SVM and Kernel Methods

Mengye Ren

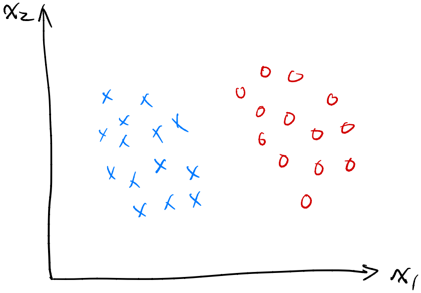
CDS, NYU

September 26, 2023

Maximum Margin Classifier

Linearly Separable Data

Consider a linearly separable dataset $\mathcal{D}$:



Find a separating hyperplane such that

- $w^T x_i > 0$ for all x_i where $y_i = +1$
- $w^T x_i < 0$ for all x_i where $y_i = -1$

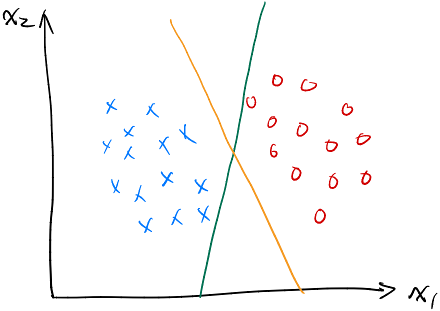
The Perceptron Algorithm

- Initialize $w \leftarrow 0$
- While not converged (exists misclassified examples)
  - For $(x_i, y_i) \in \mathcal{D}$
    - If $y_i w^T x_i < 0$ (wrong prediction)
    - Update $w \leftarrow w + y_i x_i$
- Intuition: move towards misclassified positive examples and away from negative examples
- Guarantees to find a zero-error classifier (if one exists) in finite steps
- What is the loss function if we consider this as a SGD algorithm?

Maximum-Margin Separating Hyperplane

For separable data, there are infinitely many zero-error classifiers.

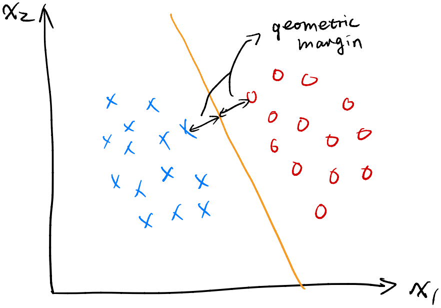
Which one do we pick?



(Perceptron does not return a unique solution.)

Maximum-Margin Separating Hyperplane

We prefer the classifier that is farthest from both classes of points



- Geometric margin: smallest distance between the hyperplane and the points
- Maximum margin: *largest* distance to the closest points

Geometric Margin

We want to maximize the distance between the separating hyperplane and the closest points.

Let's formalize the problem.

Definition (separating hyperplane)

We say (x_i, y_i) for $i = 1, \ldots, n$ are **linearly separable** if there is a $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$ such that $y_i(w^T x_i + b) > 0$ for all i. The set $\{v \in \mathbb{R}^d \mid w^T v + b = 0\}$ is called a **separating hyperplane**.

Definition (geometric margin)

Let H be a hyperplane that separates the data (x_i, y_i) for $i = 1, \ldots, n$. The **geometric margin** of this hyperplane is

$$\min_i d(x_i, H),$$

the distance from the hyperplane to the closest data point.

Distance between a Point and a Hyperplane

- Projection of $v \in \mathbb{R}^d$ onto $w \in \mathbb{R}^d$: $\frac{v \cdot w}{\|w\|_2}$
- Distance between x_i and H:

$$d(x_i, H) = \left| \frac{w^T x_i + b}{\|w\|_2} \right| = \frac{y_i(w^T x_i + b)}{\|w\|_2}$$

Maximize the Margin

We want to maximize the geometric margin:

$$\text{maximize } \min_i d(x_i, H).$$

Given separating hyperplane $H = \left\{v \mid w^T v + b = 0\right\}$, we have

$$\text{maximize } \min_i \frac{y_i(w^T x_i + b)}{\|w\|_2}.$$

Let's remove the inner minimization problem by

$$\begin{array}{ll} \text{maximize} & M \\ \text{subject to} & \frac{y_i(w^T x_i + b)}{\|w\|_2} \geqslant M \quad \text{for all } i \end{array}$$

Note that the solution is not unique (why?).

Maximize the Margin

Let's fix the norm $\|w\|_2$ to $1/M$ to obtain:

$$\begin{array}{ll} \text{maximize} & \frac{1}{\|w\|_2} \\ \text{subject to} & y_i(w^T x_i + b) \geqslant 1 \quad \text{for all } i \end{array}$$

It's equivalent to solving the minimization problem

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|w\|_2^2 \\ \text{subject to} & y_i(w^T x_i + b) \geqslant 1 \quad \text{for all } i \end{array}$$

Note that $y_i(w^T x_i + b)$ is the (functional) margin.

In words, it finds the minimum norm solution which has a margin of at least 1 on all examples.

Soft Margin SVM

What if the data is *not* linearly separable?

For any w, there will be points with a negative margin.

