CSCI567 Machine Learning (Fall 2017)

Prof. Fei Sha

U of Southern California

Lecture on Sept. 26, 2017

Outline

Outline

Administrative stuff

- Homework 2 Released
- Homework 1 Past Due: If you have not applied for Extension, your turn-in is considered late.

Outline

How to to do nonlinear prediction without specifying nonlinear basis functions?

Main idea
Use a kernel function to *implicitly* define the basis

Definition of kernel function: a (positive semidefinite) kernel function $k(\cdot, \cdot)$ is a bivariate function that satisfies the following properties. For any $\boldsymbol{x}_m$ and $\boldsymbol{x}_n$,

$$k(\boldsymbol{x}_m, \boldsymbol{x}_n) = k(\boldsymbol{x}_n, \boldsymbol{x}_m) \text{ and } k(\boldsymbol{x}_m, \boldsymbol{x}_n) = \boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n)$$

for *some* function $\boldsymbol{\phi}(\cdot)$.

How to to do nonlinear prediction without specifying nonlinear basis functions?

Main idea

Use a kernel function to *implicitly* define the basis

Definition of kernel function: a (positive semidefinite) kernel function $k(\cdot, \cdot)$ is a bivariate function that satisfies the following properties. For any $\boldsymbol{x}_m$ and $\boldsymbol{x}_n$,

$$k(\boldsymbol{x}_m, \boldsymbol{x}_n) = k(\boldsymbol{x}_n, \boldsymbol{x}_m) \text{ and } k(\boldsymbol{x}_m, \boldsymbol{x}_n) = \boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n)$$

for *some* function $\boldsymbol{\phi}(\cdot)$.

Examples we have seen

$$k(\boldsymbol{x}_m, \boldsymbol{x}_n) = (\boldsymbol{x}_m^{\mathrm{T}} \boldsymbol{x}_n)^2$$

$$k(\boldsymbol{x}_m, \boldsymbol{x}_n) = 2 - \frac{\sin(2\pi(x_{m1} - x_{n1}))}{x_{m1} - x_{n1}} - \frac{\sin(2\pi(x_{m2} - x_{n2}))}{x_{m2} - x_{n2}}$$

How do we know an arbitrary function is a kernel function?

Mercer theorem (loosely), a bivariate function $k(\cdot, \cdot)$ is a positive semidefinite kernel function, if and only if, for *any N and any $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, $\ldots$, and $\boldsymbol{x}_N$*, the matrix

$$\boldsymbol{K} = \begin{pmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & k(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ k(\boldsymbol{x}_2, \boldsymbol{x}_1) & k(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots & \vdots & \vdots & \vdots \\ k(\boldsymbol{x}_N, \boldsymbol{x}_1) & k(\boldsymbol{x}_N, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{pmatrix}$$

is positive semidefinite. We also refer $k(\cdot, \cdot)$ as a positive semidefinite kernel.

Equivalence of kernel matrices

without specifying $\phi(\cdot)$, the kernel matrix

$$\boldsymbol{K} = \begin{pmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & k(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ k(\boldsymbol{x}_2, \boldsymbol{x}_1) & k(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots & \vdots & \vdots & \vdots \\ k(\boldsymbol{x}_N, \boldsymbol{x}_1) & k(\boldsymbol{x}_N, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{pmatrix}$$

is exactly the same as

$$\begin{aligned} \boldsymbol{K} &= \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} \\ &= \begin{pmatrix} \boldsymbol{\phi}(\boldsymbol{x}_1)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_1) & \boldsymbol{\phi}(\boldsymbol{x}_1)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_2) & \cdots & \boldsymbol{\phi}(\boldsymbol{x}_1)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_N) \\ \boldsymbol{\phi}(\boldsymbol{x}_2)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_1) & \boldsymbol{\phi}(\boldsymbol{x}_2)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_2) & \cdots & \boldsymbol{\phi}(\boldsymbol{x}_2)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_N) \\ \cdots & \cdots & \cdots & \cdots \\ \boldsymbol{\phi}(\boldsymbol{x}_N)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_1) & \boldsymbol{\phi}(\boldsymbol{x}_N)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_2) & \cdots & \boldsymbol{\phi}(\boldsymbol{x}_N)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_N) \end{pmatrix} \end{aligned}$$

Examples of kernel functions

Polynomial kernel function with degree of d

$$k(\boldsymbol{x}_m, \boldsymbol{x}_n) = (\boldsymbol{x}_m^{\mathrm{T}}\boldsymbol{x}_n + c)^d$$

for $c \geq 0$ and d is a positive integer. In this case, we can write out the $\phi(\cdot)$ explicitly.

