6.7900 Fall 2024: Homework 0
Solutions

This is a set of diagnostic and warm-up problems, divided into two parts.

1. *To be handed in*: Problems in the first part represent basic background in linear algebra, applied math, and optimization. They aren't trivial, but if they aren't relatively easy for you, then it might be better to gain some more background in a prerequisite area before taking 6.7900.

2. *To be used for your own practice*: Problems in the second part are designed to help you learn/practiced numpy-style vectorized programming strategies, which will help you create efficient implementations of algorithms studied in class and also to interface with existing libraries. Please solve these problems using numpy, striving for elegant and efficient solutions.

   If you don't have a Python/numpy installation on your own computer (or even if you do!) Google Colab (`https://colab.google/`) is a good way to get going quickly.

Submission instructions

Please hand in your work via Gradescope via the link at `https://gradml.mit.edu/info/homeworks/`. If you were not added to the course automatically, please use Entry Code R7RGGX to add yourself to Gradescope.

- Latex is not required, but if you are hand-writing your solutions, **please** write clearly and carefully. You should include enough work to show how you derived your answers, but you don't have to give careful proofs.
- Homework is due on Tuesday September 10 at 11PM.
- Lateness and extension policies are described at `https://gradml.mit.edu/info/class_policy/`.

1 Math Background

1.1 Just plane fun

Consider a hyperplane in n-dimensional Euclidean space, described by the $n+1$ real values w_i for $i = 0, \ldots, n$: the hyperplane consists of points $(x_1, \ldots, x_n)$ satisfying

$$w_0 + w_1 x_1 + \ldots + w_n x_n = 0 \ .$$

1. Find a unit vector normal to the hyperplane. Given a point $\boldsymbol{v} = (v_1, \ldots, v_n)$ on the hyperplane, give the equation for the line through the point that is orthogonal to the hyperplane.

   Solution: Let $\boldsymbol{w} = (w_1, \ldots, w_n)$, then $\boldsymbol{w}/\|\boldsymbol{w}\|$ is a unit vector normal to the hyperplane. To prove this, given any two points $\boldsymbol{x}$ and $\boldsymbol{x}'$ on the hyper plane, we will show that $\boldsymbol{w}$ is perpendicular to $\boldsymbol{x} - \boldsymbol{x}'$ by showing their dot-product is 0:

   $$\boldsymbol{w}^T(\boldsymbol{x} - \boldsymbol{x}') = \sum_{i=1}^{n} w_i x_i - \sum_{i=1}^{n} w_i x_i' = (-w_0) - (-w_0) = 0$$

2. Given a point $\boldsymbol{v} = (v_1, \ldots, v_n)$, how can you determine which side of the hyperplane it is on?

   Solution: Let $f(\boldsymbol{x}) = w_0 + \boldsymbol{w}^T\boldsymbol{x} = w_0 + w_1 x_1 + \cdots + w_n x_n$. Then $f(\boldsymbol{v}) = 0$ means $\boldsymbol{v}$ is on the hyperplane; $f(\boldsymbol{v}) > 0$ means $\boldsymbol{v}$ is on one side of the hyperplane, while $f(\boldsymbol{v}) < 0$ means it is on the other side of the hyperplane.

3. What is the distance of a point $\boldsymbol{v} = (v_1, \ldots, v_n)$ to the hyperplane?

   Solution: Assume $\bar{\boldsymbol{v}}$ is the projection of $\boldsymbol{v} = (v_1, \ldots, v_n)$ on the hyperplane. Since $\boldsymbol{w}/\|\boldsymbol{w}\|$ is the unit normal vector of the hyperplane, we know that

   $$\boldsymbol{v} = \bar{\boldsymbol{v}} + t\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$$

   where $|t|$ is the distance of $\boldsymbol{v}$ to the hyperplane. We also know $f(\bar{\boldsymbol{v}}) = 0$ by the previous question, that means

   $$0 = f(\bar{\boldsymbol{v}}) = f\left(\boldsymbol{v} - t\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}\right) = w_0 + \boldsymbol{w}^T\boldsymbol{v} - t\frac{\boldsymbol{w}^T\boldsymbol{w}}{\|\boldsymbol{w}\|} = w_0 + \boldsymbol{w}^T\boldsymbol{v} - t\|\boldsymbol{w}\|$$