Introduce **slack variables** to penalize small margin:

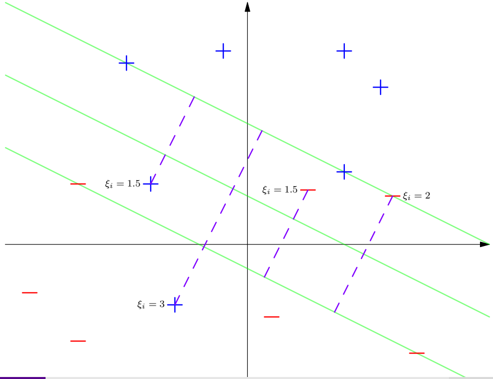
$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|w\|_2^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i \\ \text{subject to} & y_i(w^T x_i + b) \geqslant 1 - \xi_i \quad \text{for all } i \\ & \xi_i \geqslant 0 \quad \text{for all } i \end{array}$$

- If $\xi_i = 0 \; \forall i$, it's reduced to hard SVM.
- What does $\xi_i > 0$ mean?
- What does C control?

Slack Variables

$d(x_i, H) = \frac{y_i(w^T x_i + b)}{\|w\|_2} \geqslant \frac{1-\xi_i}{\|w\|_2}$, thus ξ_i measures the violation by multiples of the geometric margin:

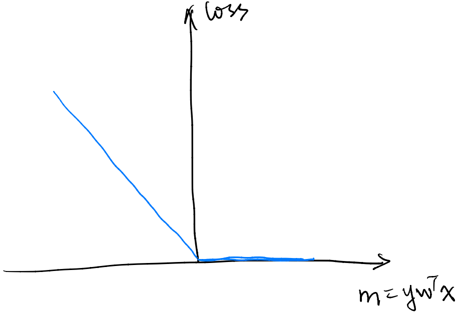
- $\xi_i = 1$: x_i lies on the hyperplane
- $\xi_i = 3$: x_i is past 2 margin width beyond the decision hyperplane



Minimize the Hinge Loss

Perceptron Loss

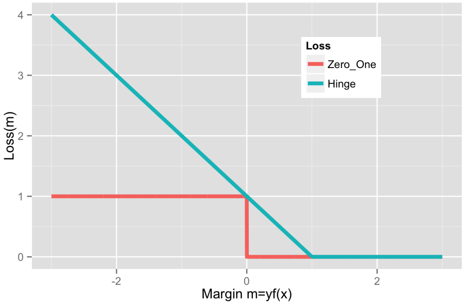
$$\ell(x, y, w) = \max(0, -yw^T x)$$



If we do ERM with this loss function, what happens?

Hinge Loss

- SVM/Hinge loss: $\ell_{\text{Hinge}} = \max\{1-m, 0\} = (1-m)_+$
- Margin $m = yf(x)$; "Positive part" $(x)_+ = x\mathbb{1}(x \geqslant 0)$.



Hinge is a **convex**, **upper bound** on $0-1$ loss. Not differentiable at $m=1$.
We have a **"margin error"** when $m < 1$.

Support Vector Machine

Using ERM:

- Hypothesis space $\mathcal{F} = \left\{ f(x) = w^T x + b \mid w \in \mathbf{R}^d, b \in \mathbf{R} \right\}$.
- ℓ_2 regularization (Tikhonov style)
- Hinge loss $\ell(m) = \max\{1 - m, 0\} = (1 - m)_+$
- The SVM prediction function is the solution to

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

- Not differentiable because of the max

SVM as a Constrained Optimization Problem

- The SVM optimization problem is equivalent to

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|w\|^2 + \frac{c}{n}\sum_{i=1}^{n}\xi_i \\ \text{subject to} & \xi_i \geqslant \max\left(0, 1 - y_i\left[w^Tx_i + b\right]\right) \text{ for } i = 1,\ldots,n. \end{array}$$

- Which is equivalent to

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|w\|^2 + \frac{c}{n}\sum_{i=1}^{n}\xi_i \\ \text{subject to} & \xi_i \geqslant \left(1 - y_i\left[w^Tx_i + b\right]\right) \text{ for } i = 1,\ldots,n \\ & \xi_i \geqslant 0 \text{ for } i = 1,\ldots,n \end{array}$$

Summary

Two ways to derive the SVM optimization problem:

- Maximize the (geometric) margin
- Minimize the hinge loss with ℓ_2 regularization

Both leads to the minimum norm solution satisfying certain margin constraints.

- **Hard-margin SVM**: all points must be correctly classified with the margin constraints
- **Soft-margin SVM**: allow for margin constraint violation with some penalty

Part II: Subgradient Descent for SVM

Now that we have the objective, can we do SGD on it?

Subgradient: generalize gradient for non-differentiable convex functions

SVM Optimization Problem (no intercept)

- SVM objective function:

$$J(w) = \frac{1}{n}\sum_{i=1}^{n} \max\left(0, 1 - y_i w^T x_i\right) + \lambda \|w\|^2.$$

- Not differentiable... but let's think about gradient descent anyway.

- Hinge loss: $\ell(m) = \max(0, 1 - m)$

$$\begin{aligned} \nabla_w J(w) &= \nabla_w \left(\frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i w^T x_i\right) + \lambda \|w\|^2 \right) \\ &= \frac{1}{n}\sum_{i=1}^{n} \nabla_w \ell\left(y_i w^T x_i\right) + 2\lambda w \end{aligned}$$

"Gradient" of SVM Objective

- Derivative of hinge loss $\ell(m) = \max(0, 1-m)$:

$$\ell'(m) = \begin{cases} 0 & m > 1 \\ -1 & m < 1 \\ \text{undefined} & m = 1 \end{cases}$$

- By chain rule, we have

$$\begin{aligned} \nabla_w \ell\left(y_i w^T x_i\right) &= \ell'\left(y_i w^T x_i\right) y_i x_i \\ &= \begin{cases} 0 & y_i w^T x_i > 1 \\ -y_i x_i & y_i w^T x_i < 1 \\ \text{undefined} & y_i w^T x_i = 1 \end{cases} \end{aligned}$$

