18-661 Introduction to Machine Learning

Support Vector Machines (SVM) – I

Spring 2023

ECE – Carnegie Mellon University

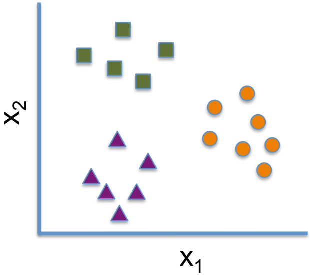
Outline

1. Review of Multi-class Logistic Regression
2. Support Vector Machines (SVM): Intuition
3. SVM: Max-Margin Formulation
4. SVM: Hinge Loss Formulation
5. Summary

Review of Multi-class Logistic Regression

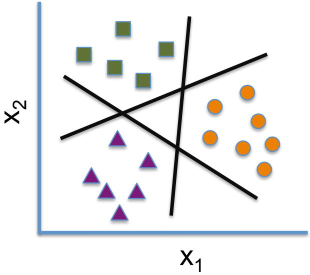
Three Approaches

- One-versus-all
- One-versus-one
- Multinomial regression



The One-versus-Rest or One-versus-All Approach

- For each class c, change the problem into binary classification
  1. Relabel training data with label c, into POSITIVE (or '1').
  2. Relabel all the rest data into NEGATIVE (or '0').
- Repeat this multiple times: Train C binary classifiers, using logistic regression to differentiate the two classes each time.
- There is ambiguity in some of the regions (the 4 triangular areas)...

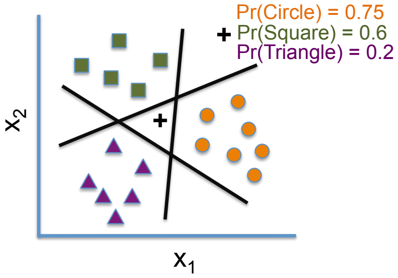


The One-versus-Rest or One-versus-All Approach

How to combine these linear decision boundaries?

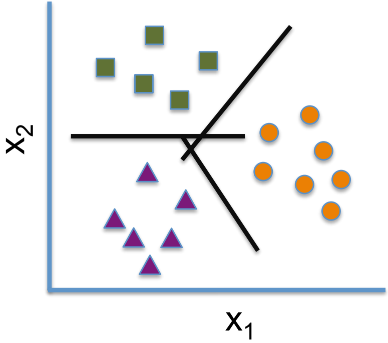
- Use the confidence estimates $\Pr(y = 1|\mathbf{x}) = \sigma(\mathbf{w}_1^\top \mathbf{x})$, $\ldots \Pr(y = C|\mathbf{x}) = \sigma(\mathbf{w}_C^\top \mathbf{x})$
- Declare class c^* that maximizes

$$c^* = \arg \max_{c=1,\ldots,C} \Pr(y = c|\mathbf{x}) = \sigma(\mathbf{w}_c^\top \mathbf{x})$$



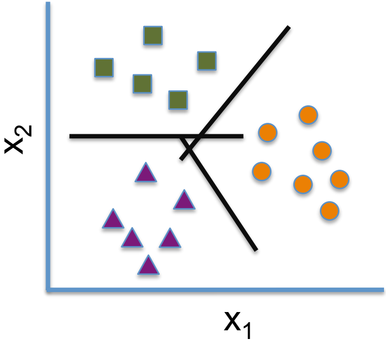
The One-versus-One Approach

- For each **pair** of classes c and c', change the problem into binary classification.
  1. Relabel training data with label c, into POSITIVE (or '1')
  2. Relabel training data with label c' into NEGATIVE (or '0')
  3. **Disregard** all other data



The One-versus-One Approach

- How many binary classifiers for C classes? $C(C-1)/2$
- How to combine their outputs?
- Given $\boldsymbol{x}$, count the $C(C-1)/2$ votes from outputs of all binary classifiers and declare the winner as the predicted class.
- Use confidence scores to resolve ties.