Therefore, the distance is given by

$$|t| = \left| \frac{w_0 + \boldsymbol{w}^T \boldsymbol{v}}{\|\boldsymbol{w}\|} \right| = \frac{|f(\boldsymbol{v})|}{\|\boldsymbol{w}\|}$$

1.2 Multivariate Gaussian

Let X be a random variable taking values in $\mathbb{R}^n$. It is normally distributed with mean $\boldsymbol{\mu}$ and covariance matrix Σ. Recall that the probability density function (pdf) $p_X(\boldsymbol{x})$, sometimes denoted $p(X = \boldsymbol{x})$, for X is given by

$$p_X(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \exp\left(-\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) \right)$$

1. Show how to obtain the normalization constant $1/\sqrt{(2\pi)^n |\Sigma|}$ for the multivariate Gaussian, starting from the fact that

$$p_X(\boldsymbol{x}) \propto \exp\left(-\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) \right)$$

Hints: It's fine to assume $\boldsymbol{\mu} = 0$ (Why?). A useful (and cool!) fact is that $\int_{\mathbb{R}} \exp(-\frac{1}{2}x^2) = \sqrt{2\pi}$.

Solution: To obtain the normalization constant we need to evaluate the integral

$$\int_{\mathbb{R}^n} \exp\left(-\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) \right) d\boldsymbol{x}.$$

Observe that wlog we can assume $\boldsymbol{\mu} = 0$. Thus, it remains to evaluate

$$\int_{\mathbb{R}^n} \exp\left(-\tfrac{1}{2} \boldsymbol{x}^T \Sigma^{-1} \boldsymbol{x} \right) d\boldsymbol{x} \tag{1}$$

Let $\boldsymbol{y} = \Sigma^{-1/2} \boldsymbol{x}$, then by Jacobian transformation (1) can be rewritten as

$$\begin{aligned} \int_{\mathbb{R}^n} \exp\left(-\tfrac{1}{2} \boldsymbol{y}^T \boldsymbol{y} \right) |\Sigma^{1/2}| d\boldsymbol{y} &= |\Sigma^{1/2}| \prod_{i=1}^{n} \int_{\mathbb{R}} \exp\left(-\tfrac{1}{2} y_i^2 \right) dy_i \\ &= |\Sigma^{1/2}| \prod_{i=1}^{n} \sqrt{2\pi} \end{aligned}$$

We use the fact that $\int_{\mathbb{R}} \exp(-\frac{1}{2}x^2) = \sqrt{2\pi}$, then using the fact that $\int p(\boldsymbol{x}) = 1$, we conclude that the normalization constant is $\frac{1}{\sqrt{(2\pi)^n |\Sigma|}}$.

2. Let the random variable $Y = 2X$. What is the pdf of Y?

Solution: By the rule of probability density function for change of variable,

$$p_Y(\boldsymbol{y}) = \left(\frac{1}{2}\right)^n \frac{1}{\sqrt{(2\pi)^n|\Sigma|}} \exp\left(-\frac{1}{2}\left(\frac{\boldsymbol{y}}{2} - \mu\right)^T \Sigma^{-1}\left(\frac{\boldsymbol{y}}{2} - \mu\right)\right)$$
$$= \frac{1}{\sqrt{(2\pi)^n|4\Sigma|}} \exp\left(-\frac{1}{2}\left(\boldsymbol{y} - 2\mu\right)^T (4\Sigma)^{-1}\left(\boldsymbol{y} - 2\mu\right)\right)$$

So Y is still normally distributed, with mean 2μ and covariance matrix 4Σ.

3. What can we say about the distribution of X if Σ is the identity matrix, I? Does this imply anything about factorization of the pdf?

Solution: When Σ is the identity matrix,

$$p_X(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^n}} \exp\left(-\frac{1}{2}\sum_{i=1}^n (x_i - \mu_i)^2\right) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(x_i - \mu_i)^2\right)$$

The factorization of the pdf implies that $X_1, \ldots, X_n$ are now independent random variables.