“Gradient” of SVM Objective

$$\nabla_w \ell\left(y_i w^T x_i\right) = \begin{cases} 0 & y_i w^T x_i > 1 \\ -y_i x_i & y_i w^T x_i < 1 \\ \text{undefined} & y_i w^T x_i = 1 \end{cases}$$

So

$$\begin{aligned} \nabla_w J(w) &= \nabla_w \left(\frac{1}{n} \sum_{i=1}^{n} \ell\left(y_i w^T x_i\right) + \lambda \|w\|^2 \right) \\ &= \frac{1}{n} \sum_{i=1}^{n} \nabla_w \ell\left(y_i w^T x_i\right) + 2\lambda w \\ &= \begin{cases} \frac{1}{n} \sum_{i: y_i w^T x_i < 1} \left(-y_i x_i\right) + 2\lambda w & \text{all } y_i w^T x_i \neq 1 \\ \text{undefined} & \text{otherwise} \end{cases} \end{aligned}$$

Gradient Descent on SVM Objective?

- The gradient of the SVM objective is

$$\nabla_w J(w) = \frac{1}{n} \sum_{i: y_i w^T x_i < 1} (-y_i x_i) + 2\lambda w$$

when $y_i w^T x_i \neq 1$ for all i, and otherwise is undefined.

Potential arguments for why we shouldn't care about the points of nondifferentiability:

- If we start with a random w, will we ever hit exactly $y_i w^T x_i = 1$?
- If we did, could we perturb the step size by ε to miss such a point?
- Does it even make sense to check $y_i w^T x_i = 1$ with floating point numbers?

However, would gradient descent work if the objective is not differentiable?

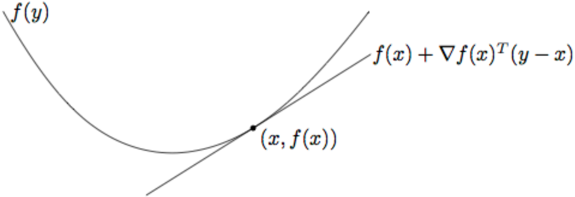
Subgradient

First-Order Condition for Convex, Differentiable Function

- Suppose $f : \mathrm{R}^d \to \mathrm{R}$ is convex and differentiable Then for any $x, y \in \mathrm{R}^d$

$$f(y) \geqslant f(x) + \nabla f(x)^T (y - x)$$

- The linear approximation to f at x is a global underestimator of f:



- This implies that if $\nabla f(x) = 0$ then x is a global minimizer of f.

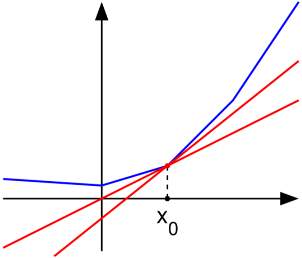
Figure from Boyd & Vandenberghe Fig. 3.2; Proof in Section 3.1.3

Subgradients

Definition

A vector $g \in \mathbb{R}^d$ is a **subgradient** of a *convex* function $f : \mathbb{R}^d \to \mathbb{R}$ at x if for all z,

$$f(z) \geqslant f(x) + g^T(z - x).$$



Blue is a graph of $f(x)$.
Each red line $x \mapsto f(x_0) + g^T(x - x_0)$ is a global lower bound on $f(x)$.

Properties

Definitions

- The set of all subgradients at x is called the **subdifferential**: $\partial f(x)$
- f is **subdifferentiable** at x if $\exists$ at least one subgradient at x.

For convex functions:

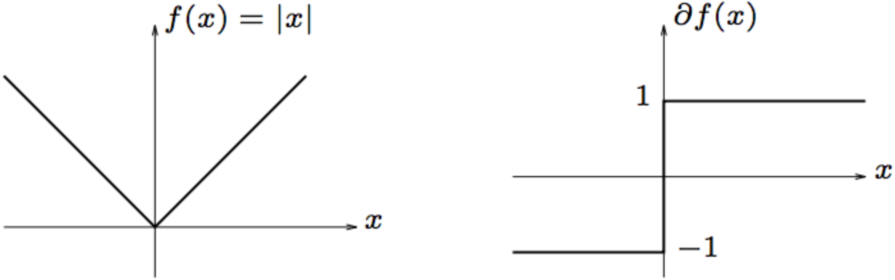
- f is differentiable at x iff $\partial f(x) = \{\nabla f(x)\}$.
- Subdifferential is always non-empty ($\partial f(x) = \emptyset \implies f$ is not convex)
- x is the global optimum iff $0 \in \partial f(x)$.

For non-convex functions:

- The subdifferential may be an empty set (no global underestimator).

