18-661 Introduction to Machine Learning

Support Vector Machines (SVM) – I

Spring 2020

ECE – Carnegie Mellon University

Midterm Information

Midterm will be on Wednesday, 2/26 in-class.

- Closed-book except for one double-sided letter-size handwritten page of notes that you can prepare as you wish.
- We will provide formulas for relevant probability distributions.
- You will not need a calculator. Only pen/pencil and scratch paper are allowed.

Will cover all topics presented through next Wednesday in class (SVM and before).

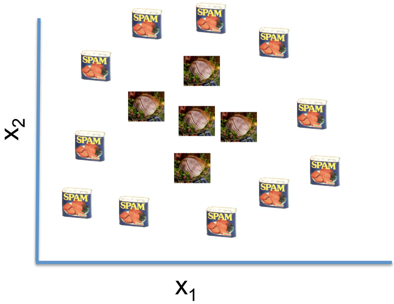
- (1) point estimation/MLE/MAP, (2) linear regression, (3) naive Bayes, (4) logistic regression, and (5) SVMs.
- Next friday's recitation will go over practice questions.
- Understand all homework questions and derivations in lecture/recitation.

Outline

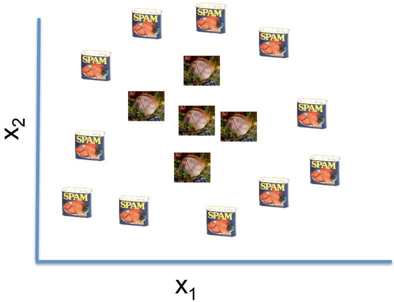
1. Review of Non-linear classification boundary
2. Review of Multi-class Logistic Regression
3. Support Vector Machines (SVM): Intuition
4. SVM: Max Margin Formulation
5. SVM: Hinge Loss Formulation
6. Equivalence of These Two Formulations

Review of Non-linear classification boundary

How to handle more complex decision boundaries?



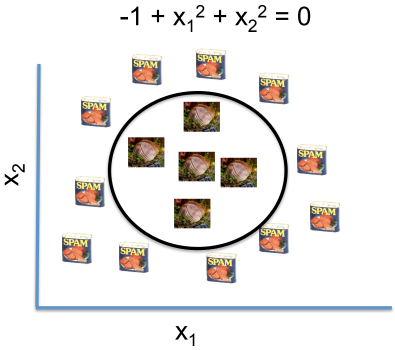
How to handle more complex decision boundaries?



- This data is not linearly separable in the original feature space
- Use non-linear basis functions to add more features, hopefully it becomes linearly separable in the "augmented" space.

Adding polynomial features

- New feature vector is $\mathbf{x} = [1, x_1, x_2, x_1^2, x_2^2]$
- $\Pr(y = 1|\mathbf{x}) = \sigma(w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2)$
- If $\mathbf{w} = [-1, 0, 0, 1, 1]$, the boundary is $-1 + x_1^2 + x_2^2 = 0$
  - If $-1 + x_1^2 + x_2^2 \geq 0$ declare spam
  - If $-1 + x_1^2 + x_2^2 < 0$ declare ham



Solution to Overfitting: Regularization

- Add regularization term to be cross entropy loss function

$$\mathcal{E}(\boldsymbol{w}) = -\sum_n \{y_n \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1-y_n)\log[1-\sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]\} + \underbrace{\frac{1}{2}\lambda\|w\|_2^2}_{\text{regularization}}$$

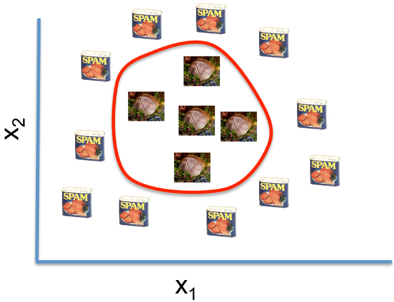
- Perform gradient descent on this regularized function
- Often, we do NOT regularize the bias term w_0

Solution to Overfitting: Regularization

- Add regularization term to be cross entropy loss function

$$\mathcal{E}(\boldsymbol{w}) = -\sum_n \{y_n \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1-y_n)\log[1-\sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]\} + \underbrace{\frac{1}{2}\lambda\|w\|_2^2}_{\text{regularization}}$$

- Perform gradient descent on this regularized function
- Often, we do NOT regularize the bias term w_0



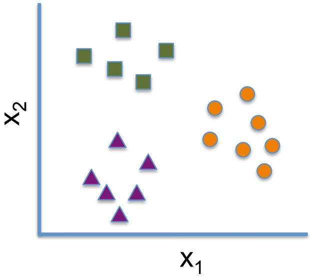
Outline

1. Review of Non-linear classification boundary
2. Review of Multi-class Logistic Regression
3. Support Vector Machines (SVM): Intuition
4. SVM: Max Margin Formulation
5. SVM: Hinge Loss Formulation
6. Equivalence of These Two Formulations