4. What can we say about the distribution of X if Σ is $\begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix}$? Approximately what shape do equiprobability contours (i.e., sets $\{x \in R^n : p_X(x) = c\}$ for some c) have?

Solution: Although the covariance matrix is not a (scaled) identity matrix, the off-diagonal elements are still zero, so the two components of the 2D random vector are still independent and the pdf factorizes. Specifically, we have

$$p_X(\boldsymbol{x}) = \frac{\exp\left(-\frac{1}{2\times 10}(x_1 - \mu_1)^2\right)}{\sqrt{2\pi \times 10}} \cdot \frac{\exp\left(-\frac{1}{2}(x_2 - \mu_2)^2\right)}{\sqrt{2\pi}}$$

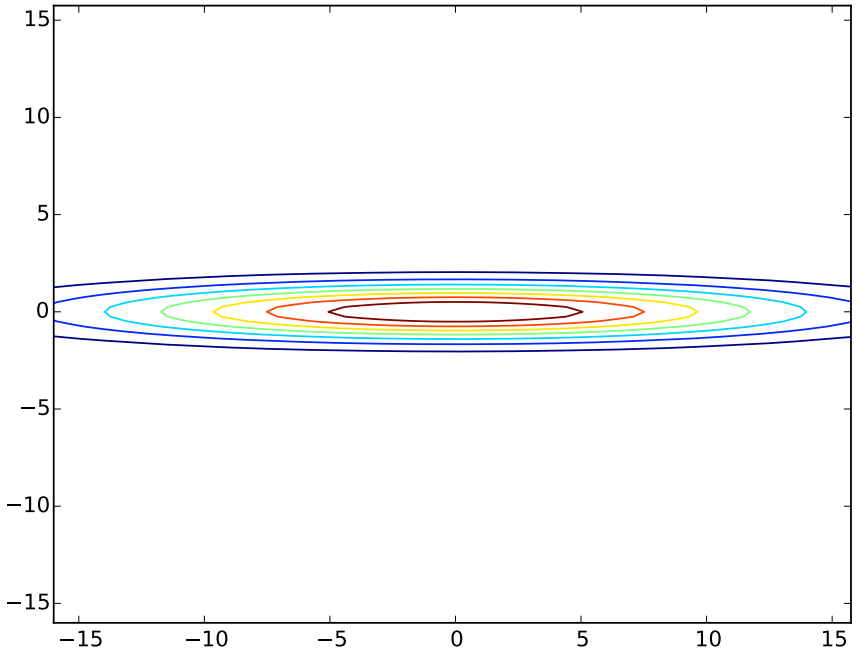
Let

$$c = p_X(\boldsymbol{x}) = \frac{\exp\left(-\frac{1}{2\times 10}(x_1 - \mu_1)^2\right)}{\sqrt{2\pi \times 10}} \cdot \frac{\exp\left(-\frac{1}{2}(x_2 - \mu_2)^2\right)}{\sqrt{2\pi}}$$

