_1(x, x') k_2(x, x') \quad \text{(product)} \\
k_{\text{new}}(x, x') &= k(\psi(x), \psi(x')) \text{ for any function } \psi(\cdot) \quad \text{(recursion)} \\
k_{\text{new}}(x, x') &= f(x) f(x') \text{ for any function } f(\cdot) \quad (f \text{ as 1D feature map})
\end{aligned}
$$

- Lots more theorems to help you construct new kernels from old.

Based on Mark Schmidt's slides: https://www.cs.ubc.ca/~schmidtm/Courses/540-W19/L12.5.pdf

Linear Kernel

- Input space: $\mathcal{X} = \mathrm{R}^d$
- Feature space: $\mathcal{H} = \mathrm{R}^d$, with standard inner product
- Feature map

$$\psi(x) = x$$

- Kernel:

$$k(x, x') = x^T x'$$

Quadratic Kernel in R^d

- Input space $\mathcal{X} = \mathrm{R}^d$
- Feature space: $\mathcal{H} = \mathrm{R}^D$, where $D = d + \binom{d}{2} \approx d^2/2$.
- Feature map:

$$\psi(x) = (x_1, \ldots, x_d, x_1^2, \ldots, x_d^2, \sqrt{2}x_1x_2, \ldots, \sqrt{2}x_ix_j, \ldots \sqrt{2}x_{d-1}x_d)^T$$

- Then for $\forall x, x' \in \mathrm{R}^d$

$$\begin{aligned} k(x, x') &= \langle \psi(x), \psi(x') \rangle \\ &= \langle x, x' \rangle + \langle x, x' \rangle^2 \end{aligned}$$

- Computation for inner product with explicit mapping: $O(d^2)$
- Computation for implicit kernel calculation: $O(d)$.

Polynomial Kernel in R^d

- Input space $\mathcal{X} = \mathrm{R}^d$
- Kernel function:
$$k(x, x') = \left(1 + \langle x, x' \rangle\right)^M$$
- Corresponds to a feature map with all monomials up to degree M.
- For any M, computing the kernel has same computational cost
- Cost of explicit inner product computation grows rapidly in M.

Radial Basis Function (RBF) / Gaussian Kernel

Input space $\mathcal{X} = \mathrm{R}^d$

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

where σ^2 is known as the bandwidth parameter.

- Probably the most common nonlinear kernel.
- Does it act like a similarity score?
- Have we departed from our "inner product of feature vector" recipe?
  - Yes and no: corresponds to an infinite dimensional feature vector

Remaining Questions

Our current recipe:

- Recognize kernelized problem: $\psi(x)$ only occur in inner products $\psi(x)^T\psi(x')$
- Pick a kernel function ("similarity score")
- Compute the kernel matrix (n by n where n is the dataset size)
- Optimize the model and make predictions by accessing the kernel matrix

Next: When can we apply kernelization?

Representer Theorem

SVM solution is in the "span of the data"

- We found the SVM dual problem can be written as:

$$\begin{aligned} \sup_{\alpha \in \mathbb{R}^n} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_j^T x_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \\ & \alpha_i \in \left[0, \frac{c}{n}\right] \; i = 1, \ldots, n. \end{aligned}$$

- Given dual solution α^*, primal solution is $w^* = \sum_{i=1}^{n} \alpha_i^* y_i x_i$.
- Notice: w^* is a linear combination of training inputs $x_1, \ldots, x_n$.
- We refer to this phenomenon by saying "w^* is in the **span of the data**."
  - Or in math, $w^* \in \text{span}(x_1, \ldots, x_n)$.

Ridge regression solution is in the "span of the data"

- The ridge regression solution for regularization parameter $\lambda > 0$ is

$$w^* = \underset{w \in \mathbb{R}^d}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \left\{ w^T x_i - y_i \right\}^2 + \lambda \|w\|_2^2.$$

- This has a closed form solution (Homework #3):

$$w^* = \left(X^T X + \lambda I \right)^{-1} X^T y,$$

where X is the design matrix, with $x_1, \ldots, x_n$ as rows.