Subdifferential of Absolute Value

- Consider $f(x) = |x|$

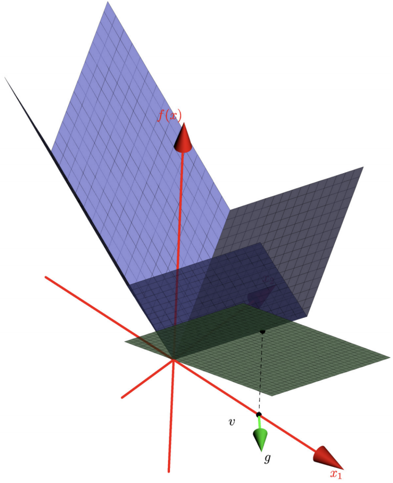


- Plot on right shows $\{(x, g) \mid x \in \mathbf{R},\ g \in \partial f(x)\}$

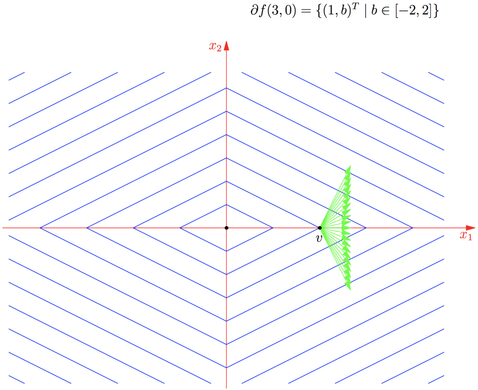
Boyd EE364b: Subgradients Slides

Subgradients of $f(x_1, x_2) = |x_1| + 2|x_2|$

- Let's find the subdifferential of $f(x_1, x_2) = |x_1| + 2|x_2|$ at $(3, 0)$.
- First coordinate of subgradient must be 1, from $|x_1|$ part (at $x_1 = 3$).
- Second coordinate of subgradient can be anything in $[-2, 2]$.
- So graph of $h(x_1, x_2) = f(3, 0) + g^T(x_1 - 3, x_2 - 0)$ is a global underestimate of $f(x_1, x_2)$, for any $g = (g_1, g_2)$, where $g_1 = 1$ and $g_2 \in [-2, 2]$.



Subdifferential on Contour Plot



Contour plot of $f(x_1, x_2) = |x_1| + 2|x_2|$, with set of subgradients at $(3,0)$. .

Plot courtesy of Brett Bernstein.

Basic Rules for Calculating Subdifferential

- Non-negative scaling: $\partial \alpha f(x) = \alpha \partial f(x)$ for $(\alpha > 0)$
- Summation: $\partial (f_1(x) + f_2(x)) = d_1 + d_2$ for any $d_1 \in \partial f_1$ and $d_2 \in \partial f_2$
- Composing with affine functions: $\partial f(Ax + b) = A^T \partial f(z)$ where $z = Ax + b$
- max: convex combinations of argmax gradients

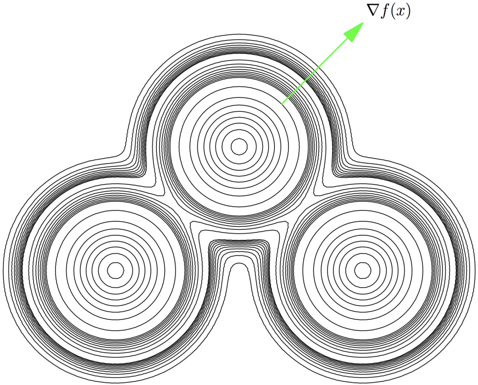
$$\partial \max(f_1(x), f_2(x)) = \begin{cases} \nabla f_1(x) & \text{if } f_1(x) > f_2(x), \\ \nabla f_2(x) & \text{if } f_1(x) < f_2(x), \\ \nabla \theta f_1(x) + (1 - \theta) f_2(x) & \text{if } f_1(x) = f_2(x), \end{cases}$$

where $\theta \in [0, 1]$.

Subgradient Descent

Gradient orthogonal to level sets

We know that gradient points to the fastest ascent direction. What about subgradients?



Plot courtesy of Brett Bernstein.

Contour Lines and Subgradients

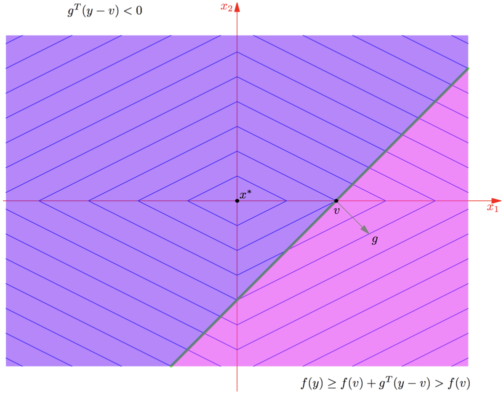
A hyperplane H **supports** a set S if H intersects S and all of S lies one one side of H.

Claim: If $f : \mathbf{R}^d \to \mathbf{R}$ has subgradient g at x_0, then the hyperplane H orthogonal to g at x_0 must **support** the level set $S = \left\{x \in \mathbf{R}^d \mid f(x) = f(x_0)\right\}$.

Proof:

- For any y, we have $f(y) \geqslant f(x_0) + g^T(y - x_0)$. (def of subgradient)
- If y is strictly on side of H that g points in,
  - then $g^T(y - x_0) > 0$.
  - So $f(y) > f(x_0)$.
  - So y is not in the level set S.
- $\therefore$ All elements of S must be on H or on the $-g$ side of H.

Subgradient of $f(x_1, x_2) = |x_1| + 2|x_2|$



- Points on g side of H have larger f-values than $f(x_0)$. (from proof)
- But points on $-g$ side may **not** have smaller f-values.
- So $-g$ may **not** be a descent direction. (shown in figure)

Plot courtesy of Brett Bernstein.

Subgradient Descent

- Move along the negative subgradient:

$$x^{t+1} = x^t - \eta g \quad \text{where } g \in \partial f(x^t) \text{ and } \eta > 0$$

- This can increase the objective but gets us closer to the minimizer if f is convex and η is small enough:

$$\|x^{t+1} - x^*\| < \|x^t - x^*\|$$

- Subgradients don't necessarily converge to zero as we get closer to x^*, so we need decreasing step sizes, e.g. $O(1/t)$ or $O(1/\sqrt{t})$.