Multinomial Logistic Regression

- **Model:** For each class c, we have a parameter vector $\boldsymbol{w}_c$ and model the posterior probability as:

$$P(c|\boldsymbol{x}) = \frac{e^{\boldsymbol{w}_c^\top \boldsymbol{x}}}{\sum_{c'} e^{\boldsymbol{w}_{c'}^\top \boldsymbol{x}}} \qquad \leftarrow \qquad \text{This is called the } \textit{softmax} \text{ function.}$$

- **Decision boundary:** Assign $\boldsymbol{x}$ with the label that is the maximum of posterior:

$$\arg\max_c P(c|\boldsymbol{x}) \to \arg\max_c \boldsymbol{w}_c^\top \boldsymbol{x}.$$

Parameter Estimation for Multinomial Logistic Regression

Discriminative approach: Maximize conditional likelihood

$$\log P(\mathcal{D}) = \sum_n \log P(y_n | \boldsymbol{x}_n)$$

We will change y_n to $\boldsymbol{y}_n = [y_{n1}\ y_{n2}\ \cdots\ y_{nC}]^\top$, a C-dimensional vector using 1-of-C encoding.

$$y_{nc} = \begin{cases} 1 & \text{if } y_n = c \\ 0 & \text{otherwise} \end{cases}$$

Ex: if $y_n = 2$, then, $\boldsymbol{y}_n = [0\ 1\ 0\ 0\ \cdots\ 0]^\top$.

$$\Rightarrow \sum_n \log P(y_n | \boldsymbol{x}_n) = \sum_n \log \prod_{c=1}^{C} P(c|\boldsymbol{x}_n)^{y_{nc}} = \sum_n \sum_c y_{nc} \log P(c|\boldsymbol{x}_n)$$

Cross-entropy Error Function

Definition: negative log-likelihood

$$\mathcal{E}(\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_C) = -\sum_n \sum_c y_{nc} \log P(c|\boldsymbol{x}_n)$$
$$= -\sum_n \sum_c y_{nc} \log \left(\frac{e^{\boldsymbol{w}_c^\top \boldsymbol{x}_n}}{\sum_{c'} e^{\boldsymbol{w}_{c'}^\top \boldsymbol{x}_n}} \right)$$

Properties of cross-entropy

- Convex in the $\mathbf{w}_c$ vectors, therefore unique global optimum
- Optimization requires numerical procedures, analogous to those used for binary logistic regression.

Outline

1. Review of Multi-class Logistic Regression
2. Support Vector Machines (SVM): Intuition
3. SVM: Max-Margin Formulation
4. SVM: Hinge Loss Formulation
5. Summary

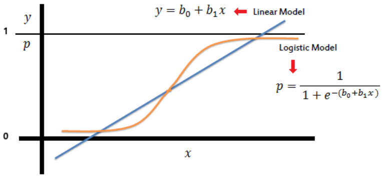
Support Vector Machines (SVM): Intuition

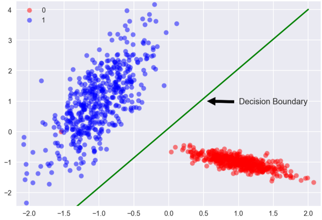
Why Do We Need SVM?

Alternative to Logistic Regression and Naïve Bayes.

- Logistic regression and Naïve Bayes train over the whole dataset.
- These can require a lot of memory in high-dimensional settings.
- SVM can give a better and more efficient solution.
- SVM is one of the most powerful and commonly used ML algorithms.

Binary Logistic Regression





- We only need to know if $p(\boldsymbol{x}) > 0.5$ or < 0.5.
- We don't (always) need to know how far $\boldsymbol{x}$ is from this boundary.

How can we use this insight to improve the classification algorithm?

- What if we just looked at the boundary?
- Maybe then we could ignore some of the samples?

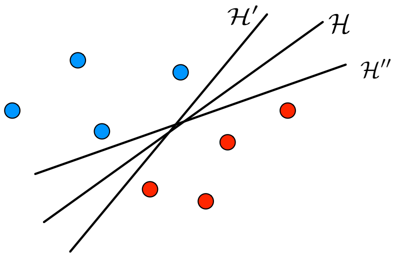
Advantages of SVM

We will see later that SVM:

1. Maximizes distance of training points 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.

SVM: Max-Margin Formulation

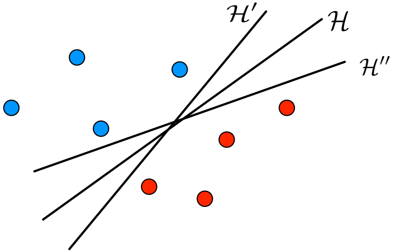
Binary Classification: Finding a Linear Decision Boundary



- Input features $\boldsymbol{x}$.
- Decision boundary is a hyperplane $\mathcal{H} : \boldsymbol{w}^\top \boldsymbol{x} + b = 0$.

