HW1_diegoek2

January 30, 2021

1 STATS 542: Homework 1

Diego Kleiman (diegoek2)

Due: Tuesday 11:59 PM CT, Feb 2nd

1.1 Basic calculus

1. Calculate the derivative of $f(x)$
   (a) $f(x) = e^x$
   (b) $f(x) = \log(1 + x)$
   (c) $f(x) = \log(1 + e^x)$

Answer:

(a)
$$f'(x) = e^x$$

(b)
$$f'(x) = \frac{1}{1+x}$$

(c)
$$f'(x) = \frac{e^x}{1+e^x}$$

2. Taylor expansion. Let f: $\mathbb{R} \to \mathbb{R}$ be a twice differentiable function. Please write down the first three terms of its Taylor expansion at point $x = 1$.

Answer:
$$f(x) \approx f(1) + (x-1) \cdot f'(1) + (x-1)^2 \cdot \frac{f''(1)}{2}$$

3. For the infinite sum $\sum_{n=1}^{\infty} \frac{1}{n^\alpha}$, where α is a positive real number, give the exact range of α such that the series converges.

Answer:
$$\alpha \in (1, \infty)$$

1.2 Linear algebra

1. What is the eigendecomposition of a real symmetric matrix $A_{n\times n}$? Write down one form of that decomposition and explain each term in your formula. Based on these terms, derive $A^{-1/2}$.

Answer:

The eigendecomposition of a real symmetric matrix $A_{n\times n}$ is $A = Q\Lambda Q^T$ where Q is an orthonormal matrix formed by the eigenvectors of A and Λ is a diagonal matrix that contains the eigenvalues of A.

Since Λ is diagonal, we can just apply the $-1/2$ power to each element of the matrix individually, and then get $A^{-1/2}$ using

$$A^{-1/2} = Q\Lambda^{-1/2}Q^T$$

2. What is a symmetric positive definite matrix $A_{n\times n}$? Give one of equivalent definitions and explain your notation.

Answer:

A symmetric positive definite matrix A is a symmetric matrix ($A_{ij} = A_{ji}$) such that the expression $v^T Av$ always evaluates to a positive number for any real-valued vector v of size n. If a matrix is positive definite, all its eigenvalues are positive.

3. True/False. If you claim a statement is false, explain why. For two real matrices $A_{m\times n}$ and $B_{n\times m}$
   (a) Rank(A) = max$\{m, n\}$
   (b) If $m = n$, then trace(A) = $\sum_{i=1}^{n} A_{ii}$
   (c) If A is a symmetric matrix, then all eigenvalues of A are real
   (d) If A is a symmetric matrix, λ_1 and λ_2 are two of its eigen-values (not necessarily different) and v_1,v_2 are the corresponding eigen-vectors, then $v_1^T v_2 = 0$.
   (e) trace(ABAB) = trace(AABB)

Answer:

(a) False. The rank can be lower if the matrix has one or more redundant (collinear) vectors. And if all the vectors are linearly independent, the correct expression would be Rank(A) = min$\{m, n\}$.

(b) True.

(c) True.

(d) False. Eigenvectors of the same eigenvalue can be chosen to be mutually orthogonal for a symmetric matrix, but it's not required.

(e) False. Cyclic permutations of matrix product preserve the trace, but other permutations don't.

1.3 Statistics

1. X_1, X_2, ..., X_n are i.i.d. $\mathcal{N}(\mu, \sigma^2)$ random variables, where $\mu \in \mathbb{R}$ and $\sigma > 0$ is finite. Let $\bar{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i$.
   (a) What is an unbiased estimator? Is $\bar{X}_n$ an unbiased estimator of μ?
   (b) What is $E[(\bar{X}_n)^2]$ in terms of n, μ, σ?
   (c) Give an unbiased estimator of σ^2.
   (d) What is a consistent estimator? Is $\bar{X}_n$ a consistent estimator of μ?