- Subgradient methods are slower than gradient descent, e.g. $O(1/\epsilon^2)$ vs $O(1/\epsilon)$ for convex functions.

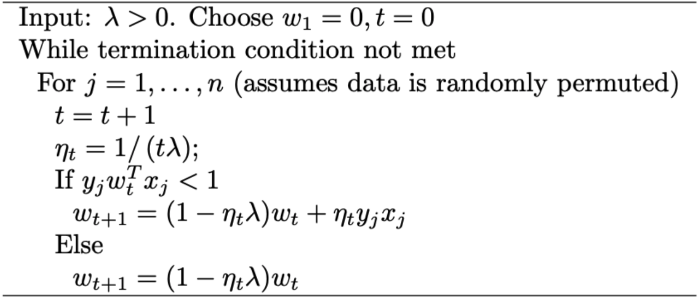
Based on https://www.cs.ubc.ca/~schmidtm/Courses/5XX-S20/S4.pdf

Subgradient descent for SVM (HW3)

SVM objective function:

$$J(w) = \frac{1}{n}\sum_{i=1}^{n} \max\left(0, 1 - y_i w^T x_i\right) + \lambda \|w\|^2.$$

Pegasos: stochastic subgradient descent with step size $\eta_t = 1/(t\lambda)$



```
Input: $\lambda > 0$. Choose $w_1 = 0, t = 0$
While termination condition not met
  For $j = 1, \ldots, n$ (assumes data is randomly permuted)
    $t = t + 1$
    $\eta_t = 1/(t\lambda)$;
    If $y_j w_t^T x_j < 1$
      $w_{t+1} = (1 - \eta_t\lambda) w_t + \eta_t y_j x_j$
    Else
      $w_{t+1} = (1 - \eta_t\lambda) w_t$
```

Summary

- Subgradient: generalize gradient for non-differentiable convex functions
- Subgradient “descent”:
  - General method for non-smooth functions
  - Simple to implement
  - Slow to converge

Part III: The Dual Problem

In addition to subgradient descent, we can directly solve the optimization problem using a QP solver.

Let's study its dual problem to gain addition insights (which will be useful for next week!)

SVM as a Quadratic Program

- The SVM optimization problem is equivalent to

$$\begin{array}{ll}\text{minimize} & \frac{1}{2}||w||^2 + \frac{c}{n}\sum_{i=1}^{n}\xi_i \\ \text{subject to} & -\xi_i \leqslant 0 \quad \text{for } i=1,\dots,n \\ & \left(1-y_i\left[w^Tx_i+b\right]\right)-\xi_i \leqslant 0 \quad \text{for } i=1,\dots,n\end{array}$$

- Differentiable objective function
- $n+d+1$ unknowns and $2n$ affine constraints.
- A **quadratic program** that can be solved by any off-the-shelf QP solver.
- Let's learn more by examining the dual.

Why Do We Care About the Dual?

The Lagrangian

The general [inequality-constrained] optimization problem is:

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leqslant 0, \quad i = 1, \ldots, m \end{array}$$

Definition

The **Lagrangian** for this optimization problem is

$$L(x, \lambda) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x).$$

- λ_i's are called **Lagrange multipliers** (also called the **dual variables**).
- Weighted sum of the objective and constraint functions
- Hard constraints → soft constraints

Lagrange Dual Function

Definition

The **Lagrange dual function** is

$$g(\lambda) = \inf_x L(x, \lambda) = \inf_x \left(f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) \right)$$

- $g(\lambda)$ is concave
- **Lower bound property**: if $\lambda \succeq 0$, $g(\lambda) \leqslant p^*$ where p^* is the optimal value of the optimization problem.
- $g(\lambda)$ can be $-\infty$ (uninformative lower bound)

The Primal and the Dual

- For any **primal form** optimization problem,

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leqslant 0, \;\; i = 1, \ldots, m, \end{array}$$

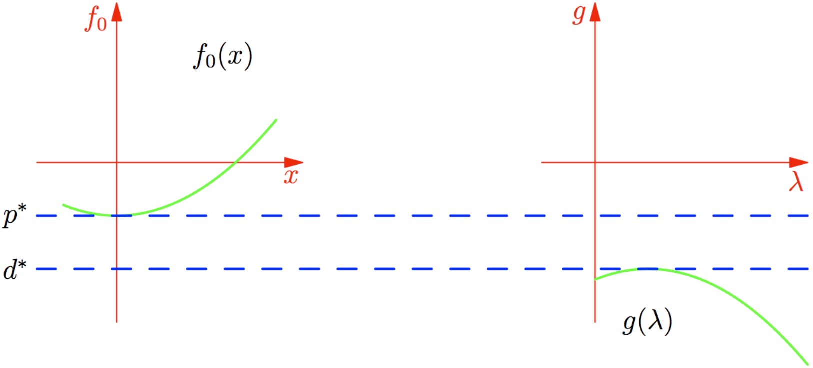
  there is a recipe for constructing a corresponding **Lagrangian dual problem:**

$$\begin{array}{ll} \text{maximize} & g(\lambda) \\ \text{subject to} & \lambda_i \geqslant 0, \;\; i = 1, \ldots, m, \end{array}$$

- The dual problem is always a convex optimization problem.
- The dual variables often have interesting and relevant interpretations.
- The dual variables provide certificates for optimality.

