

# **18-661 Introduction to Machine Learning**

## Support Vector Machines (SVM) – I

---

Fall 2020

ECE – Carnegie Mellon University

# Announcements and Reminders

- Midterm exam will be during class on Tuesday, Oct 20th.
  - Going over homework and recitation problems is a great way to study.
  - All topics through 10/15 will be covered (MLE/MAP, linear regression, bias-variance tradeoff, naive Bayes, logistic regression, multi-class, SVMs, Graphical Models).
  - Practice problems will be released next week.
- HW 3 is due on Thursday, Oct 8th
- Recitation on Friday will cover duality, an optimization concept that is useful for formulating SVMs (as we will see next week). However, you will not be tested on duality.

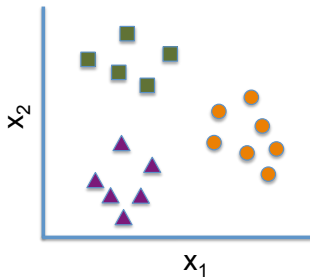
1. Review of Multi-class Logistic Regression
2. Support Vector Machines (SVM): Intuition
3. SVM: Max-Margin Formulation
4. SVM: Hinge Loss Formulation

# Review of Multi-class Logistic Regression

---

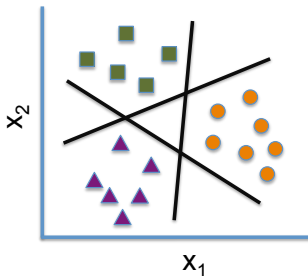
# Three approaches

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



# The One-versus-Rest or One-versus-All approach

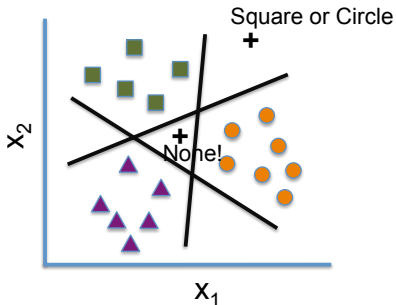
- For each class  $c$ , change the problem into binary classification
  1. Relabel training data with label  $c$ , into POSITIVE (or '1').
  2. Relabel all the rest data into NEGATIVE (or '0').
- Repeat this multiple times: Train  $C$  binary classifiers, using logistic regression to differentiate the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

- There is ambiguity in some of the regions (the 4 triangular areas).
- How do we resolve this?

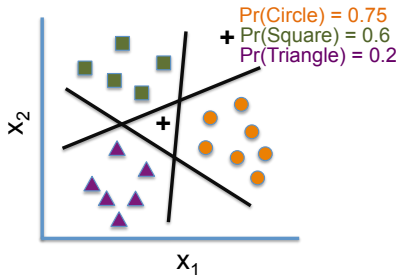


# The One-versus-Rest or One-versus-All approach

How to combine these linear decision boundaries?

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

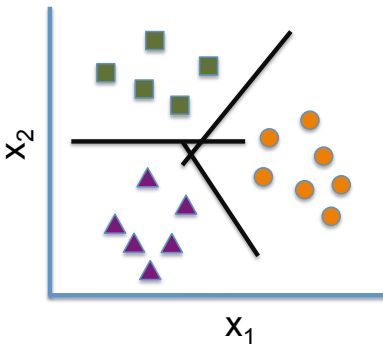
$$c^* = \arg \max_{c=1,\dots,C} \Pr(y = c|\mathbf{x}) = \sigma(\mathbf{w}_c^\top \mathbf{x})$$





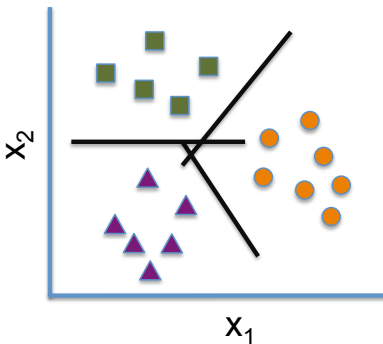
# The One-versus-One approach

- 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  $\mathbf{x}$ , 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  $\mathbf{w}_c$  and model the posterior probability as:

