CS 221 Section 1

Foundations

Jerry Qu, Rohan Sampath

Contents

Random Variables

- **Discrete variable**: random variable taking on discrete values with a probability (mass) distribution

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

- **Continuous variable:** random variable taking on a range of values with a probability density distribution.

~~$P(A = a)$~~

- **Probability Density Function (PDF):** used to calculate the probability of a random variable falling within a particular range of values.

Probability Density Function $f_A(a)$ →

$$Pr[x \leq A \leq y] = \int_x^y f_A(a)da$$

- **Cumulative Distribution Function (CDF):** the probability that the random variable will take a value less than or equal to some value.

Cumulative Distribution Function $F_A(a)$ →

$$\mathrm{F}_A(a) = Pr[A \leq a] = \int_{-\infty}^{a} f_A(u)du$$

Independence and Expectation

Independence:

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

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

Expectation:

$$E[f(A)] = \sum_a f(a)P(A = a)$$

$$E[f(A)] = \int_a f(a)P(A = a)$$

Problem: Independence and Expectation

	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 $E[A]$, $E[B]$, $E[A + B]$

Linearity of Expectation: $E[A + B] = E[A] + E[B]$

True even when A and B are dependent!

Solution

- **A** and **B** are not independent. For proof, consider P($A = 0, B = 0$) = 0.1, P($A = 0$) = 0.25 and P($B = 0$) = 0.5
- E[A] = 1.5
- E[B] = 0.5
- E[A+B] = 2.0

Problem: Expectation

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

Solution

$$X = X_1 + X_2 + ... + X_n$$

$$X_i = \begin{cases} 1 & \text{if i selects own hat} \\ 0 & \text{otherwise} \end{cases}$$

$$P[X_i = 1] = \frac{1}{n}$$

$$E[X_i] = \frac{1}{n}$$

$$E[X] = E[X_1 + X_2 + ... + X_n] = E[X_1] + E[X_2] + ... + E[X_n] = 1$$

Note that the X_i are not independent of each other

Contents

Matrix Calculus

Here's some basic notation and properties:

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_d \end{bmatrix} \qquad A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{d1} & a_{d2} & a_{d3} & \dots & a_{dn} \end{bmatrix}$$

$$\mathbf{v}^2 = \|\mathbf{v}\|_2^2 = v \cdot v = v^\top v$$

$$(A + B)^\top = A^\top + B^\top$$

$$(AB)^\top = B^\top A^\top$$

Matrix Calculus

In this class, we will often be computing things with the gradient. Here are some useful properties. In general, properties you learned from calculus on scalars also apply to vectors.

$$\nabla f(\mathbf{v}) = \left(\frac{\partial f(\mathbf{v})}{\partial v_1}, \ldots, \frac{\partial f(\mathbf{v})}{\partial v_d} \right)^\top \quad \text{where } \mathbf{v} \in \mathbb{R}^d$$

$$\nabla_{\mathbf{v}}(cf(\mathbf{v})) = c\nabla_{\mathbf{v}}(f(\mathbf{v}))$$

$$\nabla(\mathbf{v}^\top \mathbf{v}) = 2\mathbf{v}$$

$$\nabla_{\mathbf{v}}(\mathbf{a}^\top \mathbf{v}) = \mathbf{a}$$

$$\nabla_{\mathbf{v}}(\mathbf{v}^\top C\mathbf{v}) = (C + C^\top)\mathbf{v}$$

Matrix Calculus

Often, when you have trouble understanding how to compute a derivative for vectors/matrices or why a property holds, it can be helpful to break it down. Here's an example for this property:

$$\nabla(\mathbf{v}^\top \mathbf{v}) = 2\mathbf{v}$$

For this property, we have that:

$$\frac{\partial}{\partial v_i}\mathbf{v}^\top \mathbf{v} = \frac{\partial}{\partial v_i}(v_1^2 + \cdots + v_d^2) = 2v_i$$

Thus,

$$\begin{aligned}\nabla(\mathbf{v}^\top \mathbf{v}) &= \left(\frac{\partial}{\partial v_1}(\mathbf{v}^\top \mathbf{v}), \ldots, \frac{\partial}{\partial v_d}(\mathbf{v}^\top \mathbf{v})\right)^\top \\ &= (2v_1, \ldots, 2v_d)^\top = 2\mathbf{v}\end{aligned}$$

Problem: Loss Functions and Gradient Calculations

Would the following loss functions be suitable to use to solve a binary classification problem using Stochastic Gradient Descent? If not, explain why. If yes, compute the gradient Δ_w :

$$\textbf{Loss}(x, y, \mathbf{w}) = \max\{1 - \lfloor(\mathbf{w} \cdot \phi(x))y\rfloor, 0\}$$

$$\textbf{Loss}(x, y, \mathbf{w}) = \begin{cases} 1 - 2(\mathbf{w} \cdot \phi(x))y & \text{if } (\mathbf{w} \cdot \phi(x))y \leq 0 \\ (1 - (\mathbf{w} \cdot \phi(x))y)^2 & \text{if } 0 < (\mathbf{w} \cdot \phi(x))y \leq 1 \\ 0 & \text{if } (\mathbf{w} \cdot \phi(x))y > 1 \end{cases}$$

Solution

Would the following loss functions be suitable to use to solve a binary classification problem using Stochastic Gradient Descent? If not, explain why. If yes, compute the gradient Δ_w :

$$\mathbf{Loss}(x, y, \mathbf{w}) = \max\{1 - \lfloor(\mathbf{w} \cdot \phi(x))y\rfloor, 0\}$$

No. because the gradient is 0 almost everywhere (the floor operation makes it a step function), so we cannot apply gradient descent.

