DSC 140A

Probabilistic Modeling & Machine Learning

Lecture 6 | Part 1

Maximum Margin Classifiers

Recall: Perceptrons

- Linear classifier fit using loss function:

$$L_{\text{tron}}(H(\vec{x}), y) = \begin{cases} 0, & \text{sign}(H(\vec{x})) = \text{sign}(y) \\ |H(\vec{x})|, & \text{sign}(H(\vec{x})) \neq \text{sign}(y) \end{cases}$$

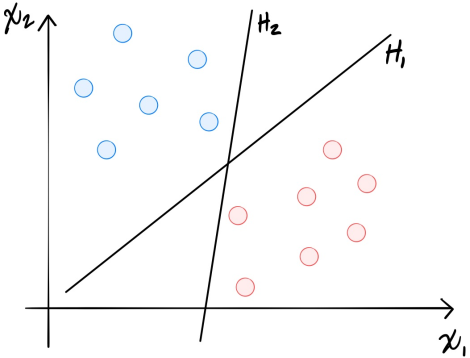
A Problem with the Perceptron

- **Recall:** the perceptron loss assigns no penalty to points that are correctly classified.

- No matter how close the point is to the boundary.

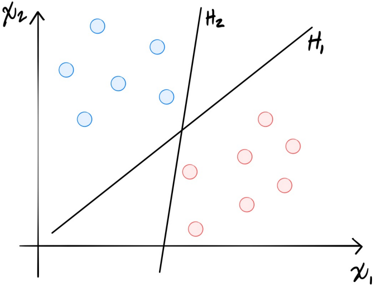
Exercise

What is the empirical risk with respect to the perceptron loss of H_1? What about H_2?



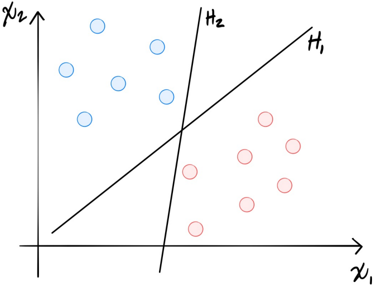
Linear Separability

- Data are **linearly separable** if there exists a linear classifier which perfectly classifies the data.



Margin

- The **margin** is the smallest distance between the decision boundary and a training point.



Maximum Margin Classifier

- If training data are linearly separable, there are many classifiers with zero error.
- We prefer classifiers with larger margins.
  - Better generalization performance.
- Can we find the **maximum margin classifier**?
  - I.e., the classifier with the largest possible margin?

Observation

- A point is classified correctly when:

$$\begin{cases} \vec{w} \cdot \operatorname{Aug}(\vec{x}^{(i)}) > 0, & \text{if } y_i = 1 \\ \vec{w} \cdot \operatorname{Aug}(\vec{x}^{(i)}) < 0, & \text{if } y_i = -1 \end{cases}$$

- Equivalently, classification is correct if:

$$y_i \, \vec{w} \cdot \operatorname{Aug}(\vec{x}^{(i)}) > 0$$

Attempt #1

- **Goal:** Assume linear separability. Find a $\vec{w}$ so that $y_i\, \vec{w} \cdot \text{Aug}(\vec{x}^{(i)}) > 0$ for all data points.
- That is, all points are correctly classified.
- Too easy!
  - Perceptron already does this.
  - Does not force margin to be maximized.

Enforce a Margin

- **Recall:** $|H(\vec{x})| = |\vec{w} \cdot \text{Aug}(\vec{x})|$ is **proportional** to distance from decision boundary.
  - Doesn't measure actual distance!
  - Scaled by a factor depending on $1/\|\vec{w}\|$.