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

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

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

# Parameter estimation for multinomial logistic regression

**Discriminative approach:** Maximize conditional likelihood

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

We will change  $y_n$  to  $\mathbf{y}_n = [y_{n1} \ y_{n2} \ \cdots \ y_{nC}]^\top$ , 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,  $\mathbf{y}_n = [0 \ \mathbf{1} \ 0 \ 0 \ \cdots \ 0]^\top$ .

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

# Cross-entropy error function

**Definition:** negative log-likelihood

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

## 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.

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

# Support Vector Machines (SVM): Intuition

---

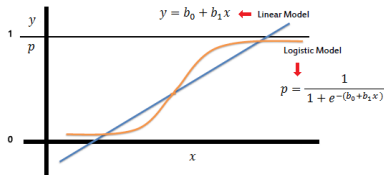
# Why do we need SVM?

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  $p(\mathbf{x}) > 0.5$  or  $< 0.5$ .
- We **don't** (always) need to know how far  $\mathbf{x}$  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:

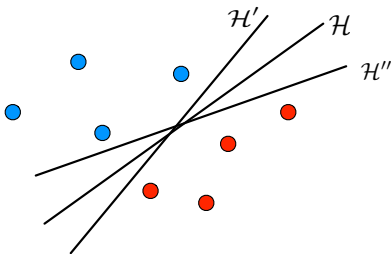
1. Is less sensitive to outliers.
2. Maximizes distance of training points from the boundary
3. Scales better with high-dimensional data.
4. Only requires a subset of the training points.
5. Generalizes well to many nonlinear models.

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

## **SVM: Max-Margin Formulation**

---

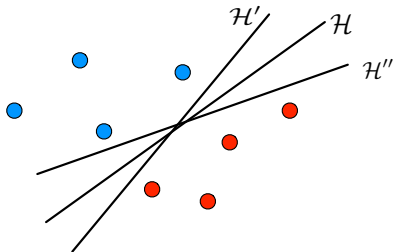
## Binary classification: Finding a linear decision boundary



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

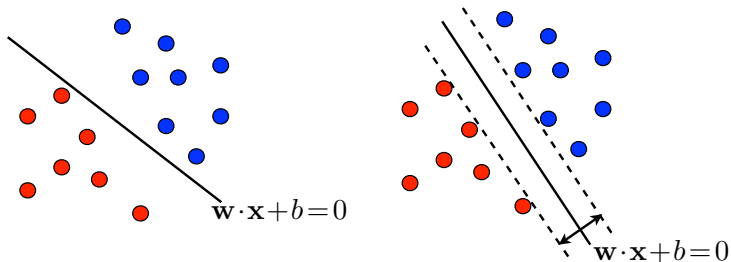
# Intuition: Where to put the decision boundary?

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



- Which one should we pick?

## Intuition: Where to put the decision boundary?



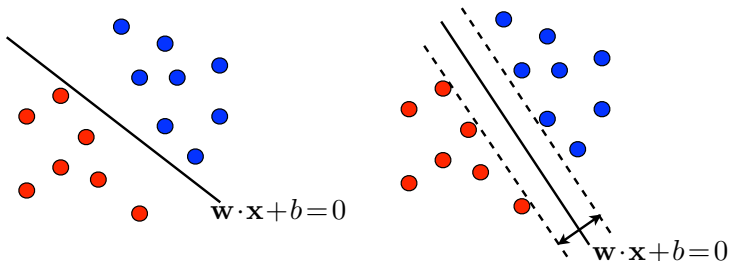
Find a decision boundary in the '*middle*' of the two classes that:

- 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, we need to review some vector geometry

What is a hyperplane?