Solution

Would the following loss functions be suitable to use to solve a binary classification problem using Stochastic Gradient Descent? If not, explain why. If yes, compute the gradient Δ_w :

$$\mathbf{Loss}(x, y, \mathbf{w}) = \begin{cases} 1 - 2(\mathbf{w} \cdot \phi(x))y & \text{if } (\mathbf{w} \cdot \phi(x))y \leq 0 \\ (1 - (\mathbf{w} \cdot \phi(x))y)^2 & \text{if } 0 < (\mathbf{w} \cdot \phi(x))y \leq 1 \\ 0 & \text{if } (\mathbf{w} \cdot \phi(x))y > 1 \end{cases}$$

Yes, it is suitable. Gradient is as follows:

$$\nabla_{\mathbf{w}}\mathbf{Loss}(x, y, \mathbf{w}) = \begin{cases} -2\phi(x)y & \text{if } (\mathbf{w} \cdot \phi(x))y \leq 0 \\ -2(1 - (\mathbf{w} \cdot \phi(x))y)\phi(x)y & \text{if } 0 < (\mathbf{w} \cdot \phi(x))y \leq 1 \ \$ \\ 0 & \text{if } (\mathbf{w} \cdot \phi(x))y > 1 \end{cases}$$

Note that the loss function is continuously differentiable at all points.

Contents

Probability Theory

Matrix Calculus

Python

Time Complexity

Recurrence Relations

Python

We will primarily be using **python 3.7**.

Official Documentation: https://docs.python.org/3.7/

Another good resource: https://learnxinyminutes.com/docs/python/

Anaconda may be useful if you don't have python 3.7 or don't want to impact existing python installations: https://www.anaconda.com/distribution/

Note: We don't use libraries e.g. numpy. **Please don't import libraries without checking first!** Past students have imported and used numpy functions, breaking the autograder and getting 0 points! It can be helpful to submit early.

Python

```
a = "Can I skip a CS221 homework? No dice!!"
```

The **split** command creates a list from a string with blank spaces as delimiters by default. You can also specify a different delimiter.

```
b = a.split()
b
```

```
['Can', 'I', 'skip', 'a', 'CS221', 'homework?', 'No', 'dice!!']
```

```
c = [len(_) for _ in b]
c
```

```
[3, 1, 4, 1, 5, 9, 2, 6]
```

Note that `_` is a valid variable name (usually for one-time use). An equivalent for-loop would be:

```
c = []
for _ in b:
    c.append(len(_))
c
```

```
[3, 1, 4, 1, 5, 9, 2, 6]
```

Python

```
# Task: Find sum of X values less than 5

points = [(1, 2), (2, 6), (3, 3), (8, 9)]

X_total = 0
for point in points:
    if point[0] < 5:
        X_total = X_total + point[0]

X_total
```

```
6
```

```
sum([x for x, _ in points if x < 5])
```

```
6
```

enumerate is a useful built-in:

```
a = enumerate(['a', 'b', 'c'])

list(enumerate(['a', 'b', 'c']))
```

```
[(0, 'a'), (1, 'b'), (2, 'c')]
```

Python

```
c = [len(_) for _ in b]

c
```

```
[3, 1, 4, 1, 5, 9, 2, 6]
```

```
c[2]    # The index starts from 0
```

```
4
```

```
c[-1]   # You can count backward from the right: c[-1] is the last element
```

```
6
```

```
c[1:4]  # Indexes are inclusive:exclusive
```

```
[1, 4, 1]
```

```
c[:4]   # = c[0:4]
```

```
[3, 1, 4, 1]
```

```
c[4:]   # = c[4:len(c)]
```

```
[5, 9, 2, 6]
```

```
c[:-1] # = c[0:len(c)-1], cutting out the last element in the list
```

```
[3, 1, 4, 1, 5, 9, 2]
```