Via simple manipulation, we get the equation

$$\frac{1}{10}(x_1 - \mu_1)^2 + (x_2 - \mu_2)^2 = C$$

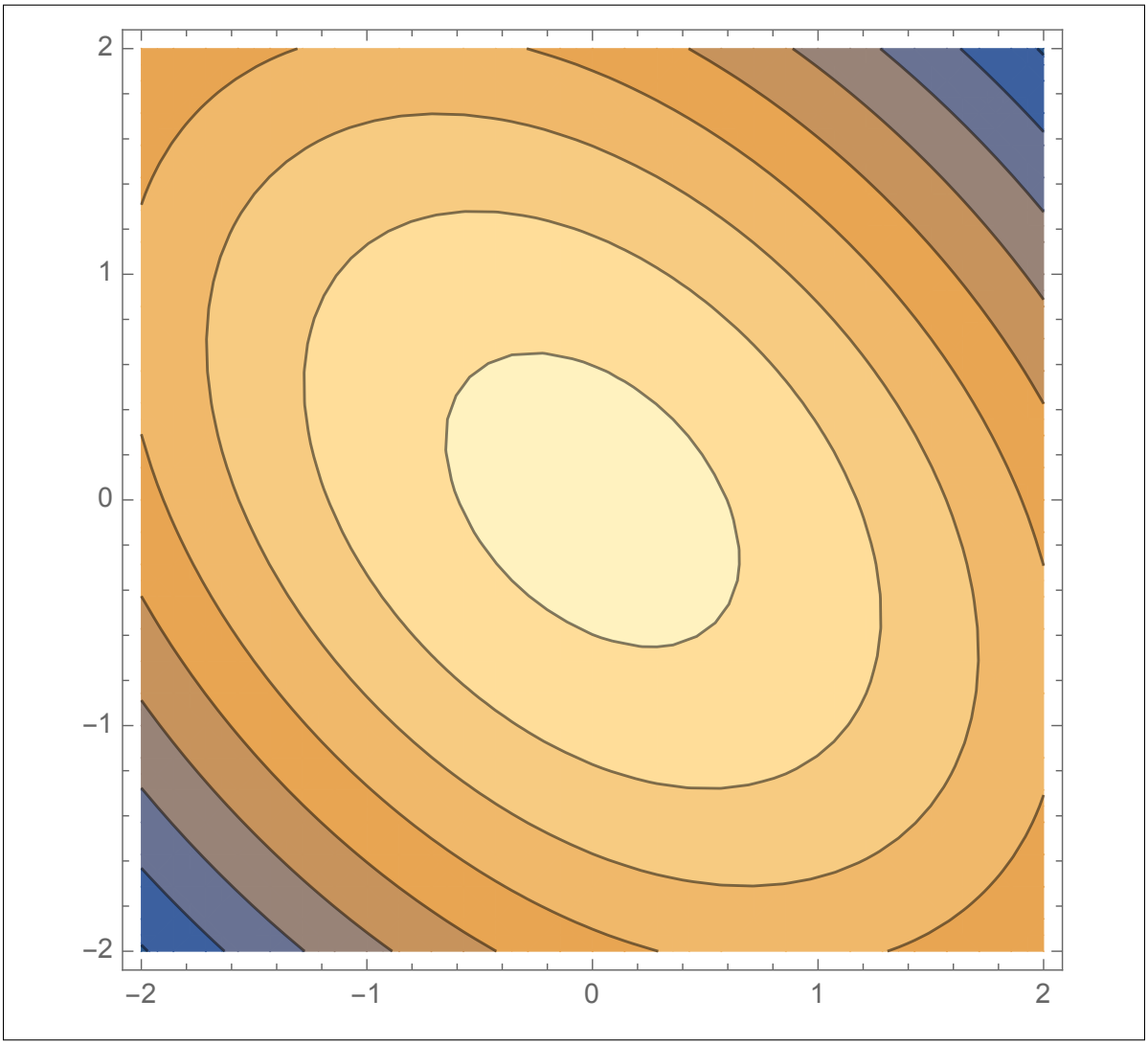
where $C > 0$ is a constant. This is a ellipse in $\mathbb{R}^2$ centered at μ. Specifically, the contours look like those:



5. What can we say about the distribution of X if Σ is $\begin{bmatrix} 10 & -4 \\ -4 & 10 \end{bmatrix}$? Approximately what shape do equiprobability contours of this distribution have?

Solution: This is a 2D normally distributed random vector. The two components are (negatively) correlated.

Similarly, let $p_X(\boldsymbol{x}) = c$, we will get an equation of an ellipse. The contours for this specific case look like those:



6. Is $\begin{bmatrix} 2 & 10 \\ 10 & 2 \end{bmatrix}$ a valid Σ? How can you tell?

Solution: No, because it is not positive semi-definite.

7. If $\mu = (1, 2)$, and $\Sigma = \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix}$, what is the conditional pdf $p_{X_1|X_2}(x_1 \mid 3)$?

Solution: Note there is no correlation between X_1 and X_2, so they are independent. In this case, conditioning on X_2 does not change the distribution of X_1, which is a normally distributed random variable with mean 1 and variance 10.