Gaussian kernel, RBF kernel, or Gaussian RBF kernel

$$k(\boldsymbol{x}_m, \boldsymbol{x}_n) = e^{-\|\boldsymbol{x}_m - \boldsymbol{x}_n\|_2^2/2\sigma^2}$$

In this case, we can *not* write out the $\phi(\cdot)$ easily.

Kernelization trick

Many learning methods depend on computing *inner products* between features — we have seen the example of regularized least squares. For those methods, we can use a kernel function in the place of the inner products, i.e., *“kernerlizing”* the methods, thus, introducing nonlinear features/basis.

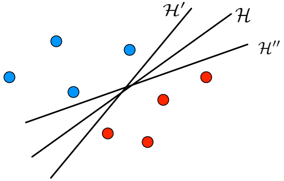
Outline

Support vector machines

- One of the most commonly used machine learning algorithms besides deep learning.
- Convex optimization for classification and regression.
- It incorporates kernel tricks to define nonlinear decision boundaries or regression functions.
- It provides theoretical guarantees on generalization errors.

Intuition: where to put the decision boundary?

Binary classification when the *training* dataset is *separable* — there is a decision boundary that separates the two classes perfectly.



The classifier makes correct prediction if

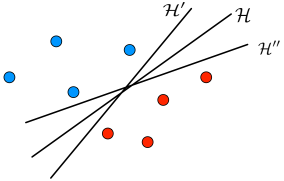
$$y[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + b] \geq 0$$

where the ground-truth y is either $+1$ or -1 and our decision rule is

$$\text{sign}[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + b]$$

Problem: there are *infinite* many ways of putting the decision boundary $\mathcal{H} : \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + b = 0$!

Best location: middle



Our intuition is to put the decision boundary to be in the *middle* of the two classes as much as possible. In other words, we want the decision boundary is to be far to every point as much as possible *as long as the decision boundary classifies every point correctly.*

We will be using $\phi(\boldsymbol{x})$ to represent a data point $\boldsymbol{x}$ as we would be thinking of transformed features anyway.

Distances

The distance from a point $\phi(x)$ to the decision boundary is

$$d_{\mathcal{H}}(\boldsymbol{\phi}(\boldsymbol{x})) = \frac{|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + b|}{\|\boldsymbol{w}\|_2}$$

We can remove the absolute $|\cdot|$ by exploiting the fact that the decision boundary classifies every point in the training dataset *correctly*. Namely, $(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + b)$ and $\boldsymbol{x}$'s label y are of the same sign. The distance is now,

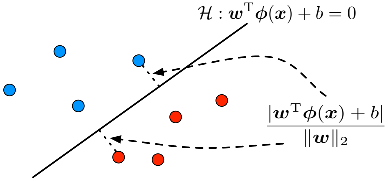
$$d_{\mathcal{H}}(\boldsymbol{\phi}(\boldsymbol{x})) = \frac{y[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + b]}{\|\boldsymbol{w}\|_2}$$

Note that $\|\boldsymbol{w}\|_2$ here represents the length of the vector $\boldsymbol{w}$.

Maximizing margin

Margin The margin is defined as the smallest distance from all the training points

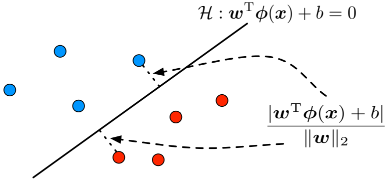
$$\text{MARGIN} = \min_n \frac{y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b]}{\|\boldsymbol{w}\|_2}$$


$\mathcal{H} : \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + b = 0$
$\frac{|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + b|}{\|\boldsymbol{w}\|_2}$

Maximizing margin

Margin The margin is defined as the smallest distance from all the training points

$$\text{MARGIN} = \min_n \frac{y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b]}{\|\boldsymbol{w}\|_2}$$


$\mathcal{H} : \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + b = 0$
$\frac{|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + b|}{\|\boldsymbol{w}\|_2}$

Since we are interested in finding a $\boldsymbol{w}$ to put all points *as distant as possible* from the decision boundary, we maximize the margin

$$\max_{\boldsymbol{w}} \quad \min_n \frac{y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b]}{\|\boldsymbol{w}\|} = \max_{\boldsymbol{w}} \frac{1}{\|\boldsymbol{w}\|_2} \min_n y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b]$$

Rescaled margin

Since the MARGIN does not change if we scale $(\boldsymbol{w}, b)$ by a constant factor c (as $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + b = 0$ and $(c\boldsymbol{w})^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + (cb) = 0$ are the same decision boundary), we fix the scale by forcing

$$\min_n y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] = 1$$

Rescaled margin

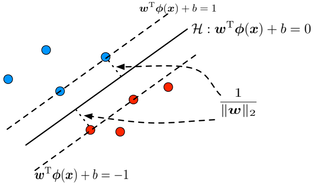
Since the MARGIN does not change if we scale $(\boldsymbol{w}, b)$ by a constant factor c (as $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + b = 0$ and $(c\boldsymbol{w})^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + (cb) = 0$ are the same decision boundary), we fix the scale by forcing

$$\min_n y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] = 1$$

In this case, our margin becomes

$$\text{MARGIN} = \frac{1}{\|\boldsymbol{w}\|_2}$$

precisely, the closest point to the decision boundary has a distance of that.