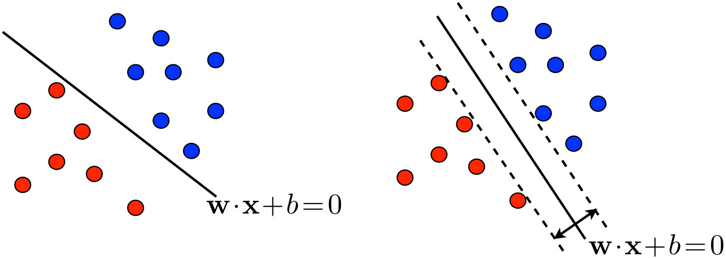
Intuition: Where to Put the Decision Boundary?

- 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?

Intuition: Where to Put the Decision Boundary?



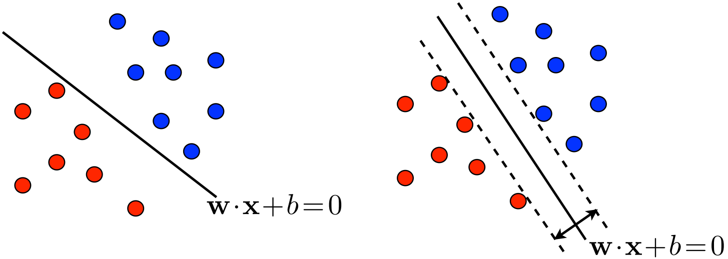
Find a decision boundary in the '*middle*' of the two classes that:

- Perfectly classifies the training data
- Is as far away from every training point as possible

Let us apply this intuition to build a classifier that maximizes the margin between training points and the decision boundary.

First, Some Vector Geometry

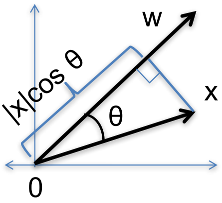
What is a hyperplane?



- General equation is $\boldsymbol{w}^\top \boldsymbol{x} + b = 0$
- Divides the space in half, i.e., $\boldsymbol{w}^\top \boldsymbol{x} + b > 0$ and $\boldsymbol{w}^\top \boldsymbol{x} + b < 0$
- A hyperplane is a line in 2D and a plane in 3D
- $\boldsymbol{w} \in \mathbb{R}^d$ is a non-zero normal vector

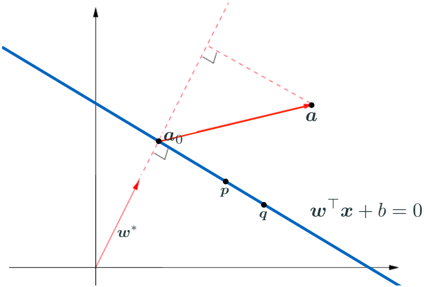
Vector Norms and Inner Products

- Given two vectors $\mathbf{w}$ and $\mathbf{x}$, what is their inner product?
- Inner Product $\mathbf{w}^\top \mathbf{x} = w_1 x_1 + w_2 x_2 + \cdots + w_d x_d$



- Inner Product $\mathbf{w}^\top \mathbf{x}$ is also equal to $\|\boldsymbol{w}\| \, \|\boldsymbol{x}\| \cos\theta$
- $\mathbf{w}^\top \mathbf{w} = \|\boldsymbol{w}\|^2$
- If $\mathbf{w}$ and $\mathbf{x}$ are perpendicular, then $\theta = \pi/2$, and thus the inner product is zero.

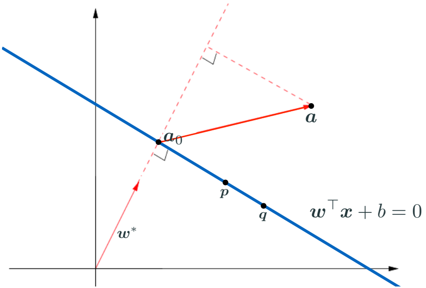
Normal Vector of a Hyperplane



Vector $\boldsymbol{w}$ is normal to the hyperplane. Why?

- If $\boldsymbol{p}$ and $\boldsymbol{q}$ are both on the line, then $\boldsymbol{w}^\top \boldsymbol{p} + b = \boldsymbol{w}^\top \boldsymbol{q} + b = 0$.
- Then $\boldsymbol{w}^\top(\boldsymbol{p} - \boldsymbol{q}) = \boldsymbol{w}^\top \boldsymbol{p} - \boldsymbol{w}^\top \boldsymbol{q} = -b - (-b) = 0$
- $\boldsymbol{p} - \boldsymbol{q}$ is an arbitrary vector parallel to the line, thus $\boldsymbol{w}$ is orthogonal
- $\boldsymbol{w}^* = \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|_2}$ is the unit normal vector

Distance from a Hyperplane



How to find the distance from $\boldsymbol{a}$ to the hyperplane?

