18-661 Introduction to Machine Learning

SVM – II

Spring 2025

ECE – Carnegie Mellon University

Announcements

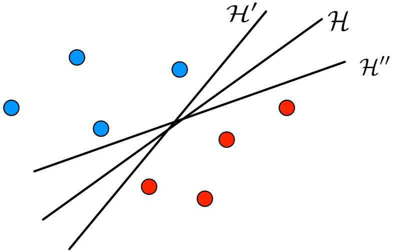
- Homework 2 is due this Friday, February 21 at 11:59pm ET.
- Midterm exam is on Wednesday, February 26 in lecture.
  - Format: In-person, paper-based exam
  - Closed-book, but you can use a letter- or A4-sized double-sided, handwritten cheat sheet.
  - No calculators allowed.
  - Topics: MLE/MAP, linear regression, overfitting and bias-variance tradeoff, naive Bayes, logistic regression, multi-class classification, SVM (max margin and hinge loss formulations), and nearest neighbors.

Outline

Review of Max Margin SVM Formulation

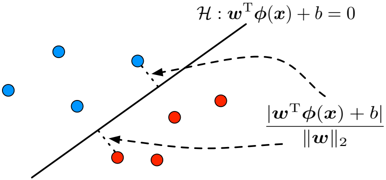
Intuition: Where to Put the Decision Boundary?

- Consider a *separable* training dataset (e.g., with two features)
- There are an infinite number of decision boundaries $\mathcal{H}: \boldsymbol{w}^\top \boldsymbol{x} + b = 0$!



- Which one should we pick?

Optimizing the Margin



How should we pick $(\boldsymbol{w}, b)$ based on its margin?
We want a decision boundary that is as far away from all training points as possible, so we to *maximize* the margin!

$$\max_{\boldsymbol{w},b}\left(\min_n \frac{y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b]}{\|\boldsymbol{w}\|_2}\right) = \max_{\boldsymbol{w},b}\left(\frac{1}{\|\boldsymbol{w}\|_2}\min_n y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b]\right)$$

Only involves points near the boundary (more on this later).

Rescaled Margin

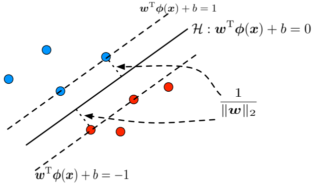
We can further constrain the problem by scaling $(\boldsymbol{w}, b)$ such that

$$\min_n y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] = 1.$$

Note that there always exists a scaling for which this is true. We've fixed the numerator in the $\textsc{margin}(\boldsymbol{w}, b)$ equation, and we have:

$$\textsc{margin}(\boldsymbol{w}, b) = \frac{\min_n y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b]}{\|\boldsymbol{w}\|_2} = \frac{1}{\|\boldsymbol{w}\|_2}$$

Hence the points closest to the decision boundary are at distance $\frac{1}{\|\boldsymbol{w}\|_2}$.



SVM: Max-margin Formulation for Separable Data

We thus want to solve:

$$\max_{\boldsymbol{w},b} \underbrace{\frac{1}{\|\boldsymbol{w}\|_2}}_{\text{margin}} \quad \text{such that} \quad \underbrace{\min_n y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] = 1}_{\text{scaling of } \boldsymbol{w},\, b}$$

which is equivalent to

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

This is further equivalent to

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

Given our geometric intuition, SVM is called a **max margin** (or large margin) classifier. The constraints are called **large margin constraints**.

SVM for Non-separable Data

Constraints in separable setting

$$y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1, \quad \forall \quad n$$

Constraints in non-separable setting
Can we modify our constraints to account for non-separability?

Specifically, we introduce slack variables $\xi_n \geq 0$:

$$y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n, \quad \forall \quad n$$

- For "hard" training points, we can increase ξ_n until the above inequalities are met.
- What does it mean when ξ_n is very large? We have violated the original constraints "by a lot."

Soft-margin SVM Formulation

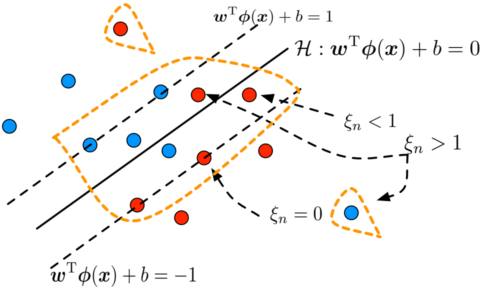
We do not want ξ_n to grow too large, and we can control their size by incorporating them into our optimization problem:

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

What is the role of C?