- **Informal:** $|H(\vec{x})|$ measures distance in "prediction units"
  - E.g., if $H(\vec{x}) = -2$, $\vec{x}$ is 2 "prediction units" away from boundary

Enforce a Margin

- We can enforce a margin in “prediction units”.
- E.g., to require a margin of one prediction unit, we must have

$$y_i H(\vec{x}^{(i)}) = y_i \vec{w} \cdot \operatorname{Aug} \vec{x}^{(i)} \geq 1$$

for each data point

Attempt #2

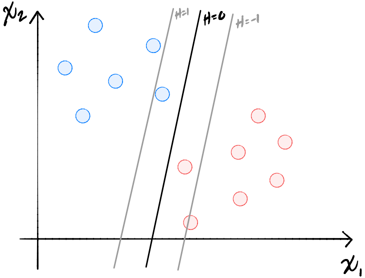
- **Goal:** Assume linear separability. Find a $\vec{w}$ so that $y_i \, \vec{w} \cdot \text{Aug}(\vec{x}^{(i)}) \geq 1$ for all data points.

- Still “too easy”.
  - Problem: prediction units aren’t actual distance.
  - We can artificially increase distance in “prediction units” by increasing $\|\vec{w}\|$.

Exercise

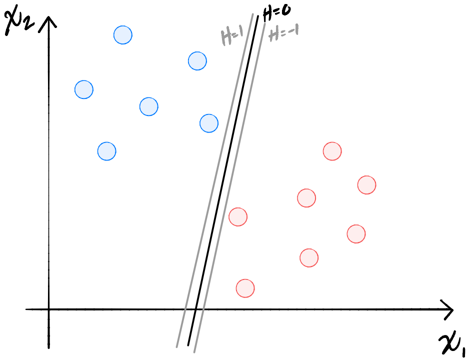
Suppose H is a linear predictor with parameter vector $\vec{w}$. Shown are the lines one "prediction unit" away from the decision boundary.

How will the decision boundary and these lines change if $\vec{w}$ is doubled?



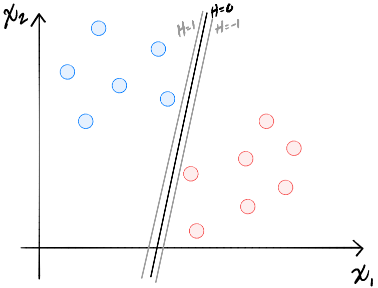
Solution

- The decision boundary remains unchanged.
- The lines one “prediction unit” away move closer.



Observe

- H satisfies $y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)}) \geq 1$



Observe

- Any vector $\vec{w}$ satisfying $y_i \, \vec{w} \cdot \text{Aug}(\vec{x}^{(i)}) > 0$ can be made to satisfy $y_i \, \vec{w} \cdot \text{Aug}(\vec{x}^{(i)}) \geq 1$ by increasing $\|\vec{w}\|$ appropriately.
- But this is **cheating!**
- Fix: search for a low-norm $\vec{w}$ satisfying $y_i \, \vec{w} \cdot \text{Aug}(\vec{x}^{(i)}) \geq 1$

Attempt #3

- **Goal:** out of all $\vec{w}$ satisfying $y_i\, \vec{w} \cdot \operatorname{Aug}(\vec{x}^{(i)}) \geq 1$ for all data points, find that with minimum $\|\vec{w}\|$

- That is, find:

$$\vec{w}^* = \arg\min_{\vec{w}} \|\vec{w}\|$$

subject to: $\forall i,\ y_i \vec{w} \cdot \operatorname{Aug}(\vec{x}) \geq 1$

Hard-SVM

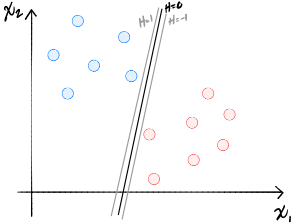
- This optimization problem is called the **Hard Support Vector Machine** classifier problem.
- Only makes sense if data are linearly separable.
- In a moment, we'll see the Soft-SVM.

How?