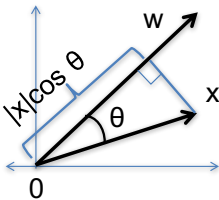


- General equation is  $\mathbf{w}^\top \mathbf{x} + b = 0$
- Divides the space in half, i.e.,  $\mathbf{w}^\top \mathbf{x} + b > 0$  and  $\mathbf{w}^\top \mathbf{x} + b < 0$
- A hyperplane is a line in 2D and a plane in 3D
- $\mathbf{w} \in \mathbb{R}^d$  is a non-zero normal vector



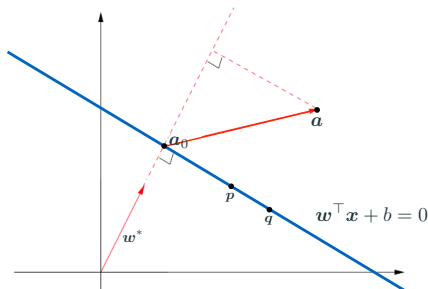
# Vector norms and inner products

- Given two vectors  $\mathbf{w}$  and  $\mathbf{x}$ , what is their inner product?
- Inner Product  $\mathbf{w}^\top \mathbf{x} = w_1x_1 + w_2x_2 + \cdots + w_dx_d$



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

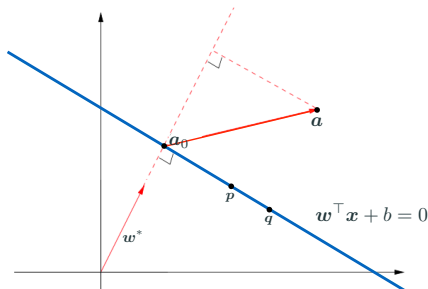
# Normal vector of a hyperplane



**Vector  $w$  is normal to the hyperplane. Why?**

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

# Distance from a hyperplane



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

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

# Distance from a point to decision boundary

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

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

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

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

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

## Notation changes from Logistic Regression

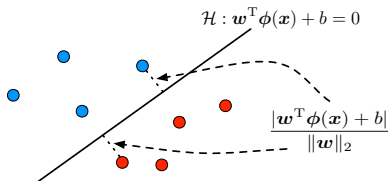
- Change of notation  $y = 0 \rightarrow y = -1$
- Separate the bias term  $b$  from  $\mathbf{w}$

# Defining the margin

## Margin

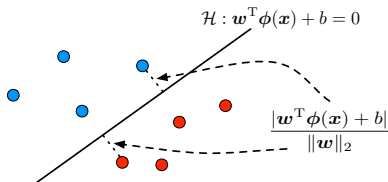
Smallest distance between the hyperplane and all training points

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



How can we use this to find the SVM solution?

# Optimizing the margin



## How should we pick $(\mathbf{w}, b)$ 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!

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

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

## Margin

Smallest distance between the hyperplane and all training points

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

Consider three hyperplanes

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

Which one has the largest margin?

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

# Rescaled margin

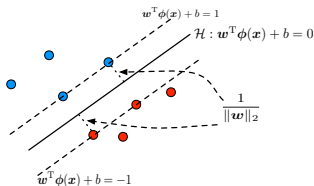
We can further constrain the problem by scaling  $(\mathbf{w}, b)$  such that

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

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

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

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





# SVM: Max margin formulation for separable data

Assuming separable training data, we thus want to solve:

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

This is equivalent to

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

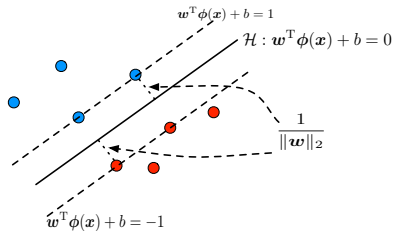
Given our geometric intuition, SVM is called a **max margin** (or large margin) classifier. The constraints are called **large margin constraints**.