Answer:

(a) An unbiased estimator is an estimator whose expected value equals the estimated parameter. $\bar{X}_n$ is an unbiased estimator of μ as shown here:

$$E[\bar{X}_n] = E\left[\frac{1}{n}\sum_{i=1}^{n} X_i\right] = \frac{1}{n}\sum_{i=1}^{n} E[X_i] = \frac{1}{n}\sum_{i=1}^{n} \mu = \frac{n}{n}\mu = \mu$$

(b) To solve this question, I will use the facts that

$$Var(\bar{X}_n) = E[(\bar{X}_n)^2] - \mu^2$$

and

$$Var(\bar{X}_n) = Var\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) = \frac{1}{n^2}\sum_{i=1}^{n} Var(X_i) = \frac{\sigma^2}{n}$$

Then the answer comes from solving for $E[(\bar{X}_n)^2]$

$$E[(\bar{X}_n)^2] = \frac{\sigma^2}{n} + \mu^2$$

(c)

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X}_n)^2$$

(d) A consistent estimator is an estimator whose value converges to the value of the estimated parameter as the size of the sample grows arbitrarily large. In other words, for an estimator to be consistent, the probability that the estimator equals the parameter must converge to one when the sample size tends to infinity. $\bar{X}_n$ is a consistent estimator of μ. To see this intuitively, we can recall that $Var(\bar{X}_n) = \frac{\sigma^2}{n}$, so as $n \to \infty$ the probability distribution of $\bar{X}_n$ (Gaussian) has arbitrarily small variance and tends to a single point at $\bar{X}_n = \mu$ with probability 1.

2. Suppose $X_{p\times 1}$ is a vector of covariates, $\beta_{p\times 1}$ is a vector of unknown parameters, ϵ is the unobserved random noise and we assume the linear model relationship $y = X^T\beta + \epsilon$. Suppose we have n i.i.d. samples from this linear model, and the observed data can be written using the matrix form: $\mathbf{y}_{n\times 1} = \mathbf{X}_{n\times p}\beta_{p\times 1} + \boldsymbol{\epsilon}_{n\times 1}$.

(a) If we want estimate the unknown β using a least square method, what is the objective function $L(\beta)$ to obtain $\widehat{\beta}$?

(b) What is the solution of $\widehat{\beta}$? Represent the solution using the observed data $\mathbf{y}$ and $\mathbf{X}_{n\times p}$. Note that you may assume that $\mathbf{X}^T\mathbf{X}$ is invertible.

Answer:

(a) $L(\beta) = \sum_{i=1}^{n}(y_i - X_i^T\beta)^2$ or in matrix form $L(\beta) = (\mathbf{y}_{n\times 1} - \mathbf{X}_{n\times p}\beta_{p\times 1})^T(\mathbf{y}_{n\times 1} - \mathbf{X}_{n\times p}\beta_{p\times 1})$

(b) $\widehat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$

1.4 Programming

1. Use the following code to generate a set of observations $\mathbf{y}$ and $\mathbf{X}_{n\times p}$. Following the previously established formula, Write your own code, instead of using existing functions such as `lm()`, to solve for the least square estimator $\widehat{\beta}$. If you are asked to add an intercept term β_0 into your estimation (even the true $\beta_0 = 0$ in our data generator), what should you do?

```
set.seed(1)
n = 100; p = 5
X = matrix(rnorm(n * p), n, p)
y = X %*% c(1, 0, 0, 1, -1) + rnorm(n)
```

[1]:
```
# Python implementation for least squares minimization
import numpy as np

def least_squares(X, y):
    '''
    Returns optimal parameters for linear model using least squares␣
 ↪optimization.

    Args:
        X (np.ndarray): covariates (nxp)
        y (np.ndarray): observed data (nx1)
    Returns:
        np.ndarray: linear fit parameters (px1)
    '''
    prod1 = np.matmul(X.T, X)
    inverse = np.linalg.inv(prod1)
    prod2 = np.matmul(X.T, y)

    return np.matmul(inverse, prod2)
```

[2]:
```
np.random.seed(1)
n = 100
p = 5
X = np.random.rand(n, p) # Random matrix of numbers sampled from uniform␣
 ↪distribution over [0, 1) of dimensions nxp
```

```
y = np.matmul(X, np.asarray([1, 0, 0, 1, -1])) + np.random.rand(n)

# Get parameters
beta_hat = least_squares(X, y)
print(beta_hat)
```

```
[ 1.04593163  0.27957623  0.24446713  1.08330839 -0.79577408]
```

In order to take β_0 into account, I should insert an extra column of ones in my X matrix such that the β_0 parameter is added in the expression.