$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + b = 1$
$\mathcal{H} : \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + b = 0$
$\frac{1}{\|\boldsymbol{w}\|_2}$
$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + b = -1$

Support vector machines (SVMs) for separable data

Combining everything we have, for a separable training dataset, we aim to

$$\max_{\boldsymbol{w}} \frac{1}{\|\boldsymbol{w}\|_2} \quad \text{such that} \quad y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] \geq 1, \quad \forall \quad n$$

This is equivalent to

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 \\ \text{s.t.} \quad & y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] \geq 1, \quad \forall \quad n \end{aligned}$$

This is called SVMs primal formulation for separable data. For this geometric intuition, SVM is called *max margin* (or large margin) classifier. The constraints are called *large margin constraints*.

SVM for non-separable data

Suppose there are training data points that cannot be classified correctly no matter how we choose $\boldsymbol{w}$. For those data points,

$$y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] \leq 0$$

for any $\boldsymbol{w}$. Thus, the previous constraint

$$y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] \geq 1, \quad \forall \quad n$$

is no longer *feasible*.

Slack variables for non-separable data

To deal with this issue, we introduce *slack variables* ξ_n to help

$$y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] \geq 1 - \xi_n, \quad \forall \quad n$$

where we also require $\xi_n \geq 0$. *Why?*

Note that, even for "hard" points that cannot be classified correctly, the slack variable will be able to make them satisfy the above constraint (we can keep increasing ξ_n until the above inequality is met.)

SVM Primal formulation with slack variables

We obviously do not want ξ_n goes to infinity, so we balance their sizes by penalizing them toward zero as much as possible

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_n \xi_n \\ \text{s.t.} \quad & y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] \geq 1 - \xi_n, \quad \forall \ n \\ & \xi_n \geq 0, \quad \forall \ n \end{aligned}$$

where C is our tradeoff (hyper)parameter. This is called the primal formulation for SVM.

Let us recall how perceptron works (see the previous lecture)?

A loss function

$$L(f(\boldsymbol{x}), y) = L(\text{sign}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}), y) = \begin{cases} 0 & \text{if } y = f(x) \\ -y\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} & \text{if } y \neq f(x) \end{cases}$$

Note that $y = f(x)$ is equivalent to

$$yf(x) > 0$$

The optimal w should minimize, given a training dataset $\{(\boldsymbol{x}_n, y_n)\}$

$$\boldsymbol{w} = \arg\min_{\boldsymbol{w}} \sum_n L(f(\boldsymbol{x}_n), y_n)$$

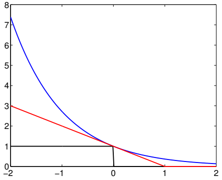
How do we do that?

We use stochastic gradient descent and this leads to the update rule for perceptron.

Hinge loss

Definition Aassuming the label $y \in \{-1, 1\}$ and the decision rule is $h(\boldsymbol{x}) = \textsc{sign}(f(\boldsymbol{x}))$ with $f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + b$,

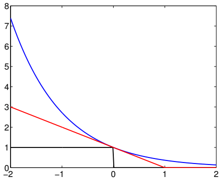
$$\ell^{\text{HINGE}}(f(\boldsymbol{x}), y) = \begin{cases} 0 & \text{if } yf(\boldsymbol{x}) \geq 1 \\ 1 - yf(\boldsymbol{x}) & \text{otherwise} \end{cases} = \max(0, 1 - yf(\boldsymbol{x}))$$



Intuition:

- penalize more if incorrectly classified (the slope)
- penalize even if correctly classified, but not with "high confidence" (between 0 and 1 on the horizontal axis)

Properties



- Upper-bound (above) the $0/1$ loss function (black line); optimizing it leads to reduced classification errors — namely, we use the hinge loss function as a *surrogate* to the true error function we care about.
- This function is not differentiable at the kink point!