- User-defined hyperparameter
- Trades off between the two terms in our objective
- Same idea as the regularization term in ridge regression

Support Vectors: Revisit



Support vectors satisfy $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] = 1 - \xi_n$:

- $\xi_n = 0$: $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] = 1$, the point is on the correct side with distance $\frac{1}{\|\boldsymbol{w}\|}$.
- $0 < \xi_n \leq 1$: $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \in [0, 1)$ on the correct side, but with distance less than $\frac{1}{\|\boldsymbol{w}\|}$.
- $\xi_n > 1$: $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] < 0$, on the wrong side of the boundary.

SVM: Hinge Loss Formulation

SVM vs. Logistic Regression

SVM soft-margin formulation

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_n \xi_n$$

$$\text{s.t.} \quad y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n, \;\; \forall\; n$$

$$\xi_n \geq 0, \quad \forall\; n$$

Logistic regression formulation

$$\min_{\boldsymbol{w}} - \sum_n \{y_n \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1 - y_n)\log[1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]\} + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$$

- Logistic regression defines a loss for each data point and minimizes the total loss plus a regularization term.
- This is convenient for assessing the "goodness" of the model on each data point.
- Can we write SVMs in this form as well? The Hinge Loss formulation!

Derive the Hinge Loss Formulation

Here's the soft-margin formulation again:

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \ \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_n \xi_n \ \text{ s.t. } y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n, \ \xi_n \geq 0, \ \forall \ n$$

Now since $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n \iff \xi_n \geq 1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b]$:

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \ C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 \text{ s.t. } \xi_n \geq \max(0, 1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b]), \ \forall \ n$$

Now since the ξ_n should always be as small as possible, we obtain:

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

Divide by C and set $\lambda = \frac{1}{C}$, we get the Hinge Loss formulation:

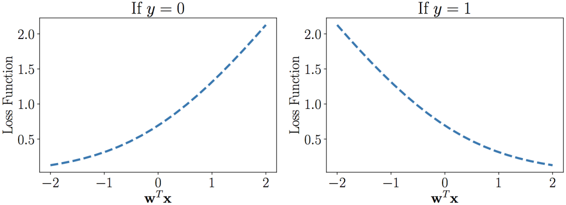
$$\min_{\boldsymbol{w},b} \ \sum_n \underbrace{\max(0, 1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b])}_{\text{Hinge Loss for } x_n, y_n} + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$$

Cross-Entropy Loss vs. Hinge Loss

Given training data (x_n, y_n), the cross entropy loss was

$$-\{y_n \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1 - y_n) \log[1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]\}$$

$$-\log\left(\frac{e^{-\mathbf{w}^T\mathbf{x_n}}}{1+e^{-\mathbf{w}^T\mathbf{x_n}}}\right) \qquad -\log\left(\frac{1}{1+e^{-\mathbf{w}^T\mathbf{x_n}}}\right)$$

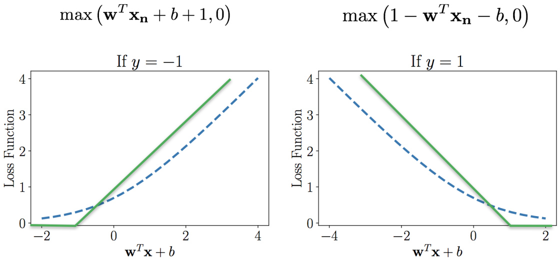


What does the Hinge Loss Function look like?

Cross-Entropy Loss vs. Hinge Loss

Given training data (x_n, y_n), the Hinge loss is

$$\max(0, 1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b])$$



- Loss grows linearly as we move away from the boundary.
- No penalty if a point is more than 1 unit from the boundary.
- Makes the search for the boundary easier (as we will see later).

Hinge Loss SVM Formulation

Minimizing the total hinge loss on all the training data

$$\min_{\boldsymbol{w},b} \sum_n \underbrace{\max(0, 1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b])}_{\text{hinge loss for sample } n} + \underbrace{\frac{\lambda}{2}\|\boldsymbol{w}\|_2^2}_{\text{regularizer}}$$

Analogous to regularized least squares or logistic regression, as we balance between two terms (the loss and the regularizer).