Ridge regression solution is in the "span of the data"

- Rearranging $w^* = \left(X^T X + \lambda I\right)^{-1} X^T y$, we can show that (also Homework #3):

$$\begin{aligned} w^* &= X^T \underbrace{\left(\frac{1}{\lambda} y - \frac{1}{\lambda} X w^*\right)}_{\alpha^*} \\ &= X^T \alpha^* = \sum_{i=1}^{n} \alpha_i^* x_i. \end{aligned}$$

- So w^* is in the span of the data.
  - i.e. $w^* \in \text{span}\left(x_1, \ldots, x_n\right)$

If solution is in the span of the data, we can reparameterize

- The ridge regression solution for regularization parameter $\lambda > 0$ is

$$w^* = \arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \left\{ w^T x_i - y_i \right\}^2 + \lambda \|w\|_2^2.$$

- We now know that $w^* \in \text{span}(x_1, \ldots, x_n) \subset \mathbb{R}^d$.
- So rather than minimizing over all of $\mathbb{R}^d$, we can minimize over $\text{span}(x_1, \ldots, x_n)$.

$$w^* = \arg\min_{w \in \text{span}(x_1, \ldots, x_n)} \frac{1}{n} \sum_{i=1}^{n} \left\{ w^T x_i - y_i \right\}^2 + \lambda \|w\|_2^2.$$

- Let's reparameterize the objective by replacing w as a linear combination of the inputs.

If solution is in the span of the data, we can reparameterize

- Note that for any $w \in \text{span}(x_1, \ldots, x_n)$, we have $w = X^T \alpha$, for some $\alpha \in \mathbb{R}^n$.
- So let's replace w with $X^T \alpha$ in our optimization problem:

$$\begin{aligned} \text{[original]} \ w^* &= \underset{w \in \mathbb{R}^d}{\arg\min} \frac{1}{n} \sum_{i=1}^n \left\{ w^T x_i - y_i \right\}^2 + \lambda \|w\|_2^2 \\ \text{[reparameterized]} \ \alpha^* &= \underset{\alpha \in \mathbb{R}^n}{\arg\min} \frac{1}{n} \sum_{i=1}^n \left\{ \left(X^T \alpha \right)^T x_i - y_i \right\}^2 + \lambda \|X^T \alpha\|_2^2. \end{aligned}$$

- To get w^* from the reparameterized optimization problem, we just take $w^* = X^T \alpha^*$.
- We changed the dimension of our optimization variable from d to n. Is this useful?

Consider very large feature spaces

- Suppose we have a 300-million dimension feature space [very large]
  - (e.g. using high order monomial interaction terms as features, as described last lecture)
- Suppose we have a training set of 300,000 examples [fairly large]
- In the original formulation, we solve a 300-million dimension optimization problem.
- In the reparameterized formulation, we solve a 300,000-dimension optimization problem.
- This is why we care about when the solution is in the span of the data.
- This reparameterization is interesting when we have more features than data ($d \gg n$).

What's next?

- For SVM and ridge regression, we found that the solution is in the span of the data.
  - derived in two rather ad-hoc ways
- Up next: The Representer Theorem, which shows that this "span of the data" result occurs far more generally, and we prove it using basic linear algebra.

Math Review: Inner Product Spaces and Hilbert Spaces

Hypothesis spaces we've seen so far

Finite-dimensional vector space (linear functions):

$$\mathcal{H} = \left\{ f : \mathcal{X} \to \mathrm{R} \mid f(x) = w^T x, \quad w, x \in \mathrm{R}^d \right\} .$$

To consider more complex input spaces (e.g. text, images), we use a feature map $\phi : \mathcal{X} \to \mathcal{F}$:

$$\mathcal{H} = \left\{ f : \mathcal{X} \to \mathrm{R} \mid f(x) = w^T \phi(x) \right\} .$$

- ϕ does not have to be linear.
- The feature space $\mathcal{F}$ can be R^d (Euclidean space) or an infinite-dimensional vector space.
- We would like more structure on $\mathcal{F}$.