Primal formulation of support vector machines (SVM)

Minimizing the total hinge loss on all the training data

$$\min_{\boldsymbol{w},b} \sum_n \max(0, 1 - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b]) + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$$

which is analogous to regularized least square, which balances two terms (the loss and the regularizer).

Primal formulation of support vector machines (SVM)

Minimizing the total hinge loss on all the training data

$$\min_{\boldsymbol{w},b} \sum_n \max(0, 1 - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b]) + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$$

which is analogous to regularized least square, which balances two terms (the loss and the regularizer). Conventionally, we rewrite the objective function as

$$\min_{\boldsymbol{w},b} C\sum_n \max(0, 1 - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b]) + \frac{1}{2}\|\boldsymbol{w}\|_2^2$$

where C is identified as $1/\lambda$.

Equivalent form

We further rewrite into another equivalent form

$$\begin{aligned} \min_{\boldsymbol{w},b,\{\xi_n\}} \quad & C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 \\ \text{s.t.} \quad & \max(0, 1 - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b]) = \xi_n, \quad \forall\ n \end{aligned}$$

Or another equivalent form

$$\begin{aligned} \min_{\boldsymbol{w},b,\{\xi_n\}} \quad & C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 \\ \text{s.t.} \quad & 1 - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] \le \xi_n, \quad \forall\ n \\ & \xi_n \ge 0, \quad \forall\ n \end{aligned}$$

where all ξ_n are called *slack variables*.

Properties of the SVM primal formulation

$$\begin{aligned}
\min_{\boldsymbol{w}, b, \{\xi_n\}} \quad & C \sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 \\
\text{s.t.} \quad & 1 - y_n[\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n) + b] \leq \xi_n, \quad \forall\ n \\
& \xi_n \geq 0, \quad \forall\ n
\end{aligned}$$

- This is a *convex quadratic programming*: the objective function is quadratic in $\boldsymbol{w}$ and the constraints are linear (inequality) constraints in $\boldsymbol{w}$ and ξ_n.

Properties of the SVM primal formulation

$$\begin{aligned}\min_{\boldsymbol{w},b,\{\xi_n\}} \quad & C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 \\ \text{s.t.} \quad & 1 - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] \le \xi_n, \quad \forall\ n \\ & \xi_n \ge 0, \quad \forall\ n\end{aligned}$$

- This is a *convex quadratic programming*: the objective function is quadratic in $\boldsymbol{w}$ and the constraints are linear (inequality) constraints in $\boldsymbol{w}$ and ξ_n.
- Given $\phi(\cdot)$, we can solve the optimization problem efficiently as it is convex, for example, using Matlab's generic `quadprog()` function.

Properties of the SVM primal formulation

$$\begin{aligned} \min_{\boldsymbol{w},b,\{\xi_n\}} \quad & C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 \\ \text{s.t.} \quad & 1 - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] \le \xi_n, \quad \forall\ n \\ & \xi_n \ge 0, \quad \forall\ n \end{aligned}$$

- This is a *convex quadratic programming*: the objective function is quadratic in $\boldsymbol{w}$ and the constraints are linear (inequality) constraints in $\boldsymbol{w}$ and ξ_n.
- Given $\phi(\cdot)$, we can solve the optimization problem efficiently as it is convex, for example, using Matlab's generic `quadprog()` function.
- However, there are *more efficient* algorithms for solving this problem, taking advantage of the special structures of the objective function and the constraints. (We will not discuss them. Most existing SVM implementation/packages implement such efficient algorithms.)

Basic Lagrange duality theory

Key concepts you should know

- What do “primal” and “dual” mean?
- How SVM exploits dual formulation, thus results in using kernel functions for nonlinear classification
- What do support vectors mean?

Our roadmap

- We will tell you what dual looks like
- We will show you how it is derived

Dual formulation

Dual is also a convex quadratic programming

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_n \alpha_n - \frac{1}{2} \sum_{m,n} y_m y_n \alpha_m \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n) \\ \text{s.t.} \quad & 0 \leq \alpha_n \leq C, \quad \forall\ n \\ & \sum_n \alpha_n y_n = 0 \end{aligned}$$

Dual formulation

Dual is also a convex quadratic programming

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_n \alpha_n - \frac{1}{2}\sum_{m,n} y_m y_n \alpha_m \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n) \\ \text{s.t.} \quad & 0 \le \alpha_n \le C, \quad \forall\ n \\ & \sum_n \alpha_n y_n = 0 \end{aligned}$$

Remarks

- The optimization is convex as we can minimize the negation of the objective function.

$$\frac{1}{2}\sum_{m,n} y_m y_n \alpha_m \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n) - \sum_n \alpha_n$$

  This is a convex quadratic function in αs – can you verify?