- Can solve using gradient descent to get the optimal $\mathbf{w}$ and b
- Gradient of the first term will be either 0, $\mathbf{x}_n$ or $-\mathbf{x}_n$ depending on y_n and $\boldsymbol{w}^\top \boldsymbol{x}_n + b$.
- Much easier to compute than in logistic regression, where we need to compute the sigmoid function $\sigma(\boldsymbol{w}^\top \boldsymbol{x}_n + b)$ in each iteration.

Three SVM Formulations

Hard-margin (for separable data)

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \ \frac{1}{2}\|\boldsymbol{w}\|_2^2 \text{ s.t. } y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1,\ \xi_n \geq 0,\ \forall\ n$$

Soft-margin (add slack variables)

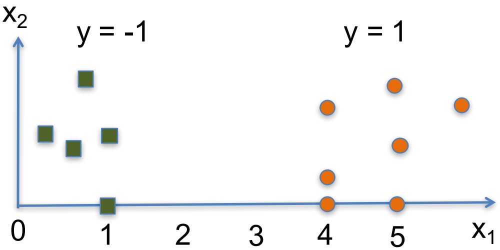
$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \ \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_n \xi_n \ \text{ s.t. } y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n,\ \xi_n \geq 0,\ \forall\ n$$

Hinge loss (define a loss function for each data point)

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

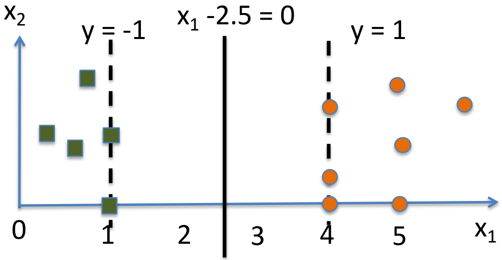
SVM: Example

Example of SVM



What will be the decision boundary learnt by solving the SVM optimization problem?

Example of SVM



Margin $= 1.5$; the decision boundary has $\mathbf{w} = [1, 0]^\top$, and $b = -2.5$.

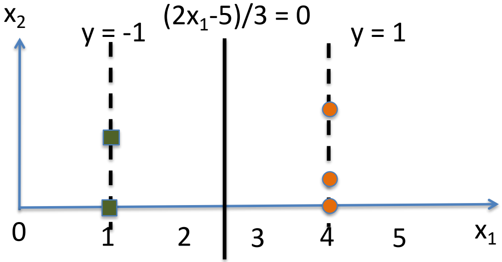
Is this the right scaling of $\mathbf{w}$ and b? We need $\min_n y_n(\mathbf{w}^\top \mathbf{x}_n + b) = 1$...

The correct parameter should be $\mathbf{w} = [2/3, 0]^\top$, and $b = -5/3$.

For example, for $\mathbf{x}_n = [1, 0]^\top$, we have

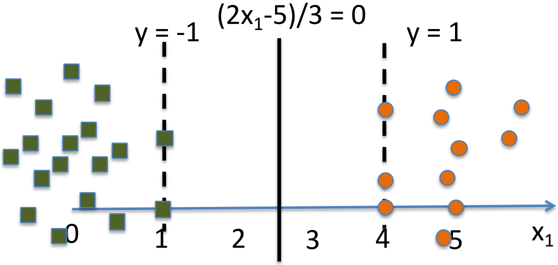
$$y_n(\mathbf{w}^\top \mathbf{x}_n + b) = (-1)[2/3 - 5/3] = 1.$$

Example of SVM: Support Vectors



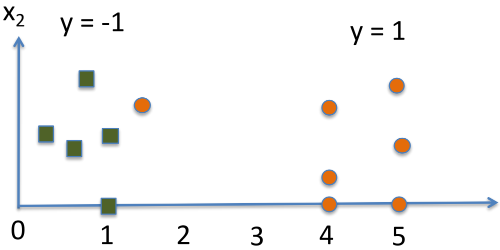
The solution to our optimization problem will be the **same** to the *reduced* dataset containing all the support vectors.