Inner Product Space (or "Pre-Hilbert" Spaces)

An **inner product space** (over reals) is a vector space $\mathcal{V}$ with an **inner product**, which is a mapping

$$\langle \cdot, \cdot \rangle : \mathcal{V} \times \mathcal{V} \to \mathrm{R}$$

that has the following properties: $\forall x, y, z \in \mathcal{V}$ and $a, b \in \mathrm{R}$:

- Symmetry: $\langle x, y \rangle = \langle y, x \rangle$
- Linearity: $\langle ax + by, z \rangle = a \langle x, z \rangle + b \langle y, z \rangle$
- Positive-definiteness: $\langle x, x \rangle \geqslant 0$ and $\langle x, x \rangle = 0 \iff x = 0_{\mathcal{V}}$.

To show a function $\langle \cdot, \cdot \rangle$ is an inner product, we need to check the above conditions.

Exercise: show that $\langle x, y \rangle \stackrel{\text{def}}{=} x^T y$ is an inner product on R^d.

Norm from Inner Product

Inner product is nice because it gives us notions of "size", "distance", "angle" in the vector space.

For an inner product space, we can ddefine a norm as

$$\|x\| \stackrel{\text{def}}{=} \sqrt{\langle x, x \rangle}.$$

Example

R^d with standard Euclidean inner product is an inner product space:

$$\langle x, y \rangle := x^T y \qquad \forall x, y \in \mathrm{R}^d.$$

Norm is

$$\|x\| = \sqrt{x^T x}.$$

Orthogonality (Definitions)

Definition

Two vectors are **orthogonal** if $\langle x, x' \rangle = 0$. We denote this by $x \perp x'$.

Definition

x is orthogonal to a set S, i.e. $x \perp S$, if $x \perp s$ for all $x \in S$.

Pythagorean Theorem

Theorem (Pythagorean Theorem)

If $x \perp x'$, then $\|x+x'\|^2 = \|x\|^2 + \|x'\|^2$.

Proof.

We have

$$\begin{aligned} \|x+x'\|^2 &= \langle x+x', x+x' \rangle \\ &= \langle x, x \rangle + \langle x, x' \rangle + \langle x', x \rangle + \langle x', x' \rangle \\ &= \|x\|^2 + \|x'\|^2. \end{aligned}$$

□

Hilbert Space

- A pre-Hilbert space is a vector space equipped with an inner product.
- We need an additional technical condition for Hilbert space: **completeness**.
- A space is **complete** if all Cauchy sequences in the space converge to a point in the space.

Definition

A **Hilbert space** is a complete inner product space.

Example

Any finite dimensional inner produce space is a Hilbert space.

The Representer Theorem

Generalize from SVM Objective

- SVM objective:

$$\min_{w \in \mathrm{R}^d} \frac{1}{2}\|w\|^2 + \frac{c}{n}\sum_{i=1}^{n} \max\left(0, 1 - y_i\left[\langle w, x_i\rangle\right]\right).$$

- **Generalized objective**:

$$\min_{w \in \mathcal{H}} R(\|w\|) + L\left(\langle w, x_1\rangle, \ldots, \langle w, x_n\rangle\right),$$

where

- $w, x_1, \ldots, x_n \in \mathcal{H}$ for some Hilbert space $\mathcal{H}$. (We typically have $\mathcal{H} = \mathrm{R}^d$.)
- $\|\cdot\|$ is the norm corresponding to the inner product of $\mathcal{H}$. (i.e. $\|w\| = \sqrt{\langle w, w\rangle}$)
- $R : [0, \infty) \to \mathrm{R}$ is nondecreasing (**Regularization term**), and
- $L : \mathrm{R}^n \to \mathrm{R}$ is arbitrary (**Loss term**).

General Objective Function for Linear Hypothesis Space (Details)

- **Generalized objective**:

$$\min_{w \in \mathcal{H}} R(\|w\|) + L(\langle w, x_1 \rangle, \ldots, \langle w, x_n \rangle)$$

- We can map x_i to a feature space.
- The prediction/score function $x \mapsto \langle w, x \rangle$ is linear in w.