- Turn it into a **convex quadratic** optimization problem:
  - Minimize $\|\vec{w}\|^2$ subject to $y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)}) \geq 1$ for all i.
- Can be solved efficiently with **quadratic programming**.
  - But there is no exact general formula for the solution

Exercise

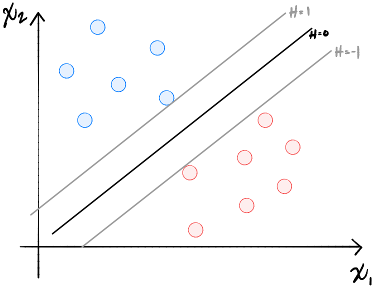
Can the below predictor be a solution of the Hard-SVM?



SVMs are Maximum Margin Classifiers

- Intuition says solutions of Hard-SVM will have large margins.

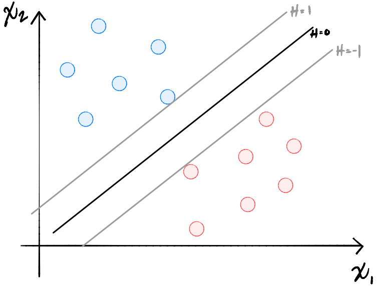
- Fact: they maximize the margin.



Support Vectors

- A **support vector** is a training point $\vec{x}^{(i)}$ such that

$$y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)}) = 1$$



Support Vectors

- **Fact:** the solution to Hard-SVM is always a linear combination of the support vectors.

- That is, let S be the set of support vectors. Then

$$\vec{w}^* = \sum_{i \in S} y_i \alpha_i \operatorname{Aug}(\vec{x}^{(i)})$$

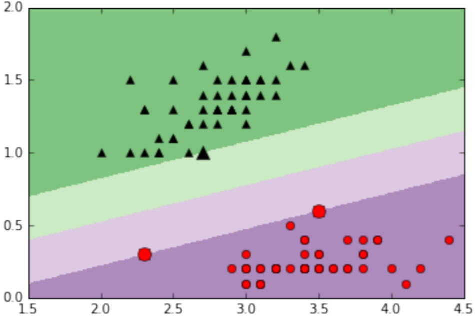
Example: Irises



- 3 classes: *iris setosa*, *iris versicolor*, *iris virginica*
- 4 measurements: petal width/height, sepal width/height

Example: Irises

- Using only sepal width/petal width
- Two classes: versicolor (black), setosa (red)



DSC 140A

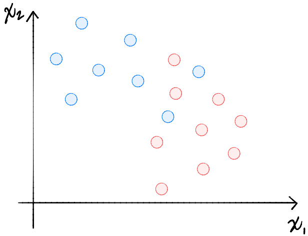
Probabilistic Modeling & Machine Learning

Lecture 6 | Part 2

Soft-Margin SVMs

Non-Separability

- So far we've assumed data is linearly separable.
- What if it isn't?

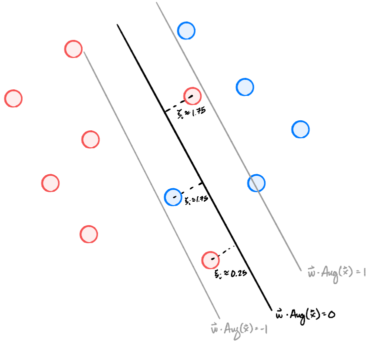


The Problem

- **Old Goal**: Minimize $\|\vec{w}\|^2$ subject to $y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)}) \geq 1$ for all i.
- This **no longer makes sense**.

Cut Some Slack

- **Idea**: allow some classifications to be ξ_i wrong, but not too wrong.



Cut Some Slack

- **New problem.** Fix some number $C \geq 0$.

$$\min_{\vec{w} \in \mathbb{R}^{d+1}, \vec{\xi} \in \mathbb{R}^n} \|\vec{w}\|^2 + C \sum_{i=1}^{n} \xi_i$$

subject to $y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)}) \geq 1 - \xi_i$ for all i, $\vec{\xi} \geq 0$.