Example of SVM: Support Vectors



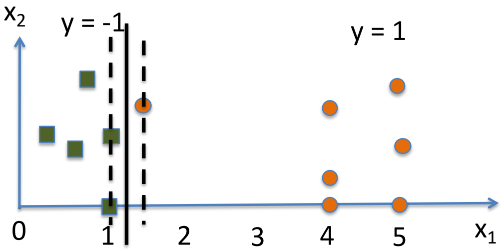
There can be many more data than the number of support vectors (so we can train on a smaller dataset).

Example of SVM: Resilience to Outliers



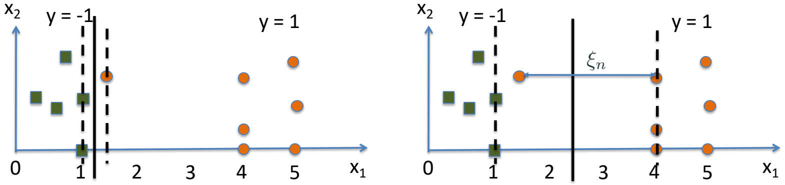
- Still linearly separable, but one of the orange dots is an "outlier".

Example of SVM: Resilience to Outliers



- Naively applying the hard-margin SVM will result in a classifier with small margin.
- So, better to use the soft-margin (or equivalently, hinge loss) formulation.

Example of SVM: Resilience to Outliers



$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_n \xi_n$$

$$\text{s.t.} \quad y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n, \quad \forall \quad n$$

$$\xi_n \geq 0, \quad \forall \ n$$

We allow the outlier to violate the constraint by ξ_n which we penalize.

- Small $C \Rightarrow$ more constraint violation, less sensitivity to outliers; but also (potentially) worse accuracy as more points are misclassified.

- $C = +\infty$ corresponds to hard margin SVM.

Advantages of SVM

So far, shown SVM:

1. Maximizes distance of training data from the boundary.
2. Only requires a subset of the training points.
3. Is less sensitive to outliers.
4. Scales better with high-dimensional data.
5. Generalizes well to many nonlinear models.

We will need to use duality to show the last two properties.

Outline

1. Review of Max Margin SVM Formulation
2. SVM: Hinge Loss Formulation
3. SVM: Example
4. A Dual View of SVMs (the short version)
5. Kernel SVM

A Dual View of SVMs (the short version)

What Is Duality?

Duality is a way of transforming a constrained optimization problem.

It tells us sometimes-useful information about the problem structure, and can sometimes make the problem easier to solve.

- Under strong duality condition (details are beyond the scope...), primal and dual problems are equivalent.
- Further, due to complementary slackness, dual variables tell us whether constraints are met with $=$ or $<$
- The strong duality condition is not always true for all optimiztion problems, but is true for the soft-margin SVM problem.

Instead of solving the max margin (primal) formulation, we solve its dual problem which will have certain advantages we will see.

Derivation of the Dual

Here is a skeleton of how to derive the dual problem.

Recipe

1. Formulate the generalized Lagrangian function (we'll define this on the next slide) that incorporates the constraints and introduces dual variables
2. Minimize the Lagrangian function over the primal variables
3. Plug in the primal variables from the previous step into the Lagrangian to get the dual function
4. Maximize the dual function with respect to dual variables
5. Recover the solution (for the primal variables) from the dual variables

Deriving the Dual for SVM

Primal SVM

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

The constraints are equivalent to the following canonical forms:

$$
-\xi_n \leq 0 \quad \text{and} \quad 1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] - \xi_n \leq 0
$$

Lagrangian

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

under the constraints that $\alpha_n \geq 0$ and $\lambda_n \geq 0$.

Deriving the Dual of SVM

Lagrangian

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

under the constraints that $\alpha_n \geq 0$ and $\lambda_n \geq 0$.

- Primal variables: $\boldsymbol{w}$, $\{\xi_n\}$, b; dual variables $\{\lambda_n\}$, $\{\alpha_n\}$
- Minimize the Lagrangian function over the primal variables by setting $\frac{\partial L}{\partial \boldsymbol{w}} = 0$, $\frac{\partial L}{\partial b} = 0$, and $\frac{\partial L}{\partial \xi_n} = 0$.
- Substitute primal variables from the above into the Lagrangian to get the dual function.
- Maximize the dual function with respect to dual variables
- After some further math and simplifications, we have...

