

SVM and Kernel Methods

Mengye Ren

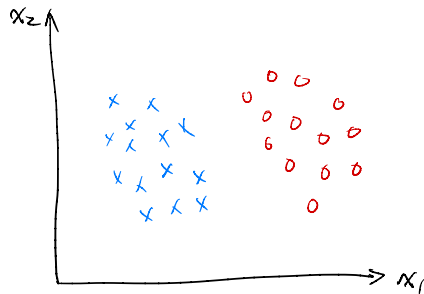
NYU

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 $y_i = -1$

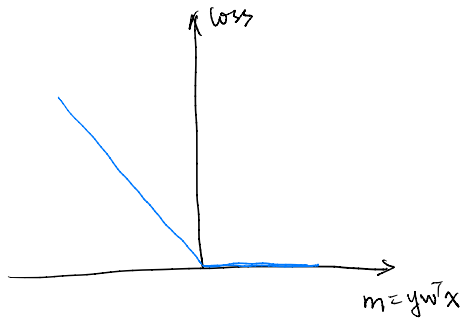
The Perceptron Algorithm

- 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?

Minimize the Hinge Loss

Perceptron Loss

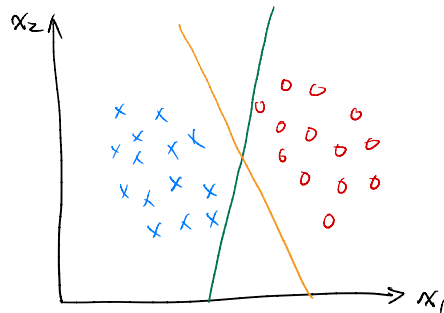
$$\ell(x, y, w) = \max(0, -yw^T x)$$



Maximum-Margin Separating Hyperplane

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

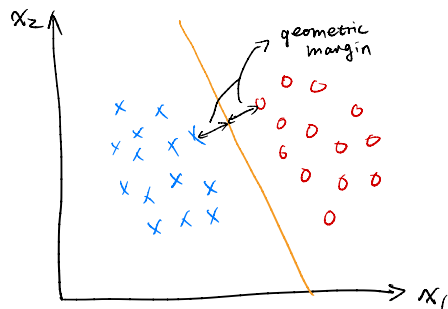
Which one do we pick?



(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 $i = 1, \dots, n$ 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**.

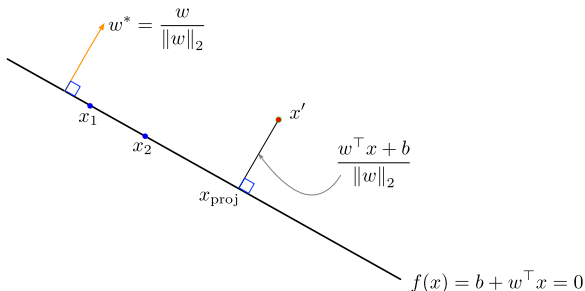
Definition (geometric margin)

Let H be a hyperplane that separates the data (x_i, y_i) for $i = 1, \dots, n$. 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



- Any point on the plane p , and normal vector $w/\|w\|_2$
- Projection of x onto the normal: $\frac{(x'-p)^T w}{\|w\|_2}$
- $(x' - p)^T w = x'^T w - p^T w = x'^T w + b$ (since $p^T w + b = 0$)
- Signed distance between x' and Hyperplane H : $\frac{w^T x' + b}{\|w\|_2}$
- Taking into account of the label y :
$$d(x', H) = \frac{y(w^T x' + b)}{\|w\|_2}$$

Maximize the Margin

We want to maximize the geometric margin:

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

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

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

Let's remove the inner minimization problem by

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

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

Maximize the Margin

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

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

It's equivalent to solving the minimization problem

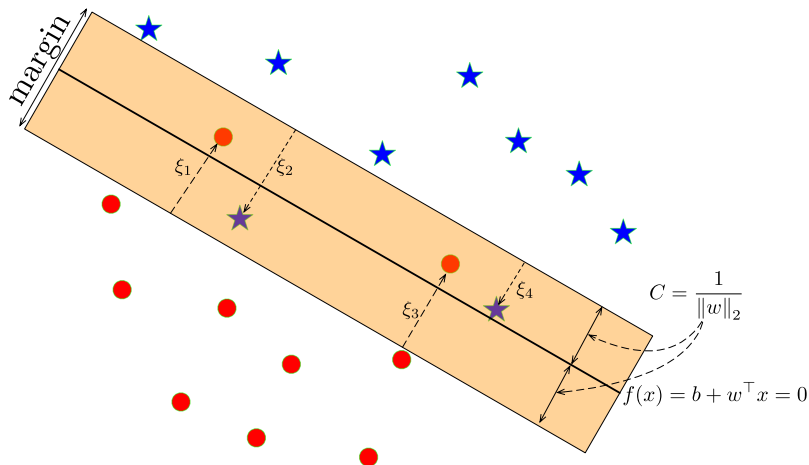
$$\begin{array}{ll}\text{minimize} & \frac{1}{2} \|w\|_2^2 \\ \text{subject to} & y_i(w^T x_i + b) \geq 1 \quad \text{for all } i\end{array}$$

Note that $y_i(w^T x_i + b)$ is the (functional) margin. The optimization finds the minimum norm solution which has a margin of at least 1 on all examples.

Not linearly separable

What if the data is *not* linearly separable?

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



Introduce **slack variables** ξ 's to penalize small margin:

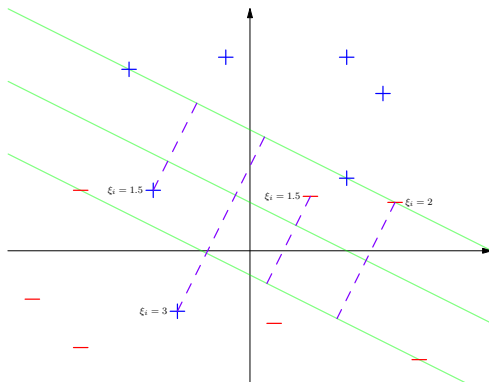
$$\begin{aligned} & \text{minimize} && \frac{1}{2} \|w\|_2^2 + \frac{C}{n} \sum_{i=1}^n \xi_i \\ & \text{subject to} && y_i(w^T x_i + b) \geq 1 - \xi_i \quad \text{for all } i \\ & && \xi_i \geq 0 \quad \text{for all } i \end{aligned}$$

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

Slack Variables

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

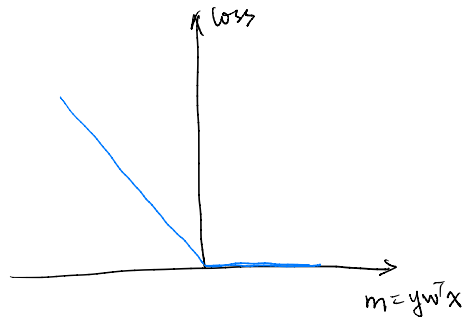
- $\xi_i = 1$: x_i lies on the hyperplane
- $\xi_i = 3$: x_i is past 2 margin width beyond the decision hyperplane



Minimize the Hinge Loss

Perceptron Loss

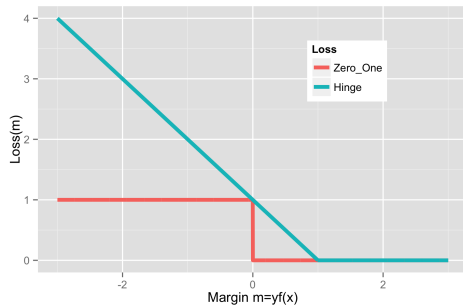
$$\ell(x, y, w) = \max(0, -yw^T x)$$



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” $(x)_+ = x\mathbb{1}[x \geq 0]$.