[3]:
```
# Python implementation for least squares minimization with intercept term

def least_squares_constant_term(X, y):
    '''
    Returns optimal parameters for linear model using least squares␣
 ↪optimization adding the beta naught term.
    Note: The intercept term is the last one.

    Args:
        X (np.ndarray): covariates (nxp)
        y (np.ndarray): observed data (nx1)
    Returns:
        np.ndarray: linear fit parameters (px1)
    '''
    ones = np.ones((X.shape[0], 1))
    X = np.append(X, ones, axis=1)

    prod1 = np.matmul(X.T, X)
    inverse = np.linalg.inv(prod1)
    prod2 = np.matmul(X.T, y)

    return np.matmul(inverse, prod2)
```

[4]:
```
np.random.seed(1)
n = 100
p = 5
X = np.random.rand(n, p) # Random matrix of numbers sampled from uniform␣
 ↪distribution over [0, 1) of dimensions nxp
y = np.matmul(X, np.asarray([1, 0, 0, 1, -1])) + np.random.rand(n)

# Get parameters
beta_hat = least_squares_constant_term(X, y)
print(beta_hat)
```

```
[ 0.89186666  0.06311393  0.09816809  0.91739109 -1.00825479  0.48634907]
```

2. Perform a simulation study to check the consistency of the sample mean estimator $\bar{X}_n$. Please

save your random seed so that the results can be replicated by others.

(a) Generate a set of $n = 20$ i.i.d. observations from uniform (0, 1) distribution and calculate the sample mean $\bar{X}_n$

(b) Repeat step (a) 1000 times to collect 1000 such sample means and plot them using a histogram.

(c) How many of such sample means (out of 1000) are at least 0.1 away from true mean parameter, which is 0.5 for uniform (0, 1)?

(d) Repeat steps (a) to (c) with $n = 100$ and $n = 500$. What conclusion can you make?

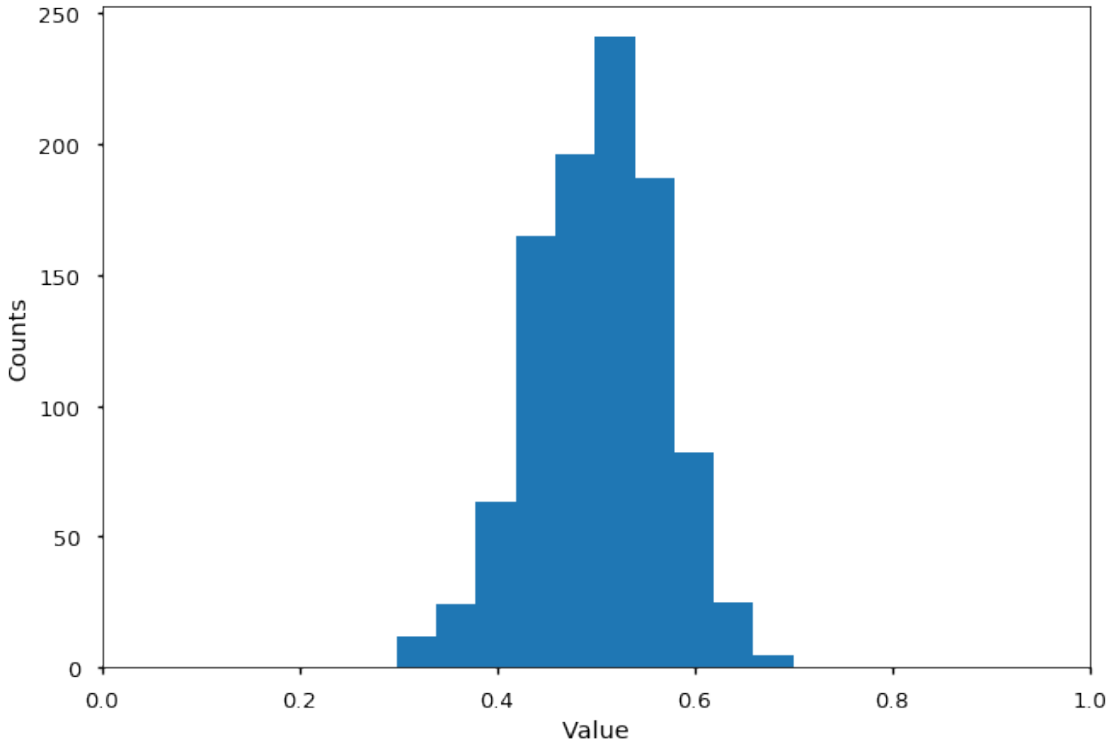
[5]:
```
import numpy as np
from matplotlib import pyplot as plt

np.random.seed(1)

# Parts (a) and (b)
def sample_n(n):
    '''
    Returns 1000 means sampled from n i.i.d. observations from uniform (0, 1)␣
 ↪distribution.
    '''
    sample_means = np.empty(1000)
    for i in range(1000):
        obs = np.random.rand(n) # i.i.d. observations from uniform (0, 1)␣
 ↪distribution
        sample_means[i] = obs.mean()
    return sample_means

sample_20 = sample_n(20)
```