Dual Formulation of SVM

Dual is also a convex quadratic program

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

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

$$\sum_n \alpha_n y_n = 0$$

- There are N dual variables α_n, one for each data point
- Independent of the size d of $\mathbf{x}$: SVM scales better for high-dimensional features.
- May seem like a lot of optimization variables when N is large, but many of the α_n's become zero. α_n is non-zero only if the n^{th} point is a support vector

Why Do Many α_n's Become Zero?

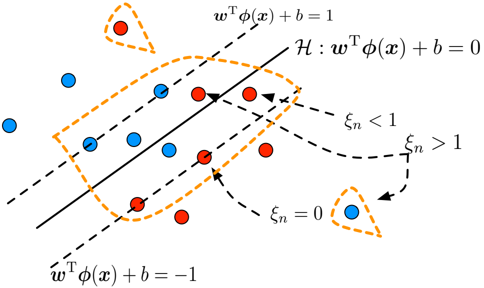
$$
\begin{aligned}
\max_{\boldsymbol{\alpha}} \quad & \sum_n \alpha_n - \frac{1}{2}\sum_{m,n} y_m y_n \alpha_m \alpha_n \boldsymbol{x}_m^\top \boldsymbol{x}_n \\
\text{s.t.} \quad & 0 \le \alpha_n \le C, \quad \forall\ n \\
& \sum_n \alpha_n y_n = 0
\end{aligned}
$$

- By complementary slackness:

$$\alpha_n\{1 - \xi_n - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b]\} = 0 \quad \forall n$$

- This tells us that $\alpha_n > 0$ only when $1 - \xi_n = y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b]$, i.e. (x_n, y_n) is a support vector. So most of the α_n is zero, and the only non-zero α_n are for the support vectors.
- Further, $\alpha_n < C$ only when $\xi_n = 0$. (The derivation of this is beyond the scope of today's lecture)

Visualizing the Support Vectors



- $\alpha_n = 0$: non-support vector.
- $0 < \alpha_n < C$: support vector with $\xi_n = 0$, i.e. $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] = 1$, distance to boundary $\frac{1}{\|w\|}$.
- $\alpha_n = C$: support vector with $\xi_n > 0$, hence $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] < 1$.

How to Get w and b?

Lagrangian

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

Recovering **w**

$$\frac{\partial L}{\partial \boldsymbol{w}} = 0 \rightarrow \boldsymbol{w} = \sum_n \alpha_n y_n \boldsymbol{x}_n$$

Only depends on support vectors, i.e., points with $\alpha_n > 0$!

Recovering b

Find a sample (x_n, y_n) such that $0 < \alpha_n < C$. Using $y_n \in \{-1, 1\}$,

$$y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] = 1$$
$$b = y_n - \boldsymbol{w}^\top \boldsymbol{x}_n$$
$$b = y_n - \sum_m \alpha_m y_m \boldsymbol{x}_m^\top \boldsymbol{x}_n$$

Summary of Dual Formulation

Primal Max-Margin Formulation

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_n \xi_n$$

$$\text{s.t.} \quad y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n, \quad \forall \ n$$

$$\xi_n \geq 0, \quad \forall \ n$$

Dual Formulation

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

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

$$\sum_n \alpha_n y_n = 0$$

- In dual formulation, the # of variables is independent of dimension.
- Most of the dual variables are 0, and the non-zero ones are the support vectors.
- Can easily recover the primal solution $\boldsymbol{w}, b$ from dual solution.

Advantages of SVM

We have shown SVM:

1. Maximizes distance of training data from the boundary
2. Only requires a subset of the training points.
3. Is less sensitive to outliers.
4. Scales better with high-dimensional data.
5. Generalizes well to many nonlinear models.

Kernel SVM

Non-linear Basis Functions in SVM

- What if the true decision boundary is not linear?
- Similar to linear regression, we can transform the feature vector **x** using non-linear basis functions. For example,

$$\phi(\boldsymbol{x}) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1 x_2 \\ x_1^2 \\ x_2^2 \end{bmatrix}$$

- Replace **x** by $\phi(\mathbf{x})$ in both the primal and dual SVM formulations

Primal and Dual SVM Formulations: Kernel Versions

Primal

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_n \xi_n$$

$$\text{s.t.} \quad y_n[\boldsymbol{w}^\top \phi(\boldsymbol{x}_n) + b] \geq 1 - \xi_n, \quad \forall \;\; n$$

$$\xi_n \geq 0, \quad \forall \; n$$

Dual

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

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

$$\sum_n \alpha_n y_n = 0$$

IMPORTANT POINT: In the dual problem, we only need $\phi(\boldsymbol{x}_m)^\top \phi(\boldsymbol{x}_n)$.