General Objective Function for Linear Hypothesis Space (Details)

- **Generalized objective**:

$$\min_{w \in \mathcal{H}} R(\|w\|) + L(\langle w, x_1 \rangle, \ldots, \langle w, x_n \rangle)$$

- Ridge regression and SVM are of this form. (Verify this!)
- What if we penalize with $\lambda \|w\|_2$ instead of $\lambda \|w\|_2^2$? Yes!
- What if we use lasso regression? No! ℓ_1 norm does not correspond to an inner product.

The Representer Theorem: Quick Summary

- **Generalized objective**:

$$w^* = \arg\min_{w \in \mathcal{H}} R(\|w\|) + L(\langle w, x_1 \rangle, \ldots, \langle w, x_n \rangle)$$

- Representer theorem tells us we can look for w^* in the span of the data:

$$w^* = \arg\min_{w \in \text{span}(x_1, \ldots, x_n)} R(\|w\|) + L(\langle w, x_1 \rangle, \ldots, \langle w, x_n \rangle).$$

- So we can reparameterize as before:

$$\alpha^* = \arg\min_{\alpha \in \mathbb{R}^n} R\left(\left\|\sum_{i=1}^n \alpha_i x_i\right\|\right) + L\left(\left\langle \sum_{i=1}^n \alpha_i x_i, x_1 \right\rangle, \ldots, \left\langle \sum_{i=1}^n \alpha_i x_i, x_n \right\rangle\right).$$

- Our reparameterization trick applies much more broadly than SVM and ridge.

The Representer Theorem

Theorem (Representer Theorem)

Let

$$J(w) = R(\|w\|) + L(\langle w, x_1 \rangle, \ldots, \langle w, x_n \rangle),$$

where

- $w, x_1, \ldots, x_n \in \mathcal{H}$ *for some Hilbert space* $\mathcal{H}$. *(We typically have* $\mathcal{H} = \mathbb{R}^d$.*)*
- $\|\cdot\|$ *is the norm corresponding to the inner product of* $\mathcal{H}$. *(i.e.* $\|w\| = \sqrt{\langle w, w \rangle}$*)*
- $R : [0, \infty) \to \mathbb{R}$ *is nondecreasing (**Regularization term**), and*
- $L : \mathbb{R}^n \to \mathbb{R}$ *is arbitrary (**Loss term**).*

Then it ***has a minimizer of the form*** $w^* = \sum_{i=1}^n \alpha_i x_i$.

The Representer Theorem (Proof sketch)

Reparameterizing our Generalized Objective Function

Rewriting the Objective Function

- Define the training score function $s : \mathrm{R}^d \to \mathrm{R}^n$ by

$$s(w) = \begin{pmatrix} \langle w, x_1 \rangle \\ \vdots \\ \langle w, x_n \rangle \end{pmatrix},$$

  which gives the **training score vector** for any w.

- We can then rewrite the objective function as

$$J(w) = R(\|w\|) + L(s(w)),$$

  where now $L : \mathrm{R}^{n \times 1} \to \mathrm{R}$ takes a column vector as input.

- This will allow us to have a slick reparameterized version...

Reparameterize the Generalized Objective

- By the Representer Theorem, it's sufficient to minimize $J(w)$ for w of the form $\sum_{i=1}^{n} \alpha_i x_i$.
- Plugging this form into $J(w)$, we see we can just minimize

$$J_0(\alpha) = R\left(\left\|\sum_{i=1}^{n} \alpha_i x_i\right\|\right) + L\left(s\left(\sum_{i=1}^{n} \alpha_i x_i\right)\right)$$

  over $\alpha = (\alpha_1, \ldots, \alpha_n)^T \in \mathbb{R}^{n \times 1}$.