# Support vectors – a first look

## SVM formulation for separable data

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

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



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

- “=”:  $y_n[\mathbf{w}^\top \mathbf{x}_n + b] = 1$ . These training data points are called “support vectors”.
- “>”:  $y_n[\mathbf{w}^\top \mathbf{x}_n + b] > 1$ . Removing these data points does not affect the optimal solution.

# SVM for non-separable data

## SVM formulation for separable data

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

## 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  $\mathbf{w}$  there exists a training point  $\mathbf{x}_i$  such that

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

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

# SVM for non-separable data

## Constraints in separable setting

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

## Constraints in non-separable setting

Can we modify our constraints to account for non-separability?

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

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

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

# Soft-margin SVM formulation

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

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

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?

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

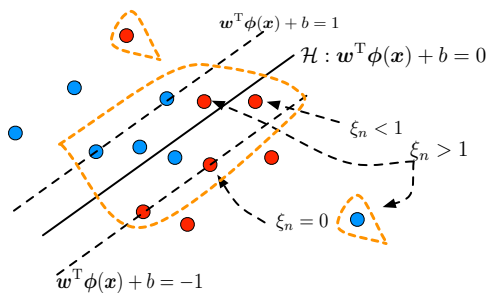
- This is a **convex quadratic program**: the objective function is quadratic in  $\mathbf{w}$  and linear in  $\xi$  and the constraints are linear (inequality) constraints in  $\mathbf{w}$ ,  $b$  and  $\xi_n$ .
- We can solve the optimization problem using general-purpose solvers, e.g., Matlab's `quadprog()`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

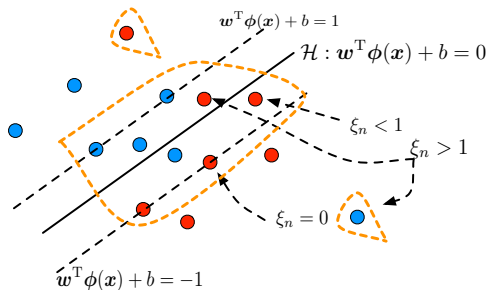
# Visualization of how training data points are categorized



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



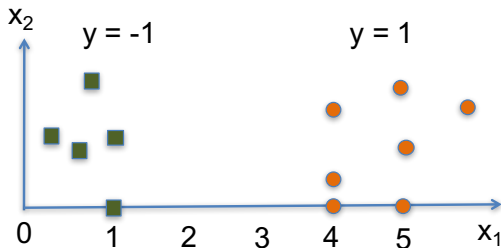
# Visualization of how training data points are categorized



Recall the constraints  $y_n[\mathbf{w}^T \mathbf{x}_n + b] \geq 1 - \xi_n$ . These define three types of support vectors:

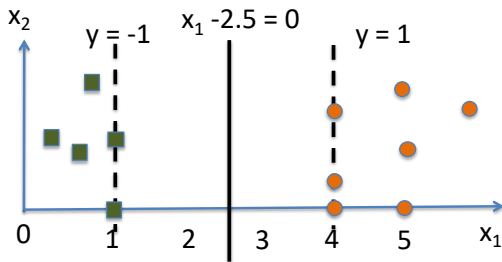
- $\xi_n = 0$ : The point is on the boundary.
- $0 < \xi_n \leq 1$ : On the correct side, but inside the margin.
- $\xi_n > 1$ : On the wrong side of the boundary.

## Example of SVM



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

## Example of SVM



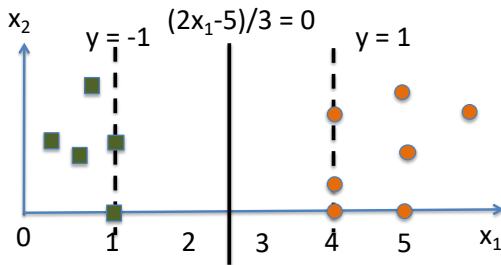
Margin = 1.5; the decision boundary has  $\mathbf{w} = [1, 0]^\top$ , and  $b = -2.5$ .