Weak Duality

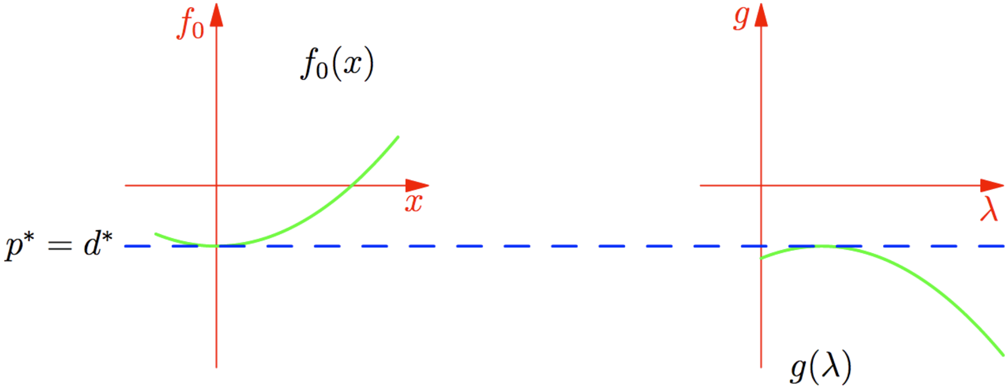
We always have **weak duality:** $p^* \geqslant d^*$.



Plot courtesy of Brett Bernstein.

Strong Duality

For some problems, we have **strong duality**: $p^* = d^*$.



For convex problems, strong duality is fairly typical.

Plot courtesy of Brett Bernstein.

Complementary Slackness

- Assume strong duality. Let x^* be primal optimal and λ^* be dual optimal. Then:

$$\begin{aligned} f_0(x^*) &= g(\lambda^*) = \inf_x L(x,\lambda^*) \quad \text{(strong duality and definition)} \\ &\leqslant L(x^*,\lambda^*) \\ &= f_0(x^*) + \sum_{i=1}^m \lambda_i^* f_i(x^*) \\ &\leqslant f_0(x^*). \end{aligned}$$

Each term in sum $\sum_{i=1} \lambda_i^* f_i(x^*)$ must actually be 0. That is

$$\lambda_i > 0 \implies f_i(x^*) = 0 \quad \text{and} \quad f_i(x^*) < 0 \implies \lambda_i = 0 \quad \forall i$$

This condition is known as **complementary slackness**.

The SVM Dual Problem

SVM Lagrange Multipliers

$$\begin{array}{ll}\text{minimize} & \frac{1}{2}||w||^2 + \frac{c}{n}\sum_{i=1}^{n}\xi_i \\ \text{subject to} & -\xi_i \leqslant 0 \quad \text{for } i = 1, \ldots, n \\ & \left(1 - y_i\left[w^T x_i + b\right]\right) - \xi_i \leqslant 0 \quad \text{for } i = 1, \ldots, n\end{array}$$

Lagrange Multiplier	Constraint
λ_i	$-\xi_i \leqslant 0$
α_i	$\left(1 - y_i\left[w^T x_i + b\right]\right) - \xi_i \leqslant 0$

$$L(w, b, \xi, \alpha, \lambda) = \frac{1}{2}||w||^2 + \frac{c}{n}\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i\left(1 - y_i\left[w^T x_i + b\right] - \xi_i\right) + \sum_{i=1}^{n}\lambda_i\left(-\xi_i\right)$$

Dual optimum value: $d^* = \sup_{\alpha,\lambda \succeq 0} \inf_{w,b,\xi} L(w, b, \xi, \alpha, \lambda)$

Strong Duality by Slater's Constraint Qualification

The SVM optimization problem:

$$\begin{array}{ll}\text{minimize} & \frac{1}{2}\|w\|^2+\frac{c}{n}\sum_{i=1}^{n}\xi_i \\ \text{subject to} & -\xi_i \leqslant 0 \text{ for } i=1,\ldots,n \\ & \left(1-y_i\left[w^Tx_i+b\right]\right)-\xi_i \leqslant 0 \text{ for } i=1,\ldots,n\end{array}$$

Slater's constraint qualification:

- Convex problem + affine constraints $\Longrightarrow$ strong duality iff problem is feasible
- Do we have a feasible point?
- For SVM, we have strong duality.

SVM Dual Function: First Order Conditions

Lagrange dual function is the inf over primal variables of L:

$$\begin{aligned}
g(\alpha,\lambda) &= \inf_{w,b,\xi} L(w,b,\xi,\alpha,\lambda) \\
&= \inf_{w,b,\xi} \left[\frac{1}{2}w^T w + \sum_{i=1}^{n} \xi_i \left(\frac{c}{n} - \alpha_i - \lambda_i\right) + \sum_{i=1}^{n} \alpha_i \left(1 - y_i \left[w^T x_i + b\right]\right)\right]
\end{aligned}$$

$$\partial_w L = 0 \iff w - \sum_{i=1}^{n} \alpha_i y_i x_i = 0 \iff \boxed{w = \sum_{i=1}^{n} \alpha_i y_i x_i}$$

$$\partial_b L = 0 \iff -\sum_{i=1}^{n} \alpha_i y_i = 0 \iff \boxed{\sum_{i=1}^{n} \alpha_i y_i = 0}$$

$$\partial_{\xi_i} L = 0 \iff \frac{c}{n} - \alpha_i - \lambda_i = 0 \iff \boxed{\alpha_i + \lambda_i = \frac{c}{n}}$$