- With some new notation, we can substantially simplify
  - the norm piece $\|w\| = \|\sum_{i=1}^{n} \alpha_i x_i\|$, and
  - the score piece $s(w) = s(\sum_{i=1}^{n} \alpha_i x_i)$.

Simplifying the Reparameterized Norm

- For the norm piece $\|w\| = \|\sum_{i=1}^{n} \alpha_i x_i\|$, we have

$$\begin{aligned} \|w\|^2 &= \langle w, w \rangle \\ &= \left\langle \sum_{i=1}^{n} \alpha_i x_i, \sum_{j=1}^{n} \alpha_j x_j \right\rangle \\ &= \sum_{i,j=1}^{n} \alpha_i \alpha_j \langle x_i, x_j \rangle . \end{aligned}$$

- This expression involves the n^2 inner products between all pairs of input vectors.
- We often put those values together into a matrix (Gram/Kernel matrix).

Example: Gram Matrix for the Dot Product

- Consider $x_1, \ldots, x_n \in \mathbb{R}^{d \times 1}$ with the standard inner product $\langle x, x' \rangle = x^T x'$.
- Let $X \in \mathbb{R}^{n \times d}$ be the **design matrix**, which has each input vector as a row:

$$X = \begin{pmatrix} -x_1^T - \\ \vdots \\ -x_n^T - \end{pmatrix}.$$

- Then the Gram matrix is

$$\begin{aligned} K &= \begin{pmatrix} x_1^T x_1 & \cdots & x_1^T x_n \\ \vdots & \ddots & \cdots \\ x_n^T x_1 & \cdots & x_n^T x_n \end{pmatrix} = \begin{pmatrix} -x_1^T - \\ \vdots \\ -x_n^T - \end{pmatrix} \begin{pmatrix} | & \cdots & | \\ x_1 & \cdots & x_n \\ | & \cdots & | \end{pmatrix} \\ &= XX^T \end{aligned}$$

Simplifying the Reparametrized Norm

- With $w = \sum_{i=1}^{n} \alpha_i x_i$, we have

$$\begin{aligned} \|w\|^2 &= \langle w, w \rangle \\ &= \left\langle \sum_{i=1}^{n} \alpha_i x_i, \sum_{j=1}^{n} \alpha_j x_j \right\rangle \\ &= \sum_{i,j=1}^{n} \alpha_i \alpha_j \langle x_i, x_j \rangle \\ &= \alpha^T K \alpha. \end{aligned}$$

Simplifying the Training Score Vector

- The score for x_j for $w = \sum_{i=1}^{n} \alpha_i x_i$ is

$$\langle w, x_j \rangle = \left\langle \sum_{i=1}^{n} \alpha_i x_i, x_j \right\rangle = \sum_{i=1}^{n} \alpha_i \langle x_i, x_j \rangle$$

- The training score vector is

$$\begin{aligned} s\left(\sum_{i=1}^{n} \alpha_i x_i\right) &= \begin{pmatrix} \sum_{i=1}^{n} \alpha_i \langle x_i, x_1 \rangle \\ \vdots \\ \sum_{i=1}^{n} \alpha_i \langle x_i, x_n \rangle \end{pmatrix} = \begin{pmatrix} \alpha_1 \langle x_1, x_1 \rangle + \cdots + \alpha_n \langle x_n, x_1 \rangle \\ \vdots \\ \alpha_1 \langle x_1, x_n \rangle + \cdots + \alpha_n \langle x_n, x_n \rangle \end{pmatrix} \\ &= \begin{pmatrix} \langle x_1, x_1 \rangle & \cdots & \langle x_1, x_n \rangle \\ \vdots & \ddots & \cdots \\ \langle x_n, x_1 \rangle & \cdots & \langle x_n, x_n \rangle \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{pmatrix} \\ &= K\alpha \end{aligned}$$

Reparameterized Objective

- Putting it all together, our reparameterized objective function can be written as

$$\begin{aligned} J_0(\alpha) &= R\left(\left\|\sum_{i=1}^{n} \alpha_i x_i\right\|\right) + L\left(s\left(\sum_{i=1}^{n} \alpha_i x_i\right)\right) \\ &= R\left(\sqrt{\alpha^T K \alpha}\right) + L(K\alpha), \end{aligned}$$

  which we minimize over $\alpha \in \mathrm{R}^n$.