Is this the right scaling of  $\mathbf{w}$  and  $b$ ? We need 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



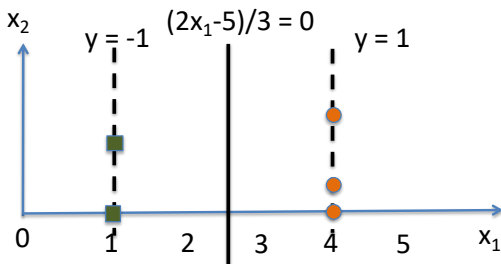
Thus, our optimization problem will re-scale  $\mathbf{w}$  and  $b$  to get this equation for the same decision boundary

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

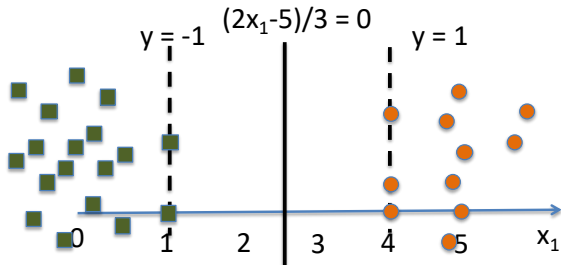
$$y_n(\mathbf{w}^\top \mathbf{x}_n + b) = (-1)[2/3 - 5/3] = 1.$$

## Example of SVM: Support vectors



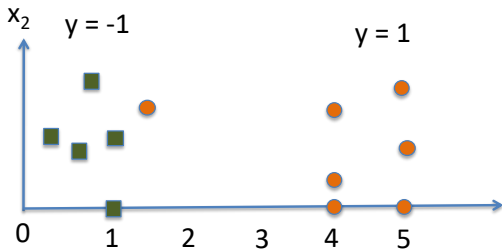
The solution to our optimization problem will be the **same** to the *reduced* dataset containing all the support vectors.

## Example of SVM: Support vectors



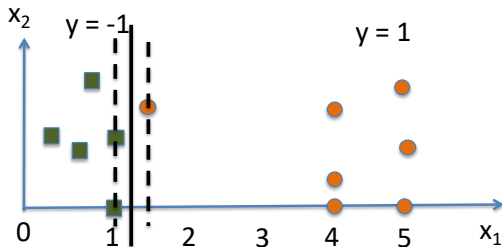
There can be many more data than the number of support vectors (so we can train on a smaller dataset).

## Example of SVM: Resilience to outliers



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

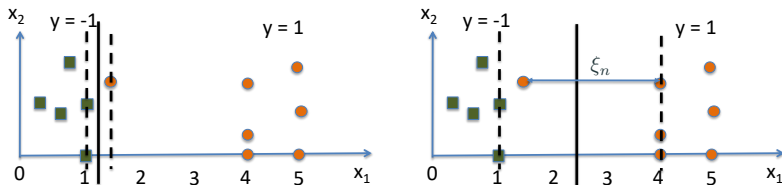
## Example of SVM: Resilience to outliers



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



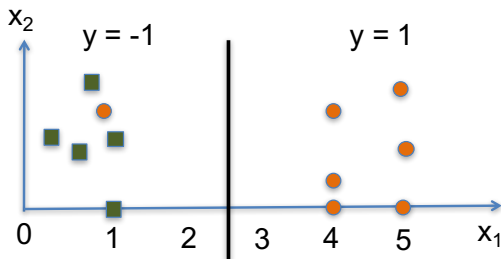
## Example of SVM: Resilience to outliers



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

- $C = \infty$  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).