Dual Kernel SVM

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

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

What is a kernel function?

- $k(\mathbf{x}_m, \mathbf{x}_n)$ is a scalar-valued function that measures the similarity of $\mathbf{x}_m$ and $\mathbf{x}_n$.
- $k(\mathbf{x}_m, \mathbf{x}_n)$ is a valid kernel function if it is symmetric and positive-definite. This ensures that there exists a $\phi(\mathbf{x})$ (even if we don't know what it is) such that $k(\mathbf{x}_m, \mathbf{x}_n) = \phi(\mathbf{x}_m)^\top \phi(\mathbf{x}_n)$.

Examples of Popular Kernel Functions

Here are some example kernel functions and the corresponding features.

- Dot product:

$$k(\mathbf{x}_m, \mathbf{x}_n) = \mathbf{x}_m^\top \mathbf{x}_n, \text{ corresponding } \phi(\mathbf{x}) = \mathbf{x}$$

- Dot product with positive-definite matrix $\mathbf{Q}$:

$$k(\mathbf{x}_m, \mathbf{x}_n) = \mathbf{x}_m^\top \mathbf{Q} \mathbf{x}_n, \text{ corresponding } \phi(\mathbf{x}) = \mathbf{Q}^{1/2}\mathbf{x}$$

- Polynomial kernels (corresponding $\phi(\mathbf{x})$ complicated):

$$k(\mathbf{x}_m, \mathbf{x}_n) = (1 + \mathbf{x}_m^\top \mathbf{x}_n)^d, \quad d \in \mathbb{Z}^+$$

- Radial basis kernel (corresponding $\phi(\mathbf{x})$ complicated):

$$k(\mathbf{x}_m, \mathbf{x}_n) = \exp\left(-\gamma \left\|\mathbf{x}_m - \mathbf{x}_n\right\|^2\right) \text{ for some } \gamma > 0$$

and many more.

The Kernel Trick

In dual SVM, we can use any of the kernel functions discussed in the previous slide.

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

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

$$\sum_n \alpha_n y_n = 0$$

Each choice of kernel function will correspond to doing SVM using the transformed data $\phi(\boldsymbol{x})$, but we do not need to know what exactly is $\phi(\boldsymbol{x})$.

This is allows us using more complicated $\phi(\boldsymbol{x})$ (like the $\phi(\boldsymbol{x})$ associated with radial basis function) to boost performance - without knowing what $\phi(\boldsymbol{x})$ is! This is known as "kernal trick".

Test Prediction

Learning $\boldsymbol{w}$ and b:

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

$$b = y_n - \boldsymbol{w}^\top \phi(\boldsymbol{x}_n) = y_n - \sum_m \alpha_m y_m k(\boldsymbol{x}_m, \boldsymbol{x}_n)$$

But for test prediction on a new point $\mathbf{x}$, do we need the form of $\phi(\mathbf{x})$ in order to find the sign of $\boldsymbol{w}^\top \phi(\boldsymbol{x}) + b$? Fortunately, no!

Test Prediction:

$$h(\boldsymbol{x}) = \text{SIGN}(\sum_n y_n \alpha_n k(\boldsymbol{x}_n, \boldsymbol{x}) + b)$$

At test time it suffices to know the kernel function! So we really do not need to know ϕ.

Summary of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n) \text{ for } n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

Select a kernel. In general, you can just use one of the popular kernel functions (polynomial kernel or radial kernel).

Training

$$\max_{\boldsymbol{\alpha}} \sum_n \alpha_n - \frac{1}{2}\sum_{m,n} y_m y_n \alpha_m \alpha_n k(\mathbf{x}_m, \mathbf{x}_m)$$

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

$$\sum_n \alpha_n y_n = 0$$

Prediction

$$h(\boldsymbol{x}) = \text{SIGN}\left(\sum_n y_n \alpha_n k(\boldsymbol{x}_n, \boldsymbol{x}) + b\right)$$

Example of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n) \text{ for } n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

What if the data is not linearly separable?