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

SVM as an Optimization Problem

- The SVM optimization problem is equivalent to

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\|w\|^2 + \frac{c}{n} \sum_{i=1}^n \xi_i \\ &\text{subject to} && \xi_i \geq (1 - y_i [w^T x_i + b]) \text{ for } i = 1, \dots, n \\ &&& \xi_i \geq 0 \text{ for } i = 1, \dots, n \end{aligned}$$

which is equivalent to

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\|w\|^2 + \frac{c}{n} \sum_{i=1}^n \xi_i \\ &\text{subject to} && \xi_i \geq \max(0, 1 - y_i [w^T x_i + b]) \text{ for } i = 1, \dots, n. \end{aligned}$$

SVM as an Optimization Problem

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} \|w\|^2 + \frac{c}{n} \sum_{i=1}^n \xi_i \\ \text{subject to} \quad & \xi_i \geq \max(0, 1 - y_i [w^T x_i + b]) \text{ for } i = 1, \dots, n. \end{aligned}$$

Move the constraint into the objective:

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

- The first term is the L2 regularizer.
- The second term is the Hinge loss.

Support Vector Machine

Using ERM:

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

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

- **Not differentiable** because of the max

Two ways to derive the SVM optimization problem:

- Maximize the 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

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:

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

- Not differentiable... but let's think about gradient descent anyway.
- Hinge loss: $\ell(m) = \max(0, 1 - m)$

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

“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

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

“Gradient” of SVM Objective

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

So

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

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?

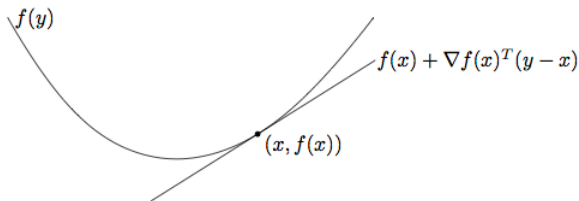
Subgradient

First-Order Condition for Convex, Differentiable Function

- Suppose $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is **convex** and **differentiable**. Then for any $x, y \in \mathbb{R}^d$

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

- 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 .

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**.
- Subgradient methods are **slower** than gradient descent.

Subgradient descent for SVM

SVM objective function:

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

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 $j = 1, \dots, n$ (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

$w_{t+1} = (1 - \eta_t \lambda) w_t$

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

The Dual Problem

- In addition to subgradient descent, we can directly solve the optimization problem using a QP solver.
- For convex optimization problem, we can also look into its **dual problem**.

The Lagrangian

The general [inequality-constrained] optimization problem is:

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

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 \rightarrow soft penalty (objective function)

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 $\lambda \succeq 0$, $g(\lambda) \leq p^*$ 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,

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

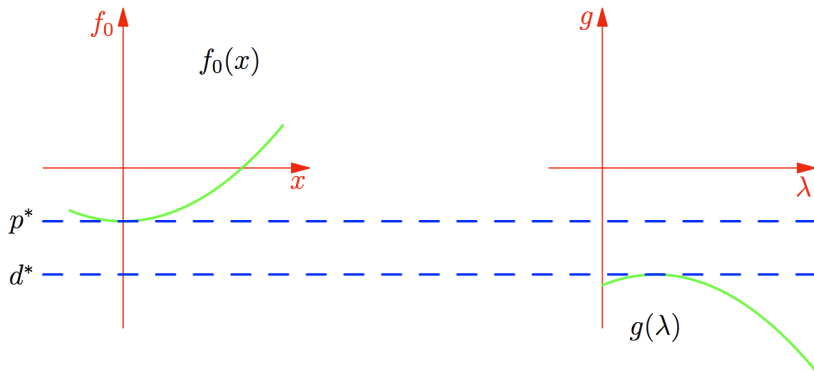
there is a recipe for constructing a corresponding **Lagrangian dual problem**:

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

- The dual problem is always a convex optimization problem.

Weak Duality

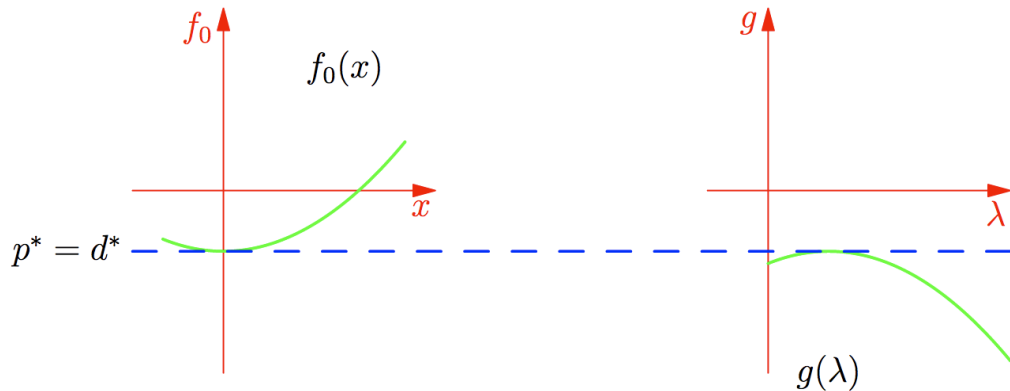
We always have **weak duality**: $p^* \geq d^*$.