- We want to find distance between $\boldsymbol{a}$ and line in the direction of $\boldsymbol{w}^*$.
- If we define point $\boldsymbol{a}_0$ on the line, then this distance corresponds to length of $\boldsymbol{a} - \boldsymbol{a}_0$ in direction of $\boldsymbol{w}^*$, which equals ${\boldsymbol{w}^*}^\top(\boldsymbol{a} - \boldsymbol{a}_0)$.
- We know $\boldsymbol{w}^\top \boldsymbol{a}_0 = -b$ since $\boldsymbol{w}^\top \boldsymbol{a}_0 + b = 0$.
- Then the distance equals $\frac{1}{\|\boldsymbol{w}\|_2}(\boldsymbol{w}^\top \boldsymbol{a} + b)$.

Distance from a Point to Decision Boundary

The *unsigned* distance from a point $\boldsymbol{x}$ to the decision boundary (hyperplane) $\mathcal{H}$ is

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

How to remove the absolute value $|\cdot|$?

Notation changes from Logistic Regression: Use $y = +1$ to represent positive label and $y = -1$ for negative label.

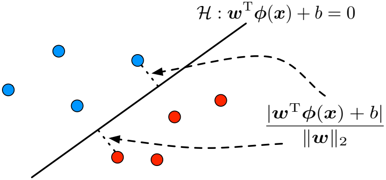
Then, exploiting the fact that the decision boundary classifies every point in the training dataset correctly, we have $(\boldsymbol{w}^\top \boldsymbol{x} + b)$ and $\boldsymbol{x}$'s label y must have the same sign. So we get

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

Defining the Margin

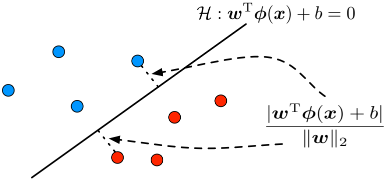
Margin
Smallest distance between the hyperplane and all training points

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



How can we use this to find the SVM solution?

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).

Scale of w

Margin
Smallest distance between the hyperplane and all training points

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

Consider three hyperplanes

$$(\boldsymbol{w}, b) \quad (2\boldsymbol{w}, 2b) \quad (.5\boldsymbol{w}, .5b)$$

Which one has the largest margin?

- The $\textsc{margin}$ doesn't change if we scale $(\boldsymbol{w}, b)$ by a constant c
- $\boldsymbol{w}^\top \boldsymbol{x} + b = 0$ and $(c\boldsymbol{w})^\top \boldsymbol{x} + (cb) = 0$: same decision boundary!
- Can we further constrain the problem so as to get a unique solution $(\boldsymbol{w}, b)$?

Rescaled Margin

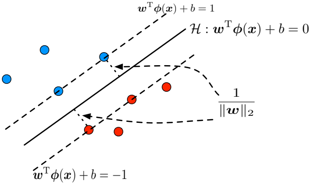
We can further constrain the problem by scaling $(\boldsymbol{w}, b)$ such that

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

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

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

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

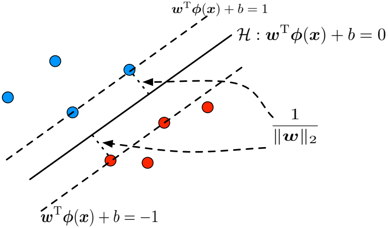
$$\begin{aligned} \min_{\boldsymbol{w},b} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 \\ \text{s.t.} \quad & y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1, \quad \forall \quad n \end{aligned}$$

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

Support Vectors: A First Look

SVM formulation for separable data

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



Two types of training data, based on the situations of the constraint:

- "$=$": $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] = 1$. These training data points are called "support vectors", which have the minimum distance ($\frac{1}{\|\boldsymbol{w}\|}$) to the boundary.
- "$>$": $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] > 1$. Distance to the boundary is larger than the minimum. Removing these data points does not affect the optimal solution (more on this next lecture).

SVM for Non-separable Data

SVM formulation for separable data

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

Non-separable setting
In practice our training data may not be separable. What issues arise with the optimization problem above when data is not separable?

- For every $\boldsymbol{w}$ there exists a training point $\boldsymbol{x}_i$ such that

$$y_i[\boldsymbol{w}^\top \boldsymbol{x}_i + b] \leq 0$$

- There is no feasible $(\boldsymbol{w}, b)$ as at least one of our constraints is violated!