- **All information** needed about $x_1, \ldots, x_n$ is summarized in the Gram matrix K.
- We're now minimizing over R^n rather than R^d.
- If $d \gg n$, this can be a big win computationally (at least once K is computed).

Reparameterizing Predictions

- Suppose we've found
$$\alpha^* \in \operatorname*{arg\,min}_{\alpha \in \mathbb{R}^n} R\left(\sqrt{\alpha^T K \alpha}\right) + L(K\alpha).$$
- Then we know $w^* = \sum_{i=1}^n \alpha^* x_i$ is a solution to
$$\operatorname*{arg\,min}_{w \in \mathcal{H}} R(\|w\|) + L(\langle w, x_1 \rangle, \ldots, \langle w, x_n \rangle).$$
- The prediction on a new point $x \in \mathcal{H}$ is
$$\hat{f}(x) = \langle w^*, x \rangle = \sum_{i=1}^n \alpha_i^* \langle x_i, x \rangle.$$
- To make a new prediction, we may need to touch all the training inputs $x_1, \ldots, x_n$.

More Notation

- It will be convenient to define the following column vector for any $x \in \mathcal{H}$:

$$k_x = \begin{pmatrix} \langle x_1, x \rangle \\ \vdots \\ \langle x_n, x \rangle \end{pmatrix}$$

- Then we can write our predictions on a new point x as

$$\hat{f}(x) = k_x^T \alpha^*$$

Summary So Far

- Original plan:
  - Find $w^* \in \arg\min_{w \in \mathcal{H}} R(\|w\|) + L(\langle w, x_1 \rangle, \ldots, \langle w, x_n \rangle)$
  - Predict with $\hat{f}(x) = \langle w^*, x \rangle$.
- We showed that the following is equivalent:
  - Find $\alpha^* \in \arg\min_{\alpha \in \mathbb{R}^n} R\left(\sqrt{\alpha^T K \alpha}\right) + L(K\alpha)$
  - Predict with $\hat{f}(x) = k_x^T \alpha^*$, where

$$K = \begin{pmatrix} \langle x_1, x_1 \rangle & \cdots & \langle x_1, x_n \rangle \\ \vdots & \ddots & \ldots \\ \langle x_n, x_1 \rangle & \cdots & \langle x_n, x_n \rangle \end{pmatrix} \quad \text{and} \quad k_x = \begin{pmatrix} \langle x_1, x \rangle \\ \vdots \\ \langle x_n, x \rangle \end{pmatrix}$$

  - Every element $x \in \mathcal{H}$ occurs inside an inner products with a training input $x_i \in \mathcal{H}$.

Kernelization

Definition

A method is **kernelized** if every feature vector $\psi(x)$ only appears inside an inner product with another feature vector $\psi(x')$. This applies to both the optimization problem and the prediction function.

- Here we are using $\psi(x) = x$. Thus finding

$$\alpha^* \in \underset{\alpha \in \mathbb{R}^n}{\arg\min}\, R\left(\sqrt{\alpha^T K \alpha}\right) + L(K\alpha)$$

and making predictions with $\hat{f}(x) = k_x^T \alpha^*$ is a **kernelization** of finding

$$w^* \in \underset{w \in \mathcal{H}}{\arg\min}\, R(\|w\|) + L(\langle w, x_1 \rangle, \ldots, \langle w, x_n \rangle)$$

and making predictions with $\hat{f}(x) = \langle w^*, x \rangle$.

Summary

- We used duality for SVM and bare hands for ridge regression to find their kernelized version.
- Many other algorithms can be kernelized.
- Our principled tool for kernelization is reparameterization by the representer theorem.
- Representer theorem says that all norm-regularized linear models can be kernelized.
- Once kernelized, we can apply the kernel trick: doesn't need to represent $\phi(x)$ explicitly.