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:

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

Each term in sum $\sum_{i=1}^m \lambda_i^* f_i(x^*)$ 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

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

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

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

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:

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

Slater's constraint qualification:

- Convex problem + affine constraints \implies 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 :

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

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

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

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

SVM Dual Function

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

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

- Putting it together, the dual function is

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

SVM Dual Problem

- The **dual function** is

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

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

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

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: $f_i(x) \leq 0 \quad \forall i$
- 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:

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

- 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	$-\xi_i \leq 0$
α_i	$(1 - y_i f(x_i)) - \xi_i \leq 0$

- 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**:

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

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

Consequences of Complementary Slackness

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

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

Recall “**slack variable**” $\xi_i^* = \max(0, 1 - y_i f^*(x_i))$ 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 $y_i f^*(x) \geq 1$.
- If $\alpha_i^* \in (0, \frac{c}{n})$, 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):

$$\alpha_i^* = 0 \implies y_i f^*(x_i) \geq 1$$

$$\alpha_i^* \in (0, \frac{c}{n}) \implies y_i f^*(x_i) = 1$$

$$\alpha_i^* = \frac{c}{n} \implies y_i f^*(x_i) \leq 1$$

$$y_i f^*(x_i) < 1 \implies \alpha_i^* = \frac{c}{n}$$

$$y_i f^*(x_i) = 1 \implies \alpha_i^* \in [0, \frac{c}{n}]$$

$$y_i f^*(x_i) > 1 \implies \alpha_i^* = 0$$

- 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 \implies **sparsity in input examples**.

Kernelization

Dual Problem: Dependence on x through inner products

- SVM Dual Problem:

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

- 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 $\mathcal{X} = \mathbb{R}^d$ for the specific methods we've developed:
 - Ridge regression
 - Lasso regression
 - Support Vector Machines
- Our hypothesis space for these was all affine functions on \mathbb{R}^d :

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

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

The Input Space \mathcal{X}

- Often want to use inputs not natively in \mathbb{R}^d :
 - Text documents
 - Image files
 - Sound recordings
 - DNA sequences
- But everything in a computer is a sequence of numbers
 - The i th entry of each sequence should have the same “meaning”
 - All the sequences should have the same length

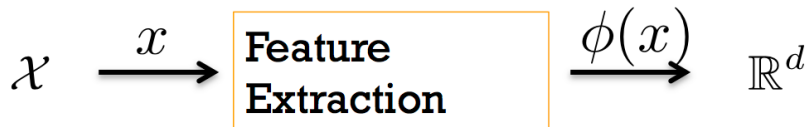
Feature Extraction

Definition

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

Raw Input

Feature Vector

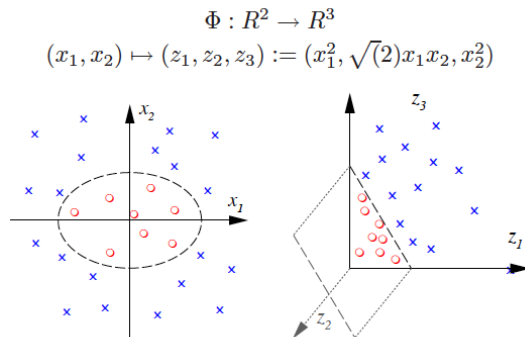


Linear Models with Explicit Feature Map

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

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

Geometric Example: Two class problem, nonlinear boundary



- 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 $\mathcal{F} = \{\text{affine functions of temperature}\}$
- 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.

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?

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.

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**.