SVM Dual Function

- Substituting these conditions back into L, the second term disappears.
- First and third terms become

$$\begin{aligned}\frac{1}{2}w^Tw &= \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j \\ \sum_{i=1}^{n}\alpha_i(1-y_i\left[w^Tx_i+b\right]) &= \sum_{i=1}^{n}\alpha_i-\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j x_j^T x_i - b\underbrace{\sum_{i=1}^{n}\alpha_i y_i}_{=0}.\end{aligned}$$

- Putting it together, the dual function is

$$g(\alpha,\lambda)=\begin{cases}\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j x_j^T x_i & \substack{\sum_{i=1}^{n}\alpha_i y_i=0 \\ \alpha_i+\lambda_i=\frac{c}{n}}, \text{ all } i \\ -\infty & \text{otherwise.}\end{cases}$$

SVM Dual Problem

- The **dual function** is

$$g(\alpha, \lambda) = \begin{cases} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_j^T x_i & \substack{\sum_{i=1}^{n} \alpha_i y_i = 0 \\ \alpha_i + \lambda_i = \frac{c}{n}, \text{ all } i} \\ -\infty & \text{otherwise.} \end{cases}$$

- The **dual problem** is $\sup_{\alpha, \lambda \succeq 0} g(\alpha, \lambda)$:

$$\begin{aligned} \sup_{\alpha, \lambda} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_j^T x_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \\ & \alpha_i + \lambda_i = \frac{c}{n} \quad \alpha_i, \lambda_i \geqslant 0, \ i = 1, \ldots, n \end{aligned}$$

Insights from the Dual Problem

KKT Conditions

For convex problems, if Slater's condition is satisfied, then **KKT conditions** provide necessary and sufficient conditions for the optimal solution.

- Primal feasibility: $f_i(x) \leqslant 0 \quad \forall i$
- Dual feasibility: $\lambda \succeq 0$
- Complementary slackness: $\lambda_i f_i(x) = 0$
- First-order condition:

$$\frac{\partial}{\partial x} L(x, \lambda) = 0$$

The SVM Dual Solution

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

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

- Given solution α^* to dual, primal solution is $w^* = \sum_{i=1}^{n} \alpha_i^* y_i x_i$.
- The solution is in the space spanned by the inputs.
- Note $\alpha_i^* \in [0, \frac{c}{n}]$. So c controls max weight on each example. (**Robustness**!)
    - What's the relation between c and regularization?

Complementary Slackness Conditions

- Recall our primal constraints and Lagrange multipliers:

Lagrange Multiplier	Constraint
λ_i	$-\xi_i \leqslant 0$
α_i	$(1 - y_i f(x_i)) - \xi_i \leqslant 0$

- Recall first order condition $\nabla_{\xi_i} L = 0$ gave us $\lambda_i^* = \frac{c}{n} - \alpha_i^*$.
- By strong duality, we must have **complementary slackness**:

$$\alpha_i^* \left(1 - y_i f^*(x_i) - \xi_i^*\right) = 0$$

$$\lambda_i^* \xi_i^* = \left(\frac{c}{n} - \alpha_i^*\right) \xi_i^* = 0$$

Consequences of Complementary Slackness

By strong duality, we must have **complementary slackness**.

$$\begin{aligned}\alpha_i^*\left(1-y_i f^*(x_i)-\xi_i^*\right) &= 0\\ \left(\frac{c}{n}-\alpha_i^*\right)\xi_i^* &= 0\end{aligned}$$

Recall "**slack variable**" $\xi_i^* = \max\left(0, 1-y_i f^*(x_i)\right)$ is the hinge loss on (x_i, y_i).

- If $y_i f^*(x_i) > 1$ then the margin loss is $\xi_i^* = 0$, and we get $\alpha_i^* = 0$.
- If $y_i f^*(x_i) < 1$ then the margin loss is $\xi_i^* > 0$, so $\alpha_i^* = \frac{c}{n}$.
- If $\alpha_i^* = 0$, then $\xi_i^* = 0$, which implies no loss, so $y_i f^*(x) \geqslant 1$.
- If $\alpha_i^* \in \left(0, \frac{c}{n}\right)$, then $\xi_i^* = 0$, which implies $1-y_i f^*(x_i) = 0$.

Complementary Slackness Results: Summary

If α^* is a solution to the dual problem, then primal solution is

$$w^* = \sum_{i=1}^{n} \alpha_i^* y_i x_i \quad \text{where} \alpha_i^* \in [0, \frac{c}{n}].$$

Relation between margin and example weights (α_i's):

$$\begin{aligned}
\alpha_i^* = 0 &\implies y_i f^*(x_i) \geqslant 1 \\
\alpha_i^* \in \left(0, \frac{c}{n}\right) &\implies y_i f^*(x_i) = 1 \\
\alpha_i^* = \frac{c}{n} &\implies y_i f^*(x_i) \leqslant 1
\end{aligned}$$

$$\begin{aligned}
y_i f^*(x_i) < 1 &\implies \alpha_i^* = \frac{c}{n} \\
y_i f^*(x_i) = 1 &\implies \alpha_i^* \in \left[0, \frac{c}{n}\right] \\
y_i f^*(x_i) > 1 &\implies \alpha_i^* = 0
\end{aligned}$$

Support Vectors

- If α^* is a solution to the dual problem, then primal solution is

$$w^* = \sum_{i=1}^{n} \alpha_i^* y_i x_i$$

with $\alpha_i^* \in [0, \frac{c}{n}]$.

