CS221 Section 1

Foundations

Roadmap

Multiclass Classification

Matrix Calculus

Python

Recurrence Relation

Probability Theory

Binary Classification

Let's review binary classification

$$x \longrightarrow \boxed{f_{\mathbf{w}}} \longrightarrow y \in \{-1, +1\}$$

Score:

$$\text{score}_{+1}(x, \mathbf{w}) = \mathbf{w} \cdot \phi(x)$$
$$\text{score}_{-1}(x, \mathbf{w}) = (-\mathbf{w}) \cdot \phi(x)$$

Prediction:

$$f_{\mathbf{w}}(x) = \begin{cases} +1 & \text{if score}_{+1}(x, \mathbf{w}) > \text{score}_{-1}(x, \mathbf{w}) \\ -1 & \text{otherwise} \end{cases}$$

$$f_{\mathbf{w}}(x) = \arg \max_{y \in \{-1,+1\}} \text{score}_y(x, \mathbf{w})$$

- The function f uses an underlying score $w \cdot \phi(x)$, which the predictor thresholds at 0 to determine which class is chosen.
- Geometric intuition: decision boundary defined by score $= w \cdot \phi(x) = 0$. The decision boundary is orthogonal to the weight vector and points towards the "positive" side of the decision boundary.
- We can also generate a score for each of the two classes. The positive class score is a measure of how confident we are that an input should be labeled as positive, and vice versa. We then assign the label that gives us the highest score

Multiclass Classification

Problem

Suppose we have 3 possible labels $y \in \{\text{R}, \text{G}, \text{B}\}$

Weight vectors: $\mathbf{w} = \{\mathbf{w}_{\text{R}}, \mathbf{w}_{\text{G}}, \mathbf{w}_{\text{B}}\}$

Scores: $[\mathbf{w}_{\text{R}} \cdot \phi(x)], [\mathbf{w}_{\text{G}} \cdot \phi(x)], [\mathbf{w}_{\text{B}} \cdot \phi(x)]$

Prediction: $\hat{y} = f_{\mathbf{w}}(x) = \arg \max_{y \in \{\text{R},\text{G},\text{B}\}} [\mathbf{w}_y \cdot \phi(x)]$

- With multiple classes, we define multiple scores, one for each class. Each score is produced using a different weight vector Again, we predict the label that gives produces the highest score.
- Geometric intuition: Each score gives us a decision boundary that separates that class from all the other classes.

Loss Functions

How to learn $\mathbf{w}$?

How about **0-1 loss**:

$$\text{Loss}_{\text{0-1}}(x, y, \mathbf{w}) = \begin{cases} 1 & \text{if } \hat{y} \neq y \\ 0 & \text{otherwise} \end{cases}$$

Problem: Gradient is 0 almost everywhere

- **Loss** is a measure of how bad our predictions are. We want to find the w that minimizes loss.
- Because the gradient of 0-1 loss is 0 almost everywhere, stochastic *gradient* descent does not know in which direction it should take a step to reach the minimum of the loss function.

Hinge Loss

How to learn $\mathbf{w}$?

Recall **hinge loss**:

$$\text{margin} = \text{score}_y(x, \mathbf{w}) - \max_{y' \neq y} \text{score}_{y'}(x, \mathbf{w})$$

$$\text{Loss}_{\text{Hinge}}(x, y, \mathbf{w}) = \max\{1 - \text{margin}, 0\}$$

What is the gradient?

- The main difference for hinge loss in the multiclass vs binary case is the definition of the margin.
- **Margin** in the multiclass case is the difference between the score of the correct class and the maximal score out of all of the incorrect classes. We want the margin to be greater than 1.
- If our classifier works, the margin should be positive (class with the max score is the correct class), else negative.
- Gradient: 0 if margin ≥ 1, else $\nabla_{w_y} L = -\phi(x)$ and $\nabla_{w_{y''}} L = \phi(x)$ where $y'' = \arg\max_{y' \neq y} \text{score}_{y'}(x, \mathbf{w})$
- We need to optimize multiple weight vectors, so we can take the gradient of loss with respect to each of these vectors separately. In one iteration of SGD, we will update both w_y and $w_{y''}$
- Intuitively, SGD boosts the correct score and suppresses the incorrect scores.

Roadmap

Multiclass Classification

Matrix Calculus

Python

Recurrence Relation

Probability Theory

Useful Properties

$$\text{"}\mathbf{v}^2\text{"} = \|\mathbf{v}\|_2^2 = \mathbf{v} \cdot \mathbf{v} = \mathbf{v}^T \mathbf{v}$$

$$(\mathbf{A} + \mathbf{B})^T = \mathbf{A}^T + \mathbf{B}^T$$

$$(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T$$

Matrix Calculus

$$f(\mathbf{w}) = (\mathbf{a} \cdot \mathbf{w} + 1)^2 + b \|\mathbf{w}\|_2^2 + \mathbf{w}^\top C \mathbf{w}$$

Compute $\nabla_{\mathbf{w}} f(\mathbf{w})$

$$\boxed{\begin{gathered} \nabla_{\mathbf{w}} \mathbf{a} \cdot \mathbf{w} = \mathbf{a} \\ \nabla_{\mathbf{w}} \|\mathbf{w}\|_2^2 = \nabla_{\mathbf{w}} \mathbf{w} \cdot \mathbf{w} = 2\mathbf{w} \\ \nabla_{\mathbf{w}} \mathbf{w}^\top C \mathbf{w} = (C + C^\top)\mathbf{w} \end{gathered}}$$

Roadmap

Multiclass Classification

Matrix Calculus

Python

Recurrence Relation

Probability Theory

Syntactic Sugar

- List comprehension
- List slicing
- Passing functions
- Reading and writing files

Gotchas

- Integer division
- Tied objects
- Global variables

References

- Official Documentation (has a tutorial):

  https://docs.python.org/2.7/

- Learn X in Y minutes:

  http://learnxinyminutes.com/docs/python/

- You don't need to know `numpy`. But if you want to:

  http://nbviewer.ipython.org/gist/rpmuller/5920182

Roadmap

Multiclass Classification

Matrix Calculus

Python

Recurrence Relation

Probability Theory

Coin Payment

Problem

Suppose you have an unlimited supply of coins with values 2, 3, and 5 cents

How many ways can you pay for an item costing 12 cents?