Dual formulation

Dual is also a convex quadratic programming

$$\max_{\boldsymbol{\alpha}} \quad \sum_n \alpha_n - \frac{1}{2}\sum_{m,n} y_m y_n \alpha_m \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n)$$

$$\text{s.t.} \quad 0 \le \alpha_n \le C, \quad \forall\ n$$

$$\sum_n \alpha_n y_n = 0$$

Remarks

- The optimization is convex as we can minimize the negation of the objective function.

$$\frac{1}{2}\sum_{m,n} y_m y_n \alpha_m \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n) - \sum_n \alpha_n$$

  This is a convex quadratic function in αs – can you verify?

- *There are N dual variable α_n, one for each constraint* $1 - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b]) \le \xi_n$ in the primal formulation.

The reason we need to go to dual: kernelized SVM!

We replace the inner products $\phi(\boldsymbol{x}_m)^{\mathrm{T}}\phi(\boldsymbol{x}_n)$ with a kernel function

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_n \alpha_n - \frac{1}{2}\sum_{m,n} y_m y_n \alpha_m \alpha_n k(\boldsymbol{x}_m, \boldsymbol{x}_n) \\ \text{s.t.} \quad & 0 \leq \alpha_n \leq C, \quad \forall\ n \\ & \sum_n \alpha_n y_n = 0 \end{aligned}$$

as in kernelized linear regression and kernerlized nearest neighbor. We only need to define a kernel function and we will automatically get (nonlinearly) mapped features and the support vector machine constructed with those features.

In dual formulation, we do not need $\phi(\boldsymbol{x})$!

Recovering solution to the primal formulation

Weights

$$\boldsymbol{w} = \sum_n y_n \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_n) \leftarrow \text{ Linear combination of the input features!}$$

Recovering solution to the primal formulation

Weights

$$\boldsymbol{w} = \sum_n y_n \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_n) \leftarrow \text{ Linear combination of the input features!}$$

b

$$b = [y_n - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n)] = [y_n - \sum_m y_m \alpha_m k(\boldsymbol{x}_m, \boldsymbol{x}_n)], \quad \text{for any } C > \alpha_n > 0$$

Recovering solution to the primal formulation

Weights

$$\boldsymbol{w} = \sum_n y_n \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_n) \leftarrow \text{ Linear combination of the input features!}$$

b

$$b = [y_n - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n)] = [y_n - \sum_m y_m \alpha_m k(\boldsymbol{x}_m, \boldsymbol{x}_n)], \quad \text{for any } C > \alpha_n > 0$$

Making prediction on a test point x

$$h(\boldsymbol{x}) = \mathrm{SIGN}(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}) + b) = \mathrm{SIGN}(\sum_n y_n \alpha_n k(\boldsymbol{x}_n, \boldsymbol{x}) + b)$$

Again, to make prediction, it suffices to know the kernel function.

Derivation of the dual

We will derive the dual formulation as the process will reveal some interesting and important properties of SVM. Particularly, why is it called “support vector”?

Steps

- Formulate a Lagrangian function that incorporates the constraints, thru introducing dual variables
- Minimize the Lagrangian function to solve the primal variables
- Put the primal variables into the Lagrangian and express in terms of dual variables
- Maximize the Lagrangian with respect to dual variables
- Recover the solution (for the primal variables) from the dual variables

A simple example

Consider the example of convex quadratic programming

$$\begin{aligned} \min \quad & \frac{1}{2}x^2 \\ \text{s.t.} \quad & -x \leq 0 \\ & 2x - 3 \leq 0 \end{aligned}$$

The Lagrangian is (note that we do not have equality constraints)

$$L(x, \mu) = \frac{1}{2}x^2 + \mu_1 \times (-x) + \mu_2 \times (2x - 3) = \frac{1}{2}x^2 + (2\mu_2 - \mu_1)x - 3\mu_2$$

under the constraint that $\mu_1 \geq 0$ and $\mu_2 \geq 0$.

A simple example

Consider the example of convex quadratic programming

$$\begin{aligned} \min \quad & \frac{1}{2}x^2 \\ \text{s.t.} \quad & -x \leq 0 \\ & 2x - 3 \leq 0 \end{aligned}$$

The Lagrangian is (note that we do not have equality constraints)

$$L(x, \mu) = \frac{1}{2}x^2 + \mu_1 \times (-x) + \mu_2 \times (2x - 3) = \frac{1}{2}x^2 + (2\mu_2 - \mu_1)x - 3\mu_2$$

under the constraint that $\mu_1 \geq 0$ and $\mu_2 \geq 0$. Its dual problem is

$$\max_{\mu_1 \geq 0, \mu_2 \geq 0} \min_x L(x, \mu) = \max_{\mu_1 \geq 0, \mu_2 \geq 0} \min_x \frac{1}{2}x^2 + (2\mu_2 - \mu_1)x - 3\mu_2$$