The Slack Parameter, C

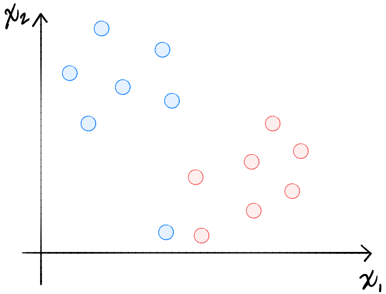
- C controls how much slack is given.

$$\min_{\vec{w}\in\mathbb{R}^{d+1},\vec{\xi}\in\mathbb{R}^n} \|\vec{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

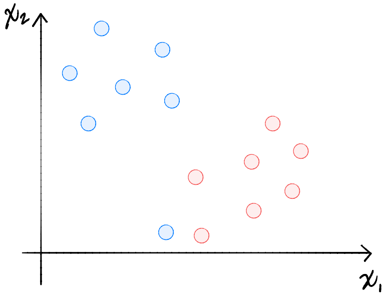
subject to $y_i\vec{w}\cdot\text{Aug}(\vec{x}^{(i)}) \geq 1 - \xi_i$ for all i, $\vec{\xi} \geq 0$.

  - Large C: don't give much slack. Avoid misclassifications.
  - Small C: allow more slack at the cost of misclassifications.

Example: Small C



Example: Large C



Soft and Hard Margins

- Max-margin SVM from before has **hard margin**.
- Now: the **soft margin** SVM.
- As $C \to \infty$, the margin hardens.

DSC 140A

Probabilistic Modeling & Machine Learning

Lecture 6 | Part 3

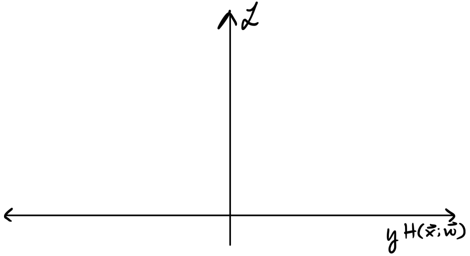
Hinge Loss

Loss Functions?

- So far, we've learned predictors by minimizing **expected loss** via ERM.
- But this isn't what we did with Hard-SVM and Soft-SVM.
- It turns out, we can frame Soft-SVM as an ERM problem.

Recall: Perceptron Loss

$$L_{\text{tron}}(H(\vec{x}), y) = \begin{cases} 0, & \text{sign}(H(\vec{x})) = \text{sign}(y) \\ |H(\vec{x})|, & \text{sign}(H(\vec{x})) \neq \text{sign}(y) \end{cases}$$

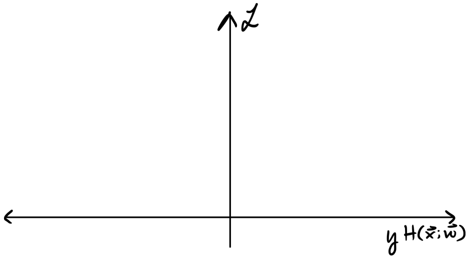


Perceptron Loss

- Perceptron loss did not penalize correct classifications.
- Even if they were very close to boundary.
- **Idea:** penalize predictions that are close to the boundary, too.

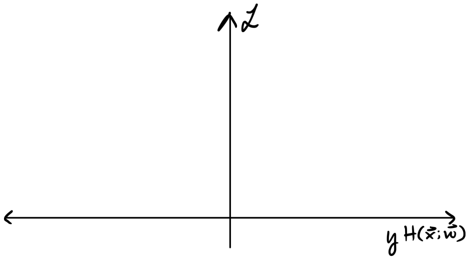
The Hinge Loss

$$L_{\text{hinge}}(H(\vec{x}), y) = \begin{cases} 0, & yH(\vec{x}) \geq 1, \\ 1 - yH(\vec{x}), & yH(\vec{x}) < 1 \end{cases}$$



The Hinge Loss

$$L_{\text{hinge}}(H(\vec{x}), y) = \max\{0, 1 - yH(\vec{x})\}$$



Equivalence

- Recall the Soft-SVM problem:

$$\min_{\vec{w} \in \mathbb{R}^{d+1}, \vec{\xi} \in \mathbb{R}^n} \|\vec{w}\|^2 + C \sum_{i=1}^{n} \xi_i$$

subject to $y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)}) \geq 1 - \xi_i$ for all i, $\vec{\xi} \geq 0$.