- The x_i's corresponding to $\alpha_i^* > 0$ are called **support vectors.**
- Few margin errors or "on the margin" examples $\Longrightarrow$ sparsity in input examples.

Teaser for Kernelization

Dual Problem: Dependence on x through inner products

- SVM Dual Problem:

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

- Note that all dependence on inputs x_i and x_j is through their inner product: $\langle x_j, x_i \rangle = x_j^T x_i$.
- We can replace $x_j^T x_i$ by other products...
- This is a "kernelized" objective function.

Feature Maps

The Input Space $\mathcal{X}$

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

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

- What if we want to do prediction on inputs not natively in $\mathbf{R}^d$?

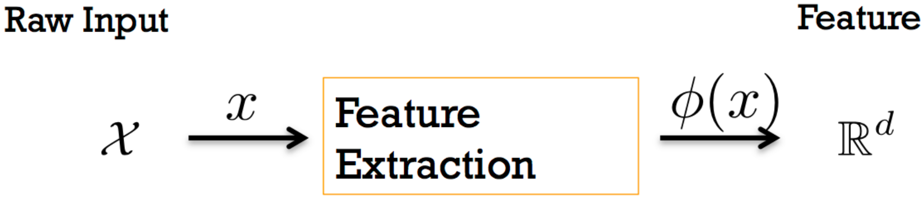
The Input Space $\mathcal{X}$

- 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**.



Raw Input

Feature Vector

$$\mathcal{X} \xrightarrow{x} \boxed{\text{Feature Extraction}} \xrightarrow{\phi(x)} \mathbb{R}^d$$

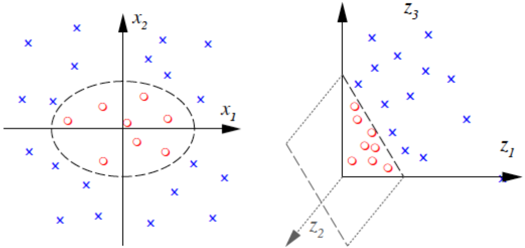
Linear Models with Explicit Feature Map

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

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

Geometric Example: Two class problem, nonlinear boundary

$$\Phi : R^2 \to R^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{(2)}x_1x_2, x_2^2)$$



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

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

- 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:

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

- If α^* is an optimal value, then

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

- 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

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

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

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

(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. $\mathbf{R}^d$)
- **Feature map**: $\psi : \mathcal{X} \to \mathcal{H}$
- The **kernel function** corresponding to ψ is

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

  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:

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

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 $x_1, \dots, x_n \in \mathcal{X}$ is

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

- 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}:

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

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 $\psi(x) : \mathcal{X} \to \mathrm{R}^d$ (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

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

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

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

- 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: $\mathcal{X} = \mathsf{R}^d$
- Feature space: $\mathcal{H} = \mathsf{R}^d$, 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:

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

- 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 $\mathcal{X} = \mathbf{R}^d$

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

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:

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

- 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, $w^* \in \operatorname{span}(x_1, \ldots, x_n)$.

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. $w^* \in \operatorname{span}(x_1, \ldots, x_n)$

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

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

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

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

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

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

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

- 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

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

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:

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

- **Generalized objective**:

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

  where

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

General Objective Function for Linear Hypothesis Space (Details)

- **Generalized objective**:

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

- 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**:

$$w^* = \operatorname*{arg\,min}_{w \in \mathcal{H}} R\left(\|w\|\right) + L\left(\langle w, x_1 \rangle, \dots, \langle w, x_n \rangle\right)$$

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

$$w^* = \operatorname*{arg\,min}_{w \in \operatorname{span}(x_1, \dots, x_n)} R\left(\|w\|\right) + L\left(\langle w, x_1 \rangle, \dots, \langle w, x_n \rangle\right).$$

- So we can reparameterize as before:

$$\alpha^* = \operatorname*{arg\,min}_{\alpha \in \mathbf{R}^n} R\left(\left\|\sum_{i=1}^n \alpha_i x_i\right\|\right) + L\left(\left\langle \sum_{i=1}^n \alpha_i x_i, x_1 \right\rangle, \dots, \left\langle \sum_{i=1}^n \alpha_i x_i, x_n \right\rangle\right).$$

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

The Representer Theorem

Theorem (Representer Theorem)

Let

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

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**).*

*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 $s : \mathbf{R}^d \to \mathbf{R}^n$ 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

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

  where now $L : \mathbf{R}^{n \times 1} \to \mathbf{R}$ 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 $\alpha = (\alpha_1, \ldots, \alpha_n)^T \in \mathrm{R}^{n \times 1}$.

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

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 $x_1, \ldots, x_n \in \mathrm{R}^{d \times 1}$ with the standard inner product $\langle x, x' \rangle = x^T x'$.
- Let $X \in \mathrm{R}^{n \times d}$ 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

$$\alpha^* \in \operatorname*{arg\,min}_{\alpha \in \mathbf{R}^n} R\left(\sqrt{\alpha^T K \alpha}\right) + L\left(K\alpha\right).$$

- Then we know $w^* = \sum_{i=1}^n \alpha^* x_i$ is a solution to

$$\operatorname*{arg\,min}_{w \in \mathcal{H}} R\left(\|w\|\right) + L\left(\langle w, x_1\rangle, \dots, \langle w, x_n\rangle\right).$$

- 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 $x_1, \dots, x_n$.

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 $\alpha^* \in \arg\min_{\alpha \in \mathbf{R}^n} R\left(\sqrt{\alpha^T K \alpha}\right) + L(K\alpha)$
    - 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

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

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

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

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.