Review of Multi-class Logistic Regression

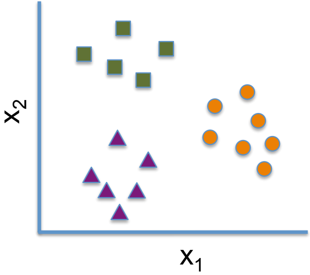
Three approaches

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



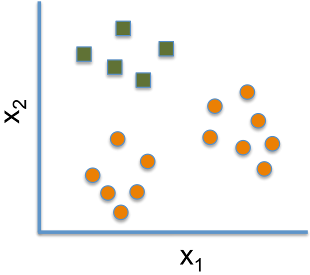
The One-versus-Rest or One-Versus-All Approach

- For each class C_k, change the problem into binary classification
  1. Relabel training data with label C_k, into POSITIVE (or '1')
  2. Relabel all the rest data into NEGATIVE (or '0')
- Repeat this multiple times: Train K binary classifiers, using logistic regression to differentiate the two classes each time



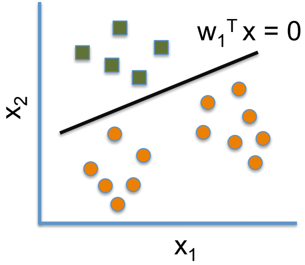
The One-versus-Rest or One-Versus-All Approach

- For each class C_k, change the problem into binary classification
  1. Relabel training data with label C_k, into POSITIVE (or '1')
  2. Relabel all the rest data into NEGATIVE (or '0')
- Repeat this multiple times: Train K binary classifiers, using logistic regression to differentiate the two classes each time



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

- For each class C_k, change the problem into binary classification
  1. Relabel training data with label C_k, into POSITIVE (or '1')
  2. Relabel all the rest data into NEGATIVE (or '0')
- Repeat this multiple times: Train K binary classifiers, using logistic regression to differentiate the two classes each time

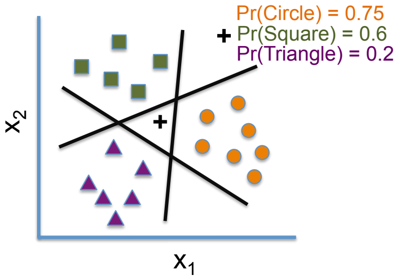


The One-versus-Rest or One-Versus-All Approach

How to combine these linear decision boundaries?

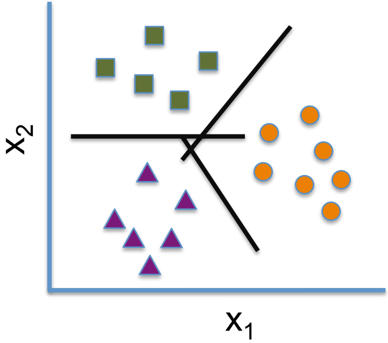
- Use the confidence estimates $\Pr(y = C_1|\mathbf{x}) = \sigma(\mathbf{w}_1^\top \mathbf{x})$, $\ldots \Pr(y = C_K|\mathbf{x}) = \sigma(\mathbf{w}_K^\top \mathbf{x})$
- Declare class C_k^* that maximizes

$$k^* = \arg \max_{k=1,\ldots,K} \Pr(y = C_k|\mathbf{x}) = \sigma(\mathbf{w}_k^\top \mathbf{x})$$



The One-Versus-One Approach

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



The One-Versus-One Approach

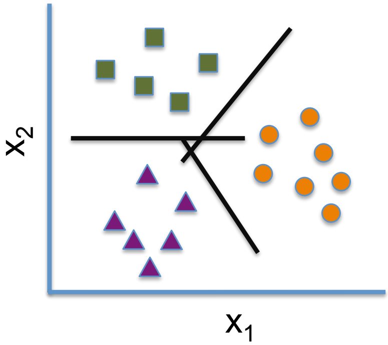
- How many binary classifiers for K classes?

The One-Versus-One Approach

- How many binary classifiers for K classes?

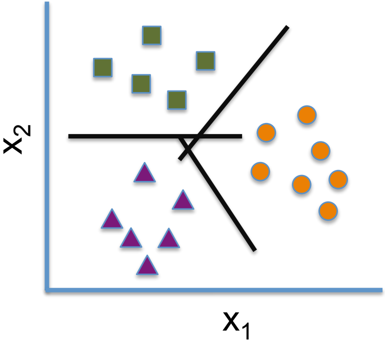
The One-Versus-One Approach

- How many binary classifiers for K classes? $K(K-1)/2$
- How to combine their outputs?