Contents

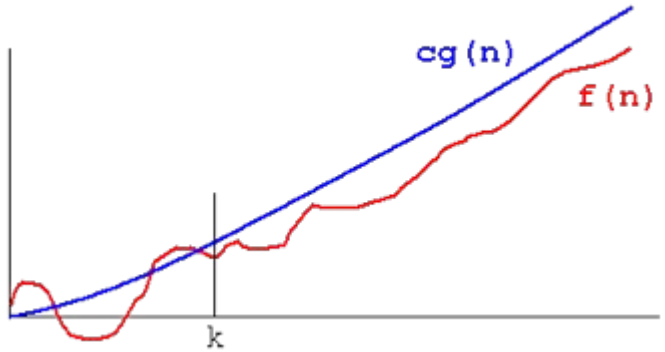
Time Complexity

Time complexity refers to how long something takes. Formally, we say that

$$f(n) \text{ is } O(g(n)) \text{ if there exist positive constants } C \text{ and } k \text{ such that } 0 \leq f(n) \leq Cg(n) \quad \forall n \geq k$$

Intuitively, a function is big O of g(n) if the time to execute that function grows no more than a constant multiple of g(n).

This means, we don't care about constant multiples. We care about how a function grows as the input size increases.



Time Complexity

Binary search

Enumerating the elements of an *n* by *n* matrix

f(n) = 2log(n) + n

```
for i = 1 to n - 1:
    for j = 1 to n - i:
        if arr[j] > arr[j+1]:
            swap(arr[j], arr[j+1])
```

Time Complexity

Binary search **O(log(n))**

Enumerating the elements of an *n* by *n* matrix

f(n) = 2log(n) + n

```
for i = 1 to n - 1:
    for j = 1 to n - i:
        if arr[j] > arr[j+1]:
            swap(arr[j], arr[j+1])
```

Time Complexity

Binary search **O(log(n))**

Enumerating the elements of an *n* by *n* matrix **O(n^2)**

f(n) = 2log(n) + n

```
for i = 1 to n - 1:
    for j = 1 to n - i:
        if arr[j] > arr[j+1]:
            swap(arr[j], arr[j+1])
```

Time Complexity

Binary search **O(log(n))**

Enumerating the elements of an *n* by *n* matrix **O(n^2)**

f(n) = 2log(n) + n **O(n)**

```
for i = 1 to n - 1:
    for j = 1 to n - i:
        if arr[j] > arr[j+1]:
            swap(arr[j], arr[j+1])
```

Time Complexity

Binary search **O(log(n))**

Enumerating the elements of an *n* by *n* matrix **O(n^2)**

f(n) = 2log(n) + n **O(n)**

O(n^2) (note that 1 + 2 + … + k = k(k - 1)/2)

```
for i = 1 to n - 1:
    for j = 1 to n - i:
        if arr[j] > arr[j+1]:
            swap(arr[j], arr[j+1])
```

Contents

Recurrence Relations

Suppose you have an unlimited supply of coins with values 2, 3, and 5 cents. You put coins into a machine.

How many ways can you pay for an item costing 12 cents? (i.e., order matters)
How about n cents?

Recurrence Relations

Recall: in lecture we discussed breaking this down into smaller subproblems.

For 12 cents, we can simply enumerate:

{2, 2, 2, 2, 2, 2}
{2, 2, 2, 3, 3}
{3, 3, 3, 3}
{2, 2, 3, 5}
{2, 5, 5}

For n cents, it is a little trickier!

Recurrence Relations

Recall: in lecture we discussed breaking this down into smaller subproblems.

For 12 cents, we can simply enumerate:

{2, 2, 2, 2, 2, 2} - 1
{2, 2, 2, 3, 3} - 5 choose 2 = 10
{3, 3, 3, 3} - 1
{2, 2, 3, 5} - 4! / 2 = 12
{2, 5, 5} - 3 choose 1 = 3

Summing, we get 1 + 10 + 1 + 12 + 3 = **27**

For n cents, it is a little trickier!

Recurrence Relations

Recall: in lecture we discussed breaking this down into smaller subproblems.

For 12 cents, we can simply enumerate:

{2, 2, 2, 2, 2, 2} - 1
{2, 2, 2, 3, 3} - 5 choose 2 = 10
{3, 3, 3, 3} - 1
{2, 2, 3, 5} - 4! / 2 = 12
{2, 5, 5} - 3 choose 1 = 3 Summing, we get 1 + 10 + 1 + 12 + 3 = **27**

For n cents, it is a little trickier!

Some questions to consider:
What information would we need to solve the problem for n cents?

How can we get that information?

Recurrence Relations

Recall: in lecture we discussed breaking this down into smaller subproblems.

For 12 cents, we can simply enumerate:

{2, 2, 2, 2, 2, 2} - 1
{2, 2, 2, 3, 3} - 5 choose 2 = 10
{3, 3, 3, 3} - 1
{2, 2, 3, 5} - 4! / 2 = 12
{2, 5, 5} - 3 choose 1 = 3 Summing, we get 1 + 10 + 1 + 12 + 3 = **27**

For n cents, it is a little trickier!

Some questions to consider:
What information would we need to solve the problem for n cents?
The number of ways to get n-2, n-3, & n-5 cents! Then we can just add a single coin.
How can we get that information?
We compute from 1, 2, 3, etc.!

Questions?