Example (cont'd)

We solve the $\min_x L(x, \mu)$ first now it is unconstrained. The optimal x is attained by

$$\frac{\partial(\frac{1}{2}x^2 + (2\mu_2 - \mu_1)x - 3\mu_2)}{\partial x} = 0 \rightarrow x = -(2\mu_2 - \mu_1)$$

Example (cont'd)

We solve the $\min_x L(x, \mu)$ first now it is unconstrained. The optimal x is attained by

$$\frac{\partial(\frac{1}{2}x^2 + (2\mu_2 - \mu_1)x - 3\mu_2)}{\partial x} = 0 \rightarrow x = -(2\mu_2 - \mu_1)$$

This gives us the dual objective function, by substituting the solution into the objective function,

$$g(\mu) = \min_x \frac{1}{2}x^2 + (2\mu_2 - \mu_1)x - 3\mu_2 = -\frac{1}{2}(2\mu_2 - \mu_1)^2 - 3\mu_2$$

Example (cont'd)

We solve the $\min_x L(x, \mu)$ first now it is unconstrained. The optimal x is attained by

$$\frac{\partial(\frac{1}{2}x^2 + (2\mu_2 - \mu_1)x - 3\mu_2)}{\partial x} = 0 \rightarrow x = -(2\mu_2 - \mu_1)$$

This gives us the dual objective function, by substituting the solution into the objective function,

$$g(\mu) = \min_x \frac{1}{2}x^2 + (2\mu_2 - \mu_1)x - 3\mu_2 = -\frac{1}{2}(2\mu_2 - \mu_1)^2 - 3\mu_2$$

We get our dual problem as

$$\max_{\mu_1 \geq 0, \mu_2 \geq 0} -\frac{1}{2}(2\mu_2 - \mu_1)^2 - 3\mu_2$$

We will solve the dual next.

Solving the dual

Note that,

$$g(\mu) = -\frac{1}{2}(2\mu_2 - \mu_1)^2 - 3\mu_2 \leq 0$$

for all $\mu_1 \geq 0, \mu_2 \geq 0$. Thus, to maximize the function, the optimal solution is

$$\mu_1^* = 0, \quad \mu_2^* = 0$$

This brings us back the optimal solution of x

$$x^* = -(2\mu_2^* - \mu_1^*) = 0$$

Namely, we have arrived at the same solution as the one we guessed from the primal formulation

Deriving the dual for SVM

From primal

$$\begin{aligned} \min_{\boldsymbol{w},b,\{\xi_n\}} \quad & C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 \\ \text{s.t.} \quad & 1 - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] \le \xi_n, \quad \forall\ n \\ & \xi_n \ge 0, \quad \forall\ n \end{aligned}$$

Lagrangian

$$\begin{aligned} L(\boldsymbol{w}, \{\xi_n\}, \{\alpha_n\}, \{\lambda_n\}) = C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \sum_n \lambda_n\xi_n \\ + \sum_n \alpha_n\{1 - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] - \xi_n\} \end{aligned}$$

under the constraint that $\alpha_n \ge 0$ and $\lambda_n \ge 0$.

Minimizing the Lagrangian

Taking derivatives with respect to the primal variables

$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_n y_n \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_n) = 0$$

Minimizing the Lagrangian

Taking derivatives with respect to the primal variables

$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_n y_n \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_n) = 0$$

$$\frac{\partial L}{\partial b} = \sum_n \alpha_n y_n = 0$$

Minimizing the Lagrangian

Taking derivatives with respect to the primal variables

$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_n y_n \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_n) = 0$$

$$\frac{\partial L}{\partial b} = \sum_n \alpha_n y_n = 0$$

$$\frac{\partial L}{\partial \xi_n} = C - \lambda_n - \alpha_n = 0$$

Minimizing the Lagrangian

Taking derivatives with respect to the primal variables

$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_n y_n \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_n) = 0$$

$$\frac{\partial L}{\partial b} = \sum_n \alpha_n y_n = 0$$

$$\frac{\partial L}{\partial \xi_n} = C - \lambda_n - \alpha_n = 0$$

This gives rise to equations linking the primal variables and the dual variables as well as new constraints on the dual variables:

$$\boldsymbol{w} = \sum_n y_n \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_n)$$

$$\sum_n \alpha_n y_n = 0$$

$$C - \lambda_n - \alpha_n = 0$$

Rewrite the Lagrange in terms of dual variables

Substitute the solution to the primal back into the Lagrangian

$$g(\{\alpha_n\},\{\lambda_n\}) = L(\boldsymbol{w}, \{\xi_n\}, \{\alpha_n\}, \{\lambda_n\})$$