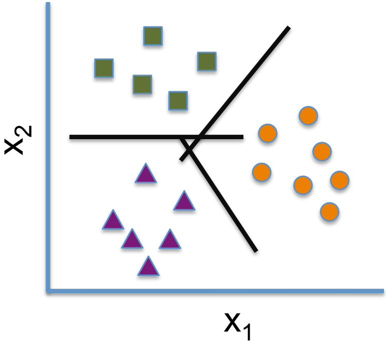
The One-Versus-One Approach

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



The One-Versus-One Approach

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



Multinomial logistic regression (Perceptron)

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

$$p(C_k|\boldsymbol{x}) = \frac{e^{\boldsymbol{w}_k^\top \boldsymbol{x}}}{\sum_{k'} e^{\boldsymbol{w}_{k'}^\top \boldsymbol{x}}} \qquad \leftarrow \qquad \text{This is called } \textit{softmax} \text{ function}$$

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

$$\arg\max_k P(C_k|\boldsymbol{x}) \rightarrow \arg\max_k \boldsymbol{w}_k^\top \boldsymbol{x}.$$

Parameter estimation

Discriminative approach: maximize conditional likelihood

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

Parameter estimation

Discriminative approach: maximize conditional likelihood

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

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

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

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

Parameter estimation

Discriminative approach: maximize conditional likelihood

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

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

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

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

$$\begin{aligned} P(y_n|\boldsymbol{x}_n) &= \prod_{k=1}^{K} P(C_k|\boldsymbol{x}_n)^{y_{nk}} \\ &= P(C_1|\boldsymbol{x}_n)^{y_{n1}} P(C_2|\boldsymbol{x}_n)^{y_{n2}} \cdots P(C_K|\boldsymbol{x}_n)^{y_{nK}} \end{aligned}$$

therefore, only the term corresponding to $y_{nk} = 1$ will survive.

Cross-entropy error function

$$\sum_n \log P(y_n|\boldsymbol{x}_n) = \sum_n \log \prod_{k=1}^{K} P(C_k|\boldsymbol{x}_n)^{y_{nk}} = \sum_n \sum_k y_{nk} \log P(C_k|\boldsymbol{x}_n)$$

Definition: negative log likelihood

$$\begin{aligned} \mathcal{E}(\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_K) &= -\sum_n \sum_k y_{nk} \log P(C_k|\boldsymbol{x}_n) \\ &= -\sum_n \sum_k y_{nk} \log \left(\frac{e^{\boldsymbol{w}_k^\top \boldsymbol{x}_n}}{\sum_{k'} e^{\boldsymbol{w}_{k'}^\top \boldsymbol{x}_n}} \right) \end{aligned}$$

Cross-entropy error function

$$\sum_n \log P(y_n|\boldsymbol{x}_n) = \sum_n \log \prod_{k=1}^{K} P(C_k|\boldsymbol{x}_n)^{y_{nk}} = \sum_n \sum_k y_{nk} \log P(C_k|\boldsymbol{x}_n)$$

Definition: negative log likelihood

$$\begin{aligned}\mathcal{E}(\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_K) &= -\sum_n \sum_k y_{nk} \log P(C_k|\boldsymbol{x}_n) \\ &= -\sum_n \sum_k y_{nk} \log \left(\frac{e^{\boldsymbol{w}_k^\top \boldsymbol{x}_n}}{\sum_{k'} e^{\boldsymbol{w}_{k'}^\top \boldsymbol{x}_n}} \right)\end{aligned}$$

Properties

- Convex, therefore unique global optimum
- Optimization requires numerical procedures, analogous to those used for binary logistic regression

Outline

1. Review of Non-linear classification boundary
2. Review of Multi-class Logistic Regression
3. Support Vector Machines (SVM): Intuition
4. SVM: Max Margin Formulation
5. SVM: Hinge Loss Formulation
6. Equivalence of These Two Formulations

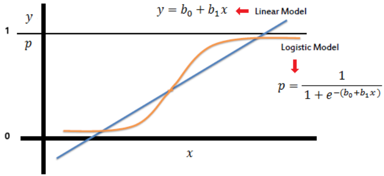
Support Vector Machines (SVM): Intuition

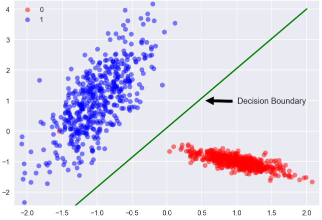
Why do we need SVM?

Alternative to Logistic Regression and Naive Bayes.

- Logistic regression and naive 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:

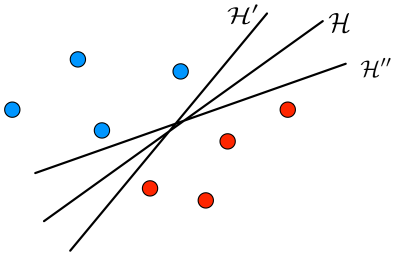
1. Is less sensitive to outliers.
2. Maximizes distance of training points from the boundary
3. Scales better with high-dimensional data.
4. Only requires a subset of the training points.
5. Generalizes well to many nonlinear models.