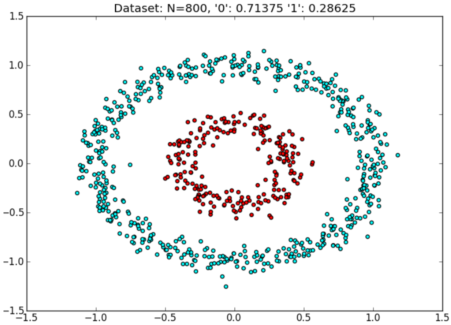
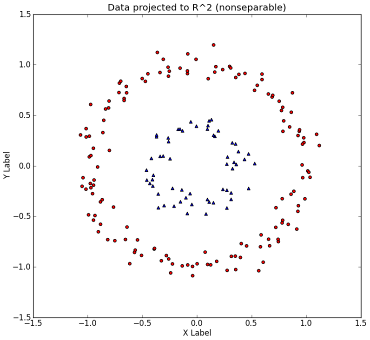


Image Source: `https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`

Example of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n)$ for $n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

Use feature $\phi(x) = [x_1, x_2, x_1^2 + x_2^2]$ to transform the data in a 3D space



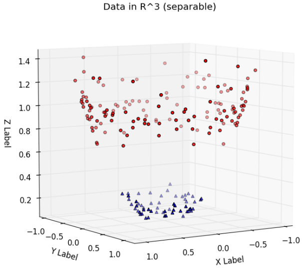


Image Source: `https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`

Example of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n)$ for $n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

Then find the decision boundary. How? Solve the dual problem!

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

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

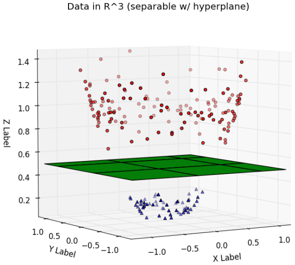
$$\sum_n \alpha_n y_n = 0$$

Then find $\mathbf{w}$ and b. Predict $y = \text{sign}(\mathbf{w}^T \phi(\mathbf{x}) + b)$.

Example of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n)$ for $n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

Here is the resulting decision boundary



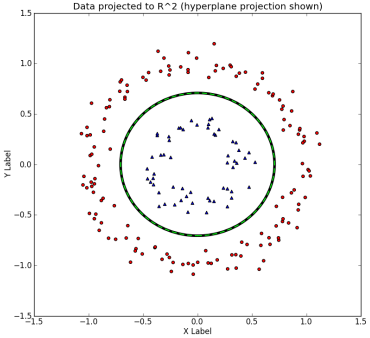


Image Source: `https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`

Example of Kernel SVM

In the previous example, we manually defined a $\phi(\mathbf{x})$.

As mentioned in the "kernel trick" slides, in general you don't need to concretely define $\phi(\mathbf{x})$. We could select a kernel function $k(\mathbf{x}_m, \mathbf{x}_n)$ and solve the following dual SVM.

$$\max_{\boldsymbol{\alpha}} \sum_n \alpha_n - \frac{1}{2}\sum_{m,n} y_m y_n \alpha_m \alpha_n k(\mathbf{x}_m, \mathbf{x}_n)$$

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

$$\sum_n \alpha_n y_n = 0$$

Test Prediction also only uses kernel:

$$h(\boldsymbol{x}) = \text{SIGN}(\sum_n y_n \alpha_n k(\boldsymbol{x}_n, \boldsymbol{x}) + b)$$

Example of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n) \text{ for } n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

Effect of the choice of kernel: Radial Basis Kernel

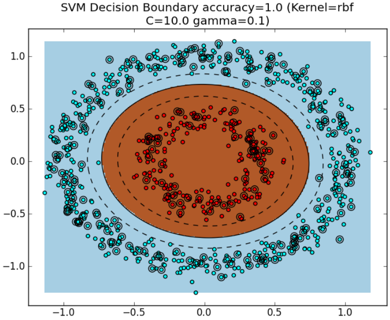


Image Source: `https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`

Advantages of SVM

Now we have shown all of the below.

1. Maximizes distance of training data from the boundary
2. Only requires a subset of the training points.
3. Is less sensitive to outliers.
4. Scales better with high-dimensional data.
5. Generalizes well to many nonlinear models.