Rewrite the Lagrange in terms of dual variables

Substitute the solution to the primal back into the Lagrangian

$$\begin{aligned} g(\{\alpha_n\},\{\lambda_n\}) &= L(\boldsymbol{w}, \{\xi_n\}, \{\alpha_n\}, \{\lambda_n\}) \\ &= \sum_n (C - \alpha_n - \lambda_n)\xi_n + \frac{1}{2}\| \sum_n y_n \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_n)\|_2^2 + \sum_n \alpha_n \\ &+ \left(\sum_n \alpha_n y_n\right) b - \sum_n \alpha_n y_n \left(\sum_m y_m \alpha_m \boldsymbol{\phi}(\boldsymbol{x}_m)\right)^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n) \end{aligned}$$

Rewrite the Lagrange in terms of dual variables

Substitute the solution to the primal back into the Lagrangian

$$
\begin{aligned}
g(\{\alpha_n\},\{\lambda_n\}) &= L(\boldsymbol{w}, \{\xi_n\}, \{\alpha_n\}, \{\lambda_n\}) \\
&= \sum_n (C - \alpha_n - \lambda_n)\xi_n + \frac{1}{2}\|\sum_n y_n\alpha_n\boldsymbol{\phi}(\boldsymbol{x}_n)\|_2^2 + \sum_n \alpha_n \\
&+ \left(\sum_n \alpha_n y_n\right) b - \sum_n \alpha_n y_n \left(\sum_m y_m\alpha_m\boldsymbol{\phi}(\boldsymbol{x}_m)\right)^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n) \\
&= \sum_n \alpha_n + \frac{1}{2}\|\sum_n y_n\alpha_n\boldsymbol{\phi}(\boldsymbol{x}_n)\|_2^2 - \sum_{m,n} \alpha_n\alpha_m y_m y_n \boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n)
\end{aligned}
$$

Rewrite the Lagrange in terms of dual variables

Substitute the solution to the primal back into the Lagrangian

$$
\begin{aligned}
g(\{\alpha_n\},\{\lambda_n\}) &= L(\boldsymbol{w}, \{\xi_n\}, \{\alpha_n\}, \{\lambda_n\}) \\
&= \sum_n (C - \alpha_n - \lambda_n)\xi_n + \frac{1}{2}\|\sum_n y_n \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_n)\|_2^2 + \sum_n \alpha_n \\
&+ \left(\sum_n \alpha_n y_n\right) b - \sum_n \alpha_n y_n \left(\sum_m y_m \alpha_m \boldsymbol{\phi}(\boldsymbol{x}_m)\right)^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n) \\
&= \sum_n \alpha_n + \frac{1}{2}\|\sum_n y_n \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_n)\|_2^2 - \sum_{m,n} \alpha_n \alpha_m y_m y_n \boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n) \\
&= \sum_n \alpha_n - \frac{1}{2}\sum_{m,n} \alpha_n \alpha_m y_m y_n \boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n)
\end{aligned}
$$

Several terms vanish because of the constraints $\sum_n \alpha_n y_n = 0$ and $C - \lambda_n - \alpha_n = 0$.

Deriving this is not required - but if you are curious, you should try to follow.

The dual problem

Maximizing the dual under the constraints

$$\max_{\boldsymbol{\alpha}} \quad g(\{\alpha_n\},\{\lambda_n\}) = \sum_n \alpha_n - \frac{1}{2}\sum_{m,n} y_m y_n \alpha_m \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n)$$

$$\begin{aligned}
\text{s.t.} \quad & \alpha_n \geq 0, \quad \forall\, n \\
& \sum_n \alpha_n y_n = 0 \\
& C - \lambda_n - \alpha_n = 0, \quad \forall\, n \\
& \lambda_n \geq 0, \quad \forall\, n
\end{aligned}$$

The dual problem

Maximizing the dual under the constraints

$$
\begin{aligned}
\max_{\boldsymbol{\alpha}} \quad & g(\{\alpha_n\}, \{\lambda_n\}) = \sum_n \alpha_n - \frac{1}{2}\sum_{m,n} y_m y_n \alpha_m \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n) \\
\text{s.t.} \quad & \alpha_n \geq 0, \quad \forall\ n \\
& \sum_n \alpha_n y_n = 0 \\
& C - \lambda_n - \alpha_n = 0, \quad \forall\ n \\
& \lambda_n \geq 0, \quad \forall\ n
\end{aligned}
$$

We can simplify as the objective function does not depend on λ_n, thus we can convert the equality constraint involving λ_n with an inequality constraint on $\alpha_n \leq C$:

$$\alpha_n \leq C \leftrightarrow \lambda_n = C - \alpha_n \geq 0 \leftrightarrow C - \lambda_n - \alpha_n = 0, \lambda_n \geq 0$$

Final form

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_n \alpha_n - \frac{1}{2}\sum_{m,n} y_m y_n \alpha_m \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n) \\ \text{s.t.} \quad & 0 \leq \alpha_n \leq C, \quad \forall\ n \\ & \sum_n \alpha_n y_n = 0 \end{aligned}$$

We call this as *dual form* of SVM.