Outline

1. Review of Non-linear classification boundary
2. Review of Multi-class Logistic Regression
3. Support Vector Machines (SVM): Intuition
4. SVM: Max Margin Formulation
5. SVM: Hinge Loss Formulation
6. Equivalence of These Two Formulations

SVM: Max Margin Formulation

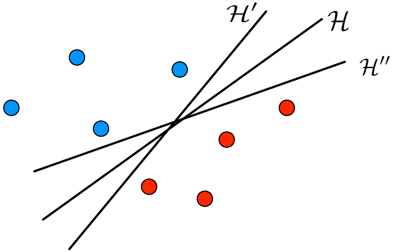
Binary Classification: Finding a Linear Decision Boundary



- Input features $\boldsymbol{x}$.
- Decision boundary is a hyperplane $\mathcal{H} : \boldsymbol{w}^\top \boldsymbol{x} + b = 0$.
- All $\boldsymbol{x}$ satisfying $\boldsymbol{w}^\top \boldsymbol{x} + b < 0$ lie on the same side of the line and are in the same "class."

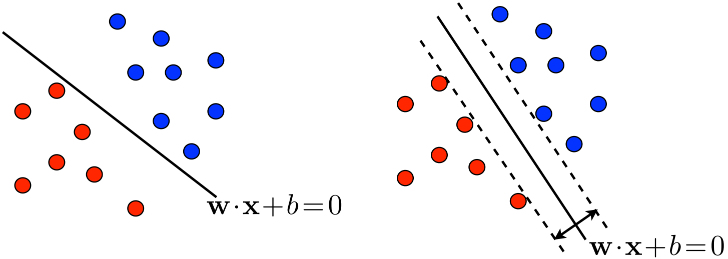
Intuition: Where to put the decision boundary?

- 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?

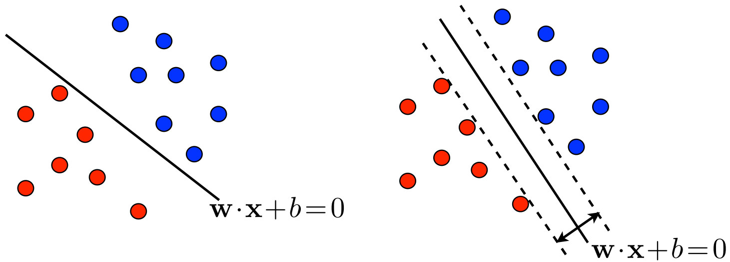
Intuition: Where to put the decision boundary?



Idea: 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

Intuition: Where to put the decision boundary?



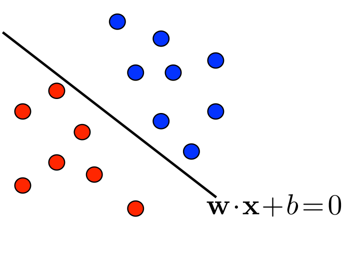
Idea: Find a decision boundary in the '*middle*' of the two classes that:

- 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, we need to review 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?

Vector Norms and Inner Products

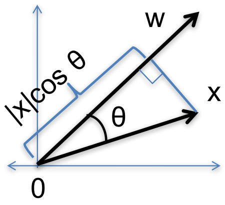
- Given two vectors $\mathbf{w}$ and $\mathbf{x}$, what is their inner product?
- Inner Product $\mathbf{w}^\top \mathbf{x} = w_1 x_1 + w_2 x_2 + \cdots + w_d x_d$

Vector Norms and Inner Products

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

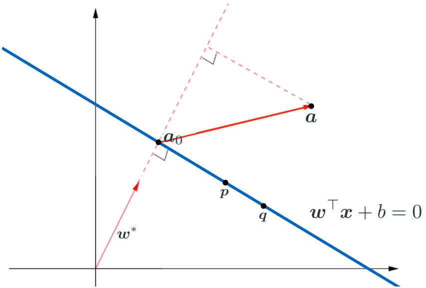
Vector Norms and Inner Products

- Given two vectors **w** and **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 **w** and **x** are perpendicular $\theta = \pi/2$, and thus the inner product is zero

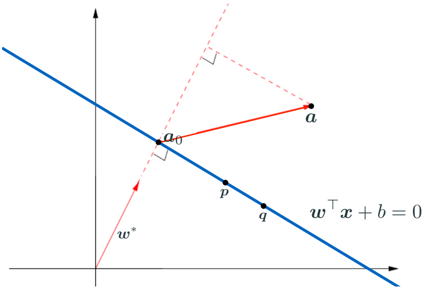
Normal vector of a hyperplane



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

- If $\boldsymbol{p}$ and $\boldsymbol{q}$ are both on the line, then $\boldsymbol{w}^\top \boldsymbol{p} + b = \boldsymbol{w}^\top \boldsymbol{q} + b = 0$.