[6]:
```
plt.style.use('seaborn-talk')
plt.hist(sample_20)
plt.xlabel("Value")
plt.ylabel("Counts")
plt.xlim(0, 1)
plt.show()
```



[7]:
```
# Part (c)

def test_dist(sample_means):
    '''
    Returns number of means that are at least 0.1 from true distribution mean␣
 ↪(0.5).
    '''
    test = np.abs(sample_means - 0.5) >= 0.1
    return np.count_nonzero(test)

count = test_dist(sample_20)
print(count)
```

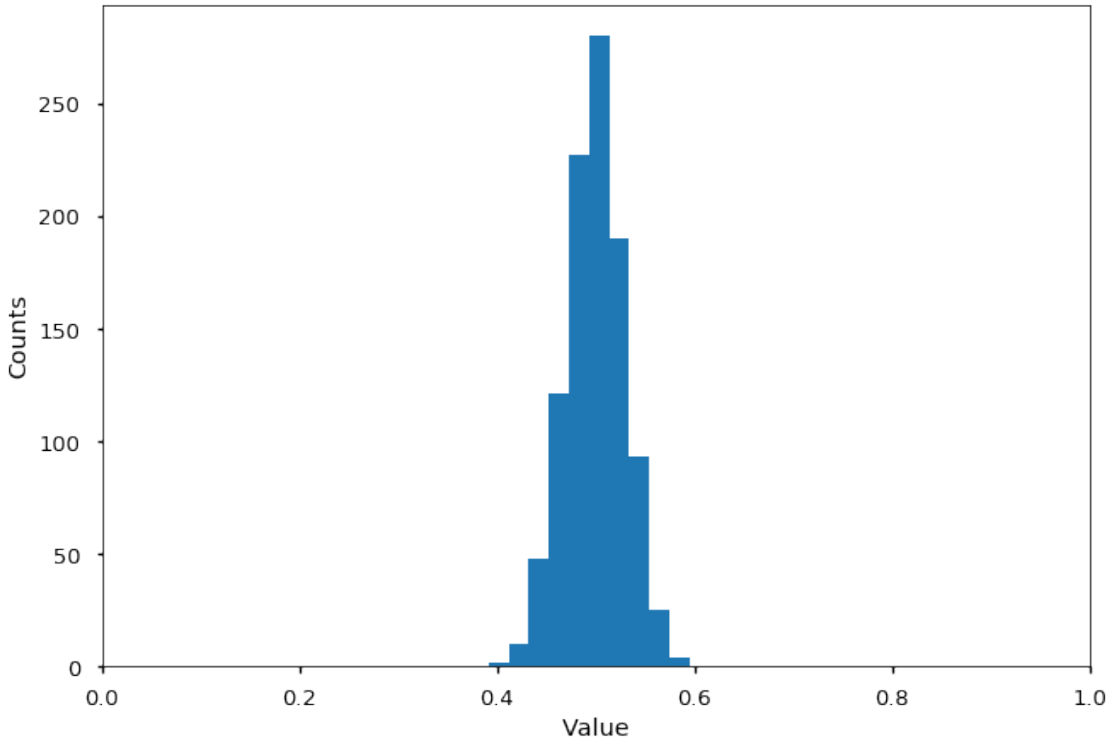
```
120
```

There are 120 samples at a distance of at least 0.1 from the mean.