- **Note**: if $\vec{x}^{(i)}$ is misclassified, then

$$\xi_i = 1 - y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)})$$

Equivalence

- The Soft-SVM problem is equivalent to finding $\vec{w}$ that minimizes:

$$R_{\text{svm}}(\vec{w}) = \|\vec{w}\|^2 + C \sum_{i=1}^{n} \max\{0, 1 - y_i \vec{w} \cdot \vec{x}^{(i)}\}$$

- R_{svm} is the **regularized** risk.

- C is a parameter affecting "softness" of boundary; chosen by you.

Another Way to Optimize

- In practice, SGD is often used to train soft SVMs.

DSC 140A

Probabilistic Modeling & Machine Learning

Lecture 6 | Part 4

Demo: Sentiment Analysis

Why use linear predictors?

- Linear classifiers look to be very simple.
- That can be both **good** and **bad**.
  - **Good**: the math is tractable, less likely to overfit
  - **Bad**: may be too simple, underfit
- They can work surprisingly well.

Sentiment Analysis

- **Given**: a piece of text.
- **Determine**: if it is **postive** or **negative** in tone
- Example: “Needless to say, I wasted my money.”

The Data

- Sentences from reviews on Amazon, Yelp, IMDB.
- Each labeled (by a human) **positive** or **negative**.
- Examples:
  - **“Needless to say, I wasted my money.”**
  - **“I have to jiggle the plug to get it to line up right.”**
  - **“Will order from them again!”**
  - **“He was very impressed when going from the original battery to the extended battery.”**

The Plan

- We'll train a soft-margin SVM.
- Problem: SVMs take **fixed-length vectors** as inputs, not sentences.

Bags of Words

To turn a document into a fixed-length vector:

- First, choose a **dictionary** of words:
  - E.g.: `["wasted", "impressed", "great", "bad", "again"]`
- Count number of occurrences of each dictionary word in document.
  - "It was bad. So bad that I was impressed at how bad it was." $\to (0, 1, 0, 3, 0)^T$
- This is called a **bag of words** representation.

Choosing the Dictionary

- Many ways of choosing the dictionary.
- Easiest: take all of the words in the training set.
  - Perhaps throw out **stop words** like “the”, “a”, etc.
- Resulting dimensionality of feature vectors: large.

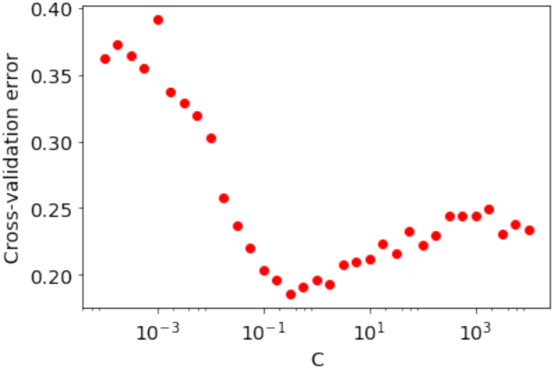
Experiment

- Bag of words features with 4500 word dictionary.
- 2500 training sentences, 500 test sentences.
- Train a soft margin SVM.

Choosing C

- We have to choose the slack parameter, C.
- Use **cross validation**!

Cross Validation



Results

- With $C = 0.32$, test error $\approx 15.6\%$.

C	training error (%)	test error (%)	# support vectors
0.01	23.72	28.4	2294
0.1	7.88	18.4	1766
1	1.12	16.8	1306
10	0.16	19.4	1105
100	0.08	19.4	1035
1000	0.08	19.4	950