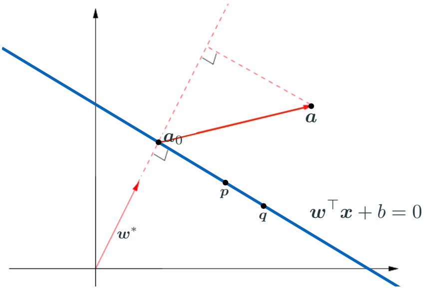
Normal vector of a hyperplane



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 $\boldsymbol{w}^\top(\boldsymbol{p} - \boldsymbol{q}) = \boldsymbol{w}^\top \boldsymbol{p} - \boldsymbol{w}^\top \boldsymbol{q} = -b - (-b) = 0$

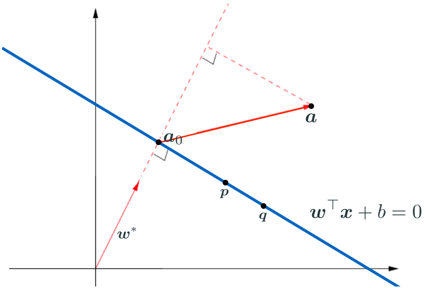
Normal vector of a hyperplane



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

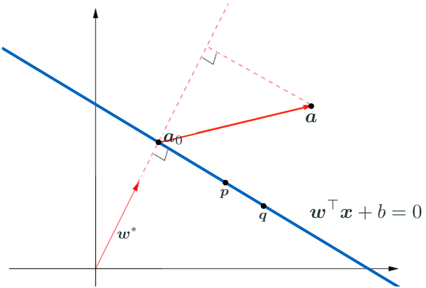
Normal vector of a hyperplane



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
- $\boldsymbol{w}^* = \frac{\boldsymbol{w}}{||\boldsymbol{w}||}$ 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 $\frac{1}{||\boldsymbol{w}||}(\boldsymbol{w}^\top \boldsymbol{a} + b)$

Distance from a point to decision boundary

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

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

Distance from a point to decision boundary

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

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

We can remove the absolute value $|\cdot|$ by exploiting the fact that the decision boundary classifies every point in the training dataset correctly.

Namely, $(\boldsymbol{w}^\top \boldsymbol{x} + b)$ and $\boldsymbol{x}$'s label y must have the same sign, so:

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

Notation change from Logistic Regression

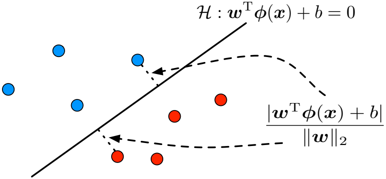
- Change of notation $y = 0 \rightarrow y = -1$
- Separate the bias term b from $\mathbf{w}$

Defining the Margin

Margin

Smallest distance between the hyperplane and all training points

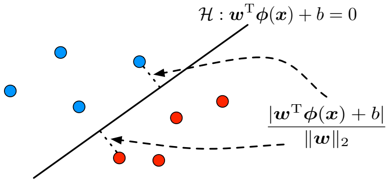
$$\textsc{margin}(\boldsymbol{w}, b) = \min_n \frac{y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b]}{\|\boldsymbol{w}\|_2}$$



Notation change from Logistic Regression

- Change of notation $y = 0 \rightarrow y = -1$
- Separate the bias term b from **w**

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

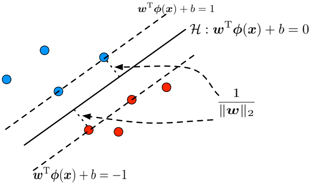
We can further constrain the problem by scaling $(\boldsymbol{w}, b)$ such that

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

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

Assuming separable training data, we thus want to solve:

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

This is equivalent to

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

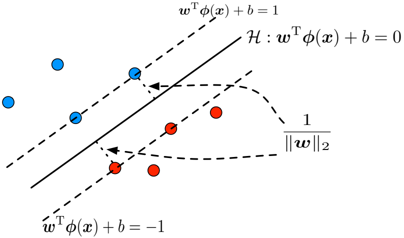
Given our geometric intuition, SVM is called a **max margin** (or large margin) classifier. The constraints are called **large margin constraints**.

Support vectors – a first look

SVM formulation for separable data

$$\min_{\boldsymbol{w},b} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2$$

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



- "$=$": $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] = 1$, these training data are "support vectors"
- "$>$": $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] > 1$, removing them do not affect the optimal solution.

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 will 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

Idea: 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?

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:

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

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, i.e., $C = \frac{1}{\lambda}$

How to solve this problem?

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

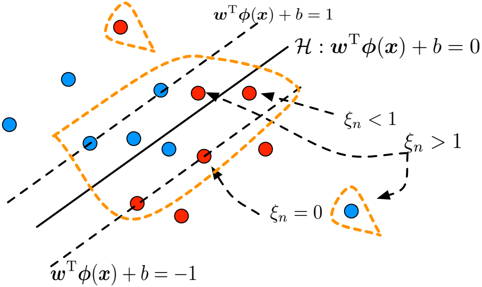
- This is a *convex quadratic program*: the objective function is quadratic in $\boldsymbol{w}$ and linear in $\boldsymbol{\xi}$ and the constraints are linear (inequality) constraints in $\boldsymbol{w}$, b and ξ_n.
- We can solve the optimization problem using general-purpose solvers, e.g., Matlab's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

Meaning of “support vectors” in SVMs

- The SVM solution is only determined by a subset of the training samples (as we will see in more detail in the next lecture)
- These samples are called support vectors
- All other training points do not affect the optimal solution, i.e., if we remove the other points and construct another SVM classifier on the reduced dataset, the optimal solution will be the same

These properties allow us to be more efficient than logistic regression or naive Bayes.