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

## **SVM: Hinge Loss Formulation**

---

# SVM vs. Logistic regression

## SVM soft-margin formulation

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

## Logistic regression formulation

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

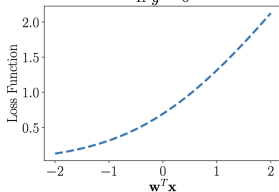
- Logistic regression defines a **loss for each data point** and minimizes the total loss plus a regularization term.
- This is convenient for assessing the “goodness” of the model on each data point.
- Can we write SVMs in this form as well?

# Logistic regression loss: Illustration

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

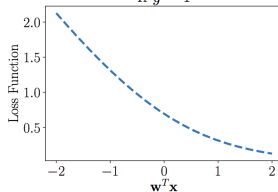
$$-\log \left( \frac{e^{-\mathbf{w}^\top \mathbf{x}_n}}{1 + e^{-\mathbf{w}^\top \mathbf{x}_n}} \right)$$

If  $y = 0$



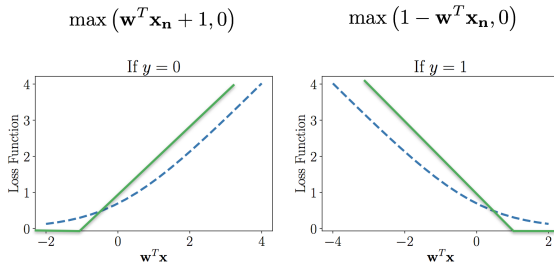
$$-\log \left( \frac{1}{1 + e^{-\mathbf{w}^\top \mathbf{x}_n}} \right)$$

If  $y = 1$



- Alternative: Hinge Loss Function

# Hinge loss: Illustration

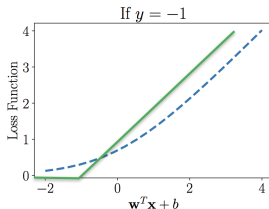


- Loss grows linearly as we move away from the boundary.
- No penalty if a point is more than 1 unit from the boundary.
- Makes the search for the boundary easier (as we will see later).

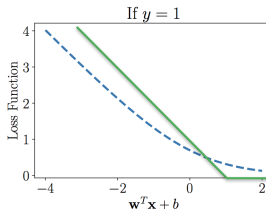
# Hinge loss: Mathematical expression

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

$$\max(\mathbf{w}^\top \mathbf{x}_n + b + 1, 0)$$



$$\max(1 - \mathbf{w}^\top \mathbf{x}_n - b, 0)$$



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

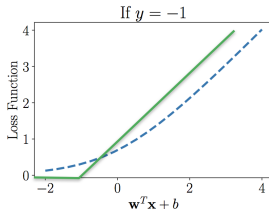


# Hinge loss: Mathematical expression

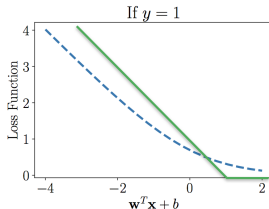
Assume  $y \in \{-1, 1\}$  and the decision rule is  $h(\mathbf{x}) = \text{SIGN}(\mathbf{w}^\top \mathbf{x} + b)$  with  $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$ ,

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

$$\max(\mathbf{w}^T \mathbf{x}_n + b + 1, 0)$$



$$\max(1 - \mathbf{w}^T \mathbf{x}_n - b, 0)$$



# Hinge loss

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

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

## Intuition

- No penalty if raw output,  $f(\mathbf{x})$ , 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

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

# Hinge loss SVM formulation

Minimizing the total hinge loss on all the training data

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

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  $\mathbf{w}^\top \mathbf{x}_n + b$ .
- Much easier to compute than in logistic regression, where we need to compute the sigmoid function  $\sigma(\mathbf{w}^\top \mathbf{x}_n + b)$  in each iteration.

# Recovering our previous SVM formulation

Rewrite the geometric formulation as the hinge loss formulation:

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

Here's the geometric formulation again:

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

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

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

Now since the  $\xi_n$  should always be as small as possible, we obtain:

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

# 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.

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.