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?

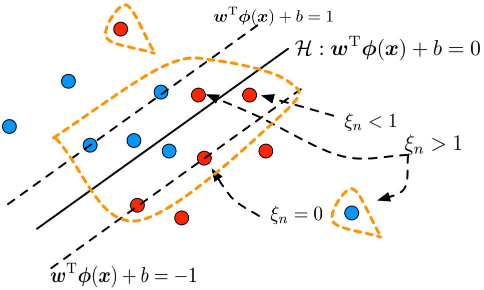
- User-defined hyperparameter
- Trades off between the two terms in our objective
- Same idea as the regularization term in ridge regression

How to Solve this Problem?

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

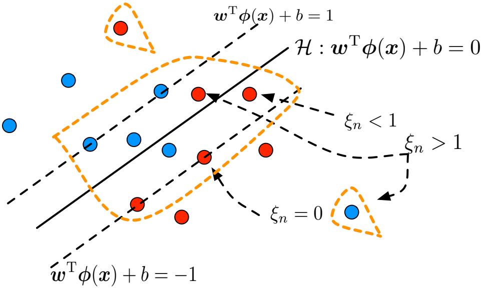
- This is a convex quadratic program: the objective function is quadratic in $\boldsymbol{w}$ and linear in $\boldsymbol{\xi}$ and the constraints are linear (inequality) constraints in $\boldsymbol{w}$, b and ξ_n.
- We can solve the optimization problem using general-purpose solvers, e.g., Matlab's `quadprog()` function, python's `scipy.optimize` package or CVXPY package.

Support Vectors: Revisit



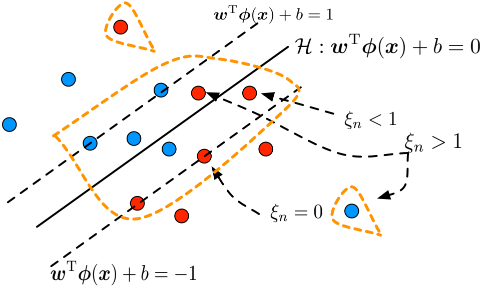
Support vectors are highlighted by the dotted orange lines. What does this mean mathematically?

Support Vectors: Revisit



Recall the constraints $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n$ from the soft-margin formulation. All the training points $(\boldsymbol{x}_n, y_n)$ that satisfies the constraint with "$=$" are support vectors.

Support Vectors: Revisit



In other words, support vectors satisfy $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] = 1 - \xi_n$, which can be further divided into several categories:

- $\xi_n = 0$: $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] = 1$, 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.

Outline

1. Review of Multi-class Logistic Regression
2. Support Vector Machines (SVM): Intuition
3. SVM: Max-Margin Formulation
4. SVM: Hinge Loss Formulation
5. Summary

SVM: Hinge Loss Formulation

SVM vs. Logistic Regression

SVM soft-margin formulation

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

Logistic regression formulation

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

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

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

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 get Hinge Loss formulation:

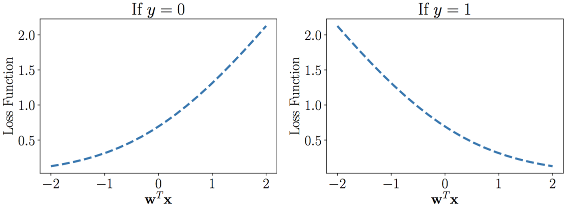
$$\min_{\boldsymbol{w},b} \ \sum_n \underbrace{\max(0, 1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b])}_{\text{Hinge Loss for } x_n, y_n} + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$$

Logistic Regression Loss vs Hinge Loss

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

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

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

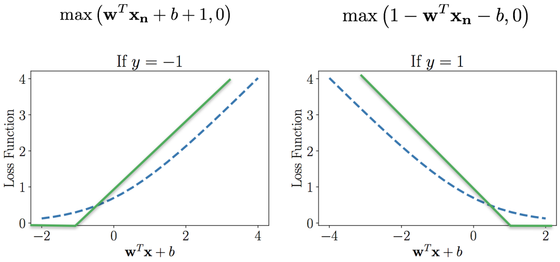


- How does the Hinge Loss Function look like?

Logistic Regression 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.

Summary

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)

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

Summary

You should know:

- Max-margin formulation for separable and non-separable SVMs.
- Definition and importance of support vectors.
- Hinge loss formulation of SVMs.
- Equivalence of the max-margin and hinge loss formulations.