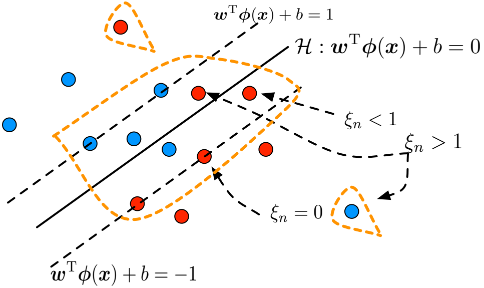
Visualization of how training data points are categorized



Support vectors are highlighted by the dotted orange lines

Recall the constraints $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n$.

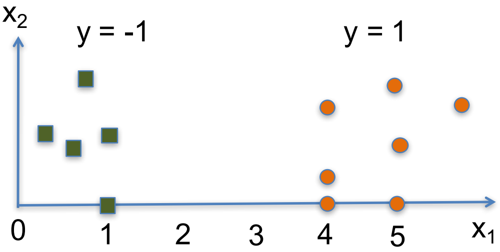
Visualization of how training data points are categorized



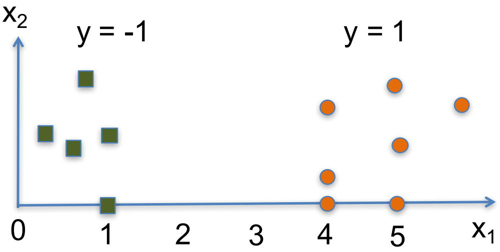
Recall the constraints $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n$. Three types of support vectors

- $\xi_n = 0$: The point is on the boundary
- $0 < \xi_n \leq 1$: On the correct side, but inside the margin
- $\xi_n > 1$: On the wrong side of the boundary

Example of SVM

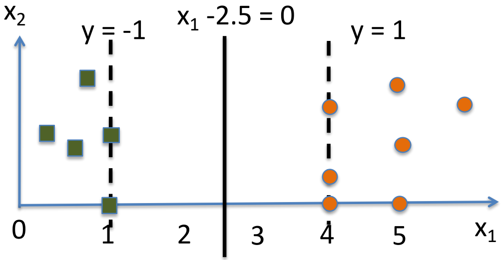


Example of SVM



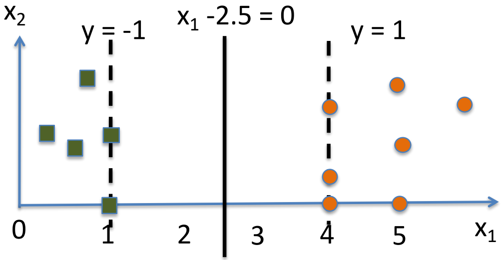
What will be the decision boundary learnt by solving the SVM optimization problem?

Example of SVM



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

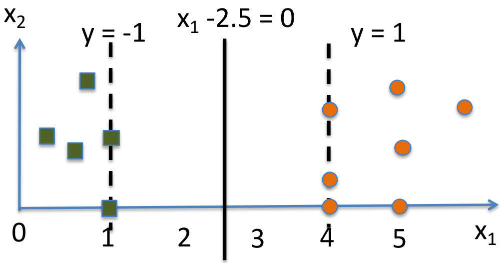
Example of SVM



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 the support vectors to satisfy to $y_n(\mathbf{w}^\top \mathbf{x}_n + b) = 1$.

Example of SVM



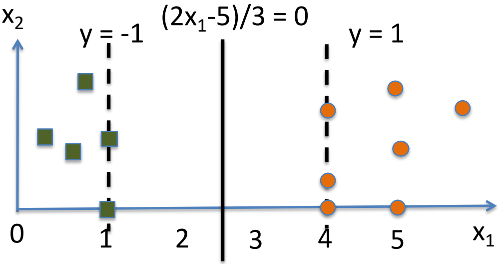
Margin $= 1.5$; the decision boundary has $\mathbf{w} = [1, 0]^\top$, and $b = -2.5$.

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

Not quite. For example, for $\mathbf{x}_n = [1, 0]^\top$, we have

$$y_n(\mathbf{w}^\top \mathbf{x}_n + b) = (-1)[1 - 2.5] = 1.5.$$

Example of SVM: scaling



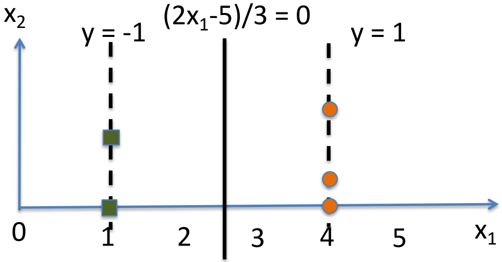
Thus, our optimization problem will re-scale $\mathbf{w}$ and b to get this equation for the same decision boundary

Margin $= 1.5$; the decision boundary has $\mathbf{w} = [2/3, 0]^\top$, and $b = -5/3$.