Recover the solution to the primal problem

The primal variable $\boldsymbol{w}$ is identified as

$$\boldsymbol{w} = \sum_n \alpha_n y_n \boldsymbol{\phi}(\boldsymbol{x}_n)$$

Recover the solution to the primal problem

The primal variable $\boldsymbol{w}$ is identified as

$$\boldsymbol{w} = \sum_n \alpha_n y_n \boldsymbol{\phi}(\boldsymbol{x}_n)$$

To identify b, we need something else.

Complementary slackness and support vectors

At the optimal solution to both primal and dual, the following must be satisfied for every inequality constraint (these are called KKT conditions)

$$\begin{aligned}&\lambda_n \xi_n = 0\\ &\alpha_n\{1 - \xi_n - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b]\} = 0\end{aligned}$$

Complementary slackness and support vectors

At the optimal solution to both primal and dual, the following must be satisfied for every inequality constraint (these are called KKT conditions)

$$\lambda_n \xi_n = 0$$
$$\alpha_n \{1 - \xi_n - y_n[\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n) + b]\} = 0$$

From the first condition, if $\alpha_n < C$, then

$$\lambda_n = C - \alpha_n > 0 \rightarrow \xi_n = 0$$

Thus, in conjunction with the second condition, we know that, if $C > \alpha_n > 0$, then

$$1 - y_n[\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n) + b] = 0 \rightarrow b = y_n - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n)$$

as $y_n \in \{-1, 1\}$.

For those n whose $\alpha_n > 0$, we call such training samples as "support vectors". (We will discuss their geometric interpretation later).

Meaning of “support vectors” in SVMs

Complementary slackness At optimum, we have to have

$$\alpha_n \{1 - \xi_n - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b]\} = 0, \quad \forall \;\; n$$

That means, for some n, $\alpha_n = 0$. Additionally, our optimal solution is given by

$$\boldsymbol{w} = \sum_n \alpha_n y_n \boldsymbol{\phi}(\boldsymbol{x}_n) = \sum_{n:\alpha_n > 0} \alpha_n y_n \boldsymbol{\phi}(\boldsymbol{x}_n)$$

In words, our solution is only determined by those training samples whose corresponding α_n is strictly positive. Those samples are called *support vectors*.

Non-support vectors whose $\alpha_n = 0$ can be removed by the training dataset — this removal will not affect the optimal solution (i.e., after the removal, if we construct another SVM classifier on the reduced dataset, the optimal solution is the same as the one on the original dataset.)

Who are support vectors?

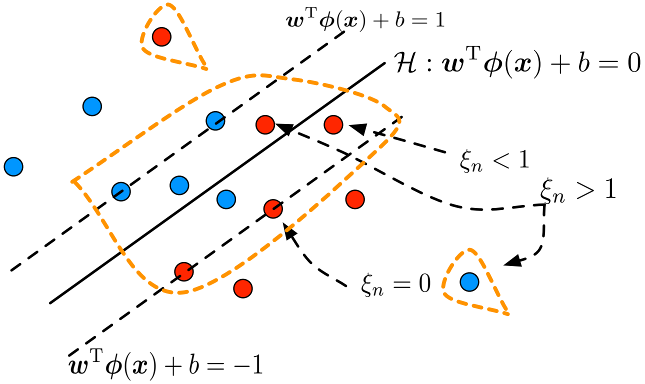
Case analysis Since, we have

$$1 - \xi_n - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b]\} = 0$$

We have

- $\xi_n = 0$. This implies $y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] = 1$. They are on points that are $1/\|\boldsymbol{w}\|_2$ away from the decision boundary.
- $\xi_n < 1$. These are points that can be classified correctly but do not satisfy the large margin constraint – they have smaller distances to the decision boundary.
- $\xi_n > 1$. These are points that are misclassified.

Visualization of how training data points are categorized


$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + b = 1$
$\mathcal{H} : \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + b = 0$
$\xi_n < 1$
$\xi_n > 1$
$\xi_n = 0$
$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + b = -1$

Support vectors are those being circled with the orange line.

Demo of SVM