- Is this problem well-defined?
- There is no mention of whether the order of coins matters or not.
- Let's consider both cases, where the order matters and where it does not matter.

Coin Payment

What if the order ...

... **matters**? ... **does not matter**?

Recurrence Relation: Break down into smaller problems

Memoization: Remember what you already calculated

- In the unordered case, there are only 5 distinct ways of counting to 12:
  (2, 2, 2, 2, 2, 2)
  (2, 2, 2, 3, 3)
  (2, 2, 3, 5)
  (2, 5, 5)
  (3, 3, 3, 3)
- Now, counting the unique permutations of every case for the ordered case, we see that that the first case produces **1** permutation, second case produces **10** permutations, third case produces **12** permutations, followed by **3** and **1** permutations by the last 2 cases, giving a total of **27** ways of getting 12.
- Refer to the extra section handout for more information regarding how the code computing these would look like.

Roadmap

Multiclass Classification

Matrix Calculus

Python

Recurrence Relation

Probability Theory

Random Variables

Discrete:

$$\mathbb{P}(A = a) \quad \text{or} \quad p_A(a)$$

Continuous:

$$\xcancel{\mathbb{P}(A = a)}$$

$$f_A(a)$$

$$\mathbb{P}(A \leq c) = \int_{--}^{c} f_A(a)\, da$$

- Discrete: random variable taking on discrete values with a probability distrubution.
- Continuous: random variable taking on a spectrum of values with a probability density distribution.
- Probability Density Function (PDF): used to calculate the probability of a random variable falling within a particular range of values.
- Cumulative Distribution Function (CDF): the probability that the random variable will take a value less than or equal to some value.

Random Variables

	A = 0	**A = 1**	**A = 2**	**A = 3**
B = 0	0.1	0.25	0.1	0.05
B = 1	0.15	0	0.15	0.2

- What is $\mathbb{P}(A = 2)$
- What is $\mathbb{P}(A = 2 \mid B = 1)$

- $\mathbb{P}(A = 2) = 0.1 + 0.15 = 0.25$
- $\mathbb{P}(A = 2|B = 1) = \frac{0.15}{0.15+0+0.15+0.2} = 0.3$

Random Variables

Independence:

$$\forall a, b, \quad \mathbb{P}(A = a, B = b) = \mathbb{P}(A = a)\mathbb{P}(B = b)$$

$$\forall a, b, \quad f_{A,B}(a, b) = f_A(a) f_B(b)$$

Expectation:

$$\mathbb{E}[A] = \sum_a a\,\mathbb{P}[A = a]$$

$$\mathbb{E}[A] = \int a f_A(a)\,da$$

Random Variables

	A = 0	**A = 1**	**A = 2**	**A = 3**
B = 0	0.1	0.25	0.1	0.05
B = 1	0.15	0	0.15	0.2

- Are A and B independent?
- What are $\mathbb{E}[A]$, $\mathbb{E}[B]$, $\mathbb{E}[A + B]$

Linearity of Expectation: $\mathbb{E}[A + B] = \mathbb{E}[A] + \mathbb{E}[B]$

True even when A and B are dependent!

- **A** and **B** are not independent. For proof, consider $\mathbb{P}(A = 0, B = 0)$, $\mathbb{P}(A = 0)$ and $\mathbb{P}(B = 0)$
- $\mathbb{E}[A] = 1.5$
- $\mathbb{E}[B] = 0.5$

Hat Toss

Problem

Suppose n hatted people toss their hats into the air and pick up one hat at random

In expectation, how many people get their own hats back?

Hint: linearity of expectation

- $X = X_1 + X_2 + ... + X_n$
- $X_i = \begin{cases} 1 & \text{if i selects own hat} \\ 0 & \text{otherwise} \end{cases}$
- $\mathbb{P}(X_i = 1) = \frac{1}{n}$
- $\mathbb{E}[X_i] = \frac{1}{n}$
- X_i are not independent, why?
- $\mathbb{E}[X] = n\frac{1}{n} = 1$

Random Variables

Variance:

$$\mathrm{Var}[A] = \mathbb{E}[(A - \mathbb{E}[A])^2] = \mathbb{E}[A^2] - \mathbb{E}[A]^2$$

Covariance:

$$\begin{aligned}\mathrm{Cov}[A, B] &= \mathbb{E}[(A - \mathbb{E}[A])(B - \mathbb{E}[B])] \\ &= \mathbb{E}[AB] - \mathbb{E}[A]\mathbb{E}[B]\end{aligned}$$

If $\mathrm{Cov}[A, B] = 0$, we say A and B are **uncorrelated**

Random Variables

If A and B are independent, then

- $\mathrm{Cov}[A, B] = \mathbb{E}[AB] - \mathbb{E}[A]\mathbb{E}[B] = 0$

  Independence implies uncorrelatedness

- $\mathrm{Var}[A + B] = \mathrm{Var}[A] + \mathrm{Var}[B]$

  Noise adds up

But the converse is **not** true!

Questions?