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

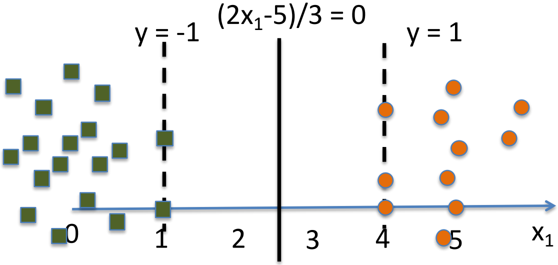
$$y_n(\mathbf{w}^\top \mathbf{x}_n + b) = (-1)[2/3 - 5/3] = 1.$$

Example of SVM: support vectors



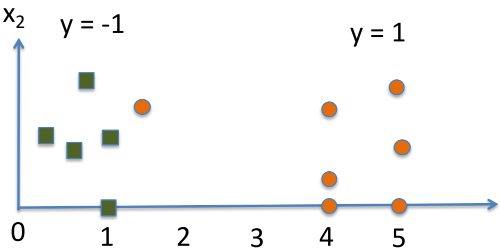
The solution to our optimization problem will be the **same** to the *reduced* dataset containing all the support vectors.

Example of SVM: support vectors



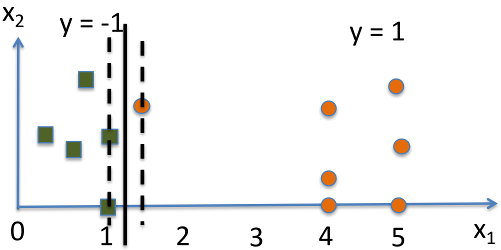
There can be many more data than the number of support vectors (so we can train on a smaller dataset).

Example of SVM: resilience to outliers



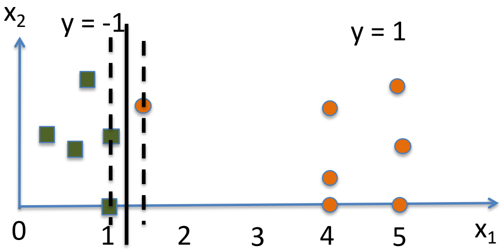
- Still linearly separable, but one of the orange dots is an "outlier".

Example of SVM: resilience to outliers



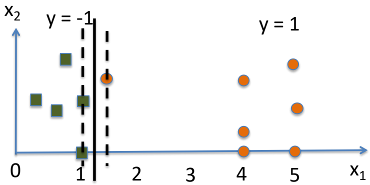
- Naively applying the hard-margin SVM will result in a classifier with small margin.

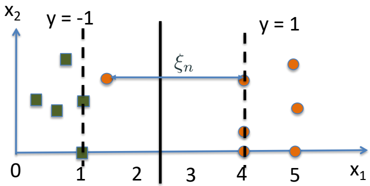
Example of SVM: resilience to outliers



- Naively applying the hard-margin SVM will result in a classifier with small margin.
- So, better to use the soft-margin formulation.

Example of SVM: resilience to outliers

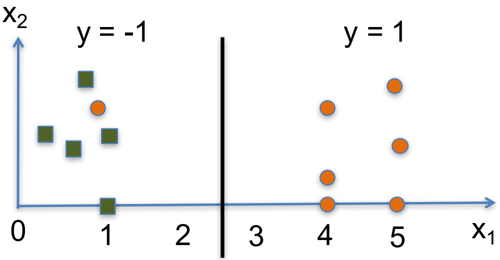




$$
\begin{aligned}
\min_{\boldsymbol{w}, b, \boldsymbol{\xi}} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C \sum_n \xi_n \\
\text{s.t.} \quad & y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n, \quad \forall \quad n \\
& \xi_n \geq 0, \quad \forall \ n
\end{aligned}
$$

- $C = \infty$ corresponds to the hard-margin SVM;
- Due to the flexibility in C, SVM is also less sensitive to outliers.

Example of SVM



- Similar reasons apply to the case when the data is not linearly separable.
- The value of C determines how much the boundary will shift: trade-off of accuracy and robustness (sensitivity to outliers).