[8]:
```
# Part (d)
# n = 100
np.random.seed(1)
sample_100 = sample_n(100)

plt.style.use('seaborn-talk')
plt.hist(sample_100)
```

```
plt.xlabel("Value")
plt.ylabel("Counts")
plt.xlim(0, 1)
plt.show()
```



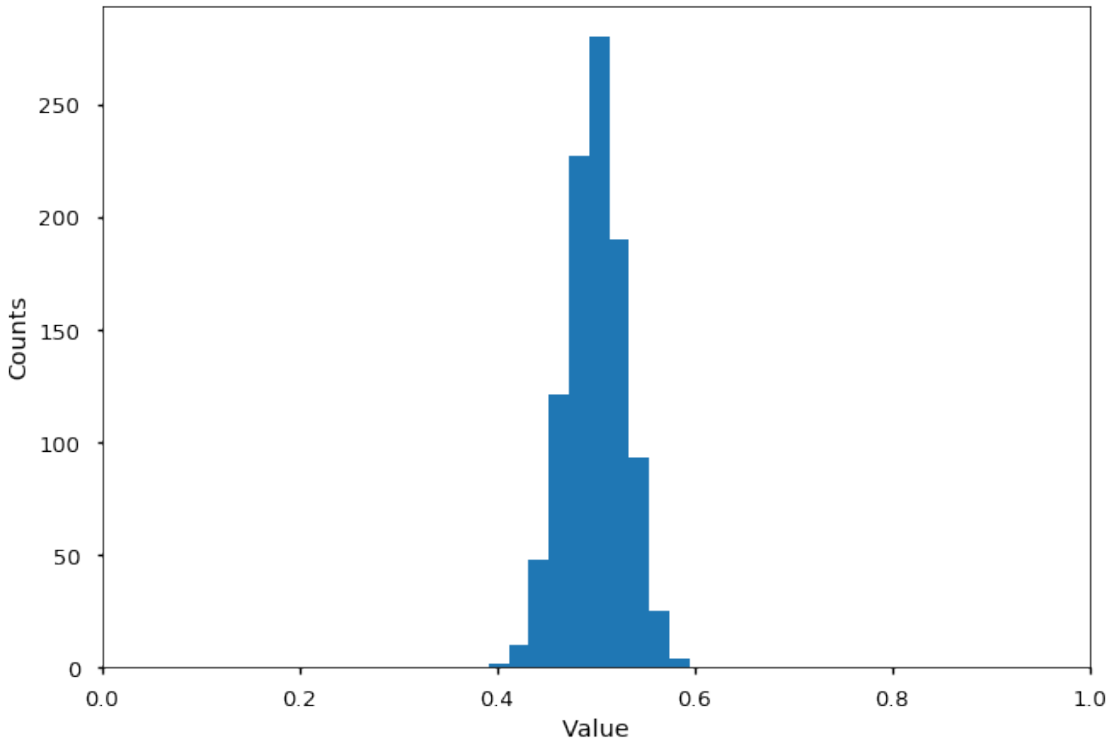
[10]:
```
count = test_dist(sample_100)
print(count)
```

2

There are 2 samples at a distance of at least 0.1 from the true mean.

[9]:
```
# n = 500
np.random.seed(1)
sample_500 = sample_n(500)

plt.style.use('seaborn-talk')
plt.hist(sample_100)
plt.xlabel("Value")
plt.ylabel("Counts")
plt.xlim(0, 1)
plt.show()
```



[11]:
```
count = test_dist(sample_500)
print(count)
```

0

There are no samples at a distance of at least 0.1 from the true mean.

We can conclude that the estimator is unbiased (the value of the true mean is accurate) and consistent (increasing the size of the sample reduces the error).