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):

$$\phi(x) = (\mathbb{1}[0 \leq N(x) < 10], \mathbb{1}[10 \leq N(x) < 100], \dots)$$

- Small buckets allow quite flexible relationship

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.

Interactions: Approach 1

- Google “ideal weight from height”
- J. D. Robinson’s “ideal weight” formula (for a male):

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

- Make score square deviation between $\text{height}(h)$ and ideal weight(w)

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

- WolframAlpha for complicated Mathematics:

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

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.

Monomial Interaction Terms

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

- Suppose we start with $x = (1, x_1, \dots, x_d) \in \mathbb{R}^{d+1} = \mathcal{X}$.
- 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_1x_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 $\psi : \mathcal{X} \rightarrow \mathbb{R}^d$ be a feature map.
- The SVM objective (with explicit feature map):

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

- 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 $\psi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$.

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

The inner product is

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

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. \mathbb{R}^d)
- **Feature map:** $\psi : \mathcal{X} \rightarrow \mathcal{H}$
- 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

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

- 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, \dots, 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} :

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

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 $d \gg n$.
- 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 kernels”.
- How do we know that our kernel functions actually correspond to inner products in some feature space?

How to Get Kernels?

- Explicitly construct $\psi(x) : \mathcal{X} \rightarrow \mathbb{R}^d$ (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 $M \in \mathbb{R}^{n \times n}$ is **positive semidefinite (psd)** if for any $x \in \mathbb{R}^n$,

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

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

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

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: $\sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j k(x_i, x_j) \geq 0$ given $\alpha_i \in \mathbb{R} \forall i$.

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

$$k_{\text{new}}(x, x') = \alpha k(x, x') \quad \text{for } \alpha \geq 0 \quad (\text{non-negative scaling})$$

$$k_{\text{new}}(x, x') = k_1(x, x') + k_2(x, x') \quad (\text{sum})$$

$$k_{\text{new}}(x, x') = k_1(x, x') k_2(x, x') \quad (\text{product})$$

$$k_{\text{new}}(x, x') = k(\psi(x), \psi(x')) \quad \text{for any function } \psi(\cdot) \quad (\text{recursion})$$

$$k_{\text{new}}(x, x') = f(x)f(x') \quad \text{for any function } f(\cdot) \quad (f \text{ as 1D feature map})$$

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

Linear Kernel

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

$$\psi(x) = x$$

- Kernel:

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

Quadratic Kernel in \mathbb{R}^d

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

$$\psi(x) = (x_1, \dots, x_d, x_1^2, \dots, x_d^2, \sqrt{2}x_1x_2, \dots, \sqrt{2}x_1x_d, \dots, \sqrt{2}x_{d-1}x_d)^T$$

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

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

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

Polynomial Kernel in \mathbb{R}^d

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

- Kernel function:

$$k(x, x') = (1 + \langle x, x' \rangle)^M$$

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

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

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?

SVM solution is in the “span of the data”

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

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

- 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, \dots, x_n .
- We refer to this phenomenon by saying “ w^* is in the **span of the data**.”
 - Or in math, $w^* \in \text{span}(x_1, \dots, x_n)$.

Ridge regression solution is in the “span of the data”

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

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

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

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

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

Ridge regression solution is in the “span of the data”

- Rearranging $w^* = (X^T X + \lambda I)^{-1} X^T y$, 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. $w^* \in \text{span}(x_1, \dots, x_n)$

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

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

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

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

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

- 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 $w \in \text{span}(x_1, \dots, x_n)$, 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:

$$\text{[original]} \quad w^* = \arg \min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n \{w^T x_i - y_i\}^2 + \lambda \|w\|_2^2$$

$$\text{[reparameterized]} \quad \alpha^* = \arg \min_{\alpha \in \mathbb{R}^n} \frac{1}{n} \sum_{i=1}^n \{(X^T \alpha)^T x_i - y_i\}^2 + \lambda \|X^T \alpha\|_2^2.$$

- 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 ($d \gg n$).

- For SVM and ridge regression, we found that the solution is in the span of the data.
- The Representer Theorem shows that this “span of the data” result occurs far more generally.

The Representer Theorem (Optional)

- Generalized objective:

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

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

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

- So we can reparameterize as before:

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

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

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.