Outline

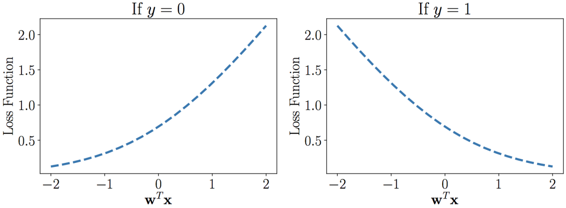
1. Review of Non-linear classification boundary
2. Review of Multi-class Logistic Regression
3. Support Vector Machines (SVM): Intuition
4. SVM: Max Margin Formulation
5. SVM: Hinge Loss Formulation
6. Equivalence of These Two Formulations

SVM: Hinge Loss Formulation

Logistic Regression Loss: Illustration

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

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

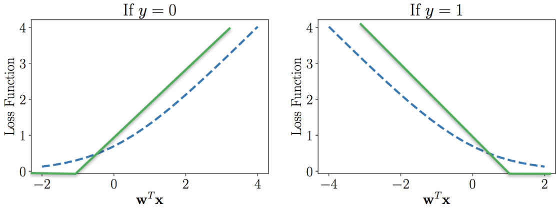


- Loss grows approx. linearly as we move away from the boundary
- Alternative: Hinge Loss Function

Hinge Loss: Illustration

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

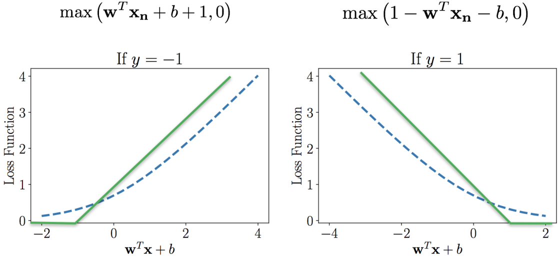
$\max\left(\mathbf{w}^T\mathbf{x_n} + 1, 0\right)$ $\max\left(1 - \mathbf{w}^T\mathbf{x_n}, 0\right)$



- 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: Mathematical Expression

$$\mathcal{L}(\boldsymbol{w}) = -\sum_{n} \max(0, 1 - y_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b))$$



- Change of notation $y = 0 \rightarrow y = -1$
- Separate the bias term b from $\mathbf{w}$
- Makes the mathematical expression more compact

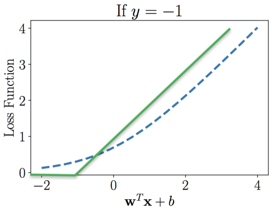
Hinge Loss: Mathematical Expression

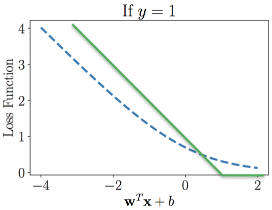
Definition

Assume $y \in \{-1, 1\}$ and the decision rule is $h(\boldsymbol{x}) = \textsc{sign}(\boldsymbol{w}^\top \boldsymbol{x})$ with $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$,

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

$$\max\left(\mathbf{w}^T \mathbf{x_n} + b + 1, 0\right) \qquad \max\left(1 - \mathbf{w}^T \mathbf{x_n} - b, 0\right)$$





Hinge loss

Definition

Assume $y \in \{-1, 1\}$ and the decision rule is $h(\boldsymbol{x}) = \text{SIGN}(f(\boldsymbol{x}))$ with $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$,

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

Intuition

- No penalty if raw output, $f(\boldsymbol{x})$, has same sign and is far enough from decision boundary (i.e., if 'margin' is large enough)
- Otherwise pay a growing penalty, between 0 and 1 if signs match, and greater than one otherwise

Convenient shorthand

$$\ell^{\text{HINGE}}(f(\boldsymbol{x}), y) = \max(0, 1 - yf(\boldsymbol{x})) = (1 - yf(\boldsymbol{x}))_+$$

Optimization Problem of support vector machines (SVM)

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, as we balance between two terms (the loss and the regularizer).

Optimization Problem of support vector machines (SVM)

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, as we balance between two terms (the loss and the regularizer).

- Can solve using gradient descent to get the optimal $\mathbf{w}$ and b

Optimization Problem of support vector machines (SVM)

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, 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 $\boldsymbol{w}^\top \boldsymbol{x}_n + b$

Optimization Problem of support vector machines (SVM)

Minimizing the total hinge loss on all the training data

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

Analogous to regularized least squares, 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

Outline

1. Review of Non-linear classification boundary
2. Review of Multi-class Logistic Regression
3. Support Vector Machines (SVM): Intuition
4. SVM: Max Margin Formulation
5. SVM: Hinge Loss Formulation
6. Equivalence of These Two Formulations

Equivalence of These Two Formulations

Recovering our previous SVM formulation

Rewrite the geometric formulation as the hinge loss formulation:

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

Here's the geometric formulation again:

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

Recovering our previous SVM formulation

Rewrite the geometric formulation as the hinge loss formulation:

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

Here's the geometric formulation again:

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

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

Recovering our previous SVM formulation

Rewrite the geometric formulation as the hinge loss formulation:

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

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

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

Recovering our previous SVM formulation

Rewrite the geometric formulation as the hinge loss formulation:

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

Here's the geometric 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 $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n \iff \xi_n \geq 1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b]$:

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

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

Recovering our previous SVM formulation

Rewrite the geometric formulation as the hinge loss formulation:

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

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

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

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

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

Advantages of SVM

We've seen that the geometric formulation of SVM is equivalent to minimizing the empirical hinge loss. This explains why SVM:

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

We will need to use duality to show the next three properties.