1.3 Probability

1. Let A, B be $p \times q$ matrices and x be a random $q \times 1$ vector. Prove that

$$\text{cov}(Ax, Bx) = A\,\text{cov}(x)B^T$$

where $\text{cov}(u, v) = E\left[(u - E[u])(v - E[v])^T\right]$ is the cross-covariance matrix between random vectors u and v, while $\text{cov}(u) = E\left[(u - E[u])(u - E[u])^T\right]$ is the covariance matrix for u.

Solution:

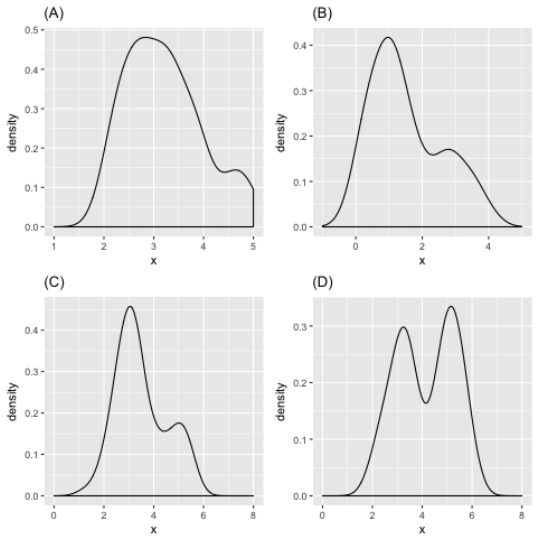
$$\begin{aligned}
\text{cov}(Ax, Bx) &= E\left[(Ax - E[Ax])(Bx - E[Bx])^T\right] \\
&= E\left[A(x - E[x])(B(x - E[x]))^T\right] \\
&= E\left[A(x - E[x])(x - E[x])^T B^T\right] \\
&= AE\left[(x - E[x])(x - E[x])^T\right] B^T \\
&= A \cdot \text{cov}(x) \cdot B^T.
\end{aligned}$$

2. You go for your annual checkup and have several lab tests performed. A week later your doctor calls you and says she has good and bad news. The bad news is that you tested positive for a marker of a serious disease, and that the test is 97% accurate (i.e. the probability of testing positive given that you have the disease is 0.97, as is the probability of testing negative given that you don't have the disease). The good news is that this is a rare disease, striking only 1 in 20,000 people. What are the chances that you actually have the disease?

Solution: Let A be the event of you having the disease and B be the event of testing positive. By Bayes rule we have:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B|A)P(A) + P(B|\neg A)P(\neg A)} = \frac{0.97 \cdot 0.00005}{0.97 \cdot 0.00005 + 0.03 \cdot 0.99995} \approx 0.0016$$

3. Consider the following generative process describing the joint distribution $p(Z, X)$: $Z \sim$ Bernoulli(0.2), $X \mid Z = 0 \sim$ Gaussian$(\mu_0, 0.5)$ and $X \mid Z = 1 \sim$ Gaussian$(\mu_1, 0.5)$, where $\mu_0 = 3$ and $\mu_1 = 5$. Which of the following plots is the marginal distribution $p(X)$?



Solution: The marginal distribution of X is given by:

$$p(X) = p(Z = 0)p(X|Z = 0) + p(Z = 1)p(X|Z = 1) = 0.8 \cdot \mathcal{N}(3, 0.5) + 0.2 \cdot \mathcal{N}(5, 0.5).$$

Hence, the solution is (C).

4. Alice and Bob were driving through a tunnel while listening to the Billion-dollar lottery drawing live on the radio. Due to the weak signal, they couldn't hear the last number perfectly clearly. Alice and Bob think the number was A and B, respectively. Is A independent of B? Is A independent of B given the true lottery number T?

Solution: A is not independent of B. Because for example, the chance of $B = 30$ is very high if $A = 13$; much higher than if we knew nothing about A. In other words, $P(B)! = P(B|A)$. By similar reasoning, $P(B|T) = P(B|T, A)$, and therefore A is independent of B given the true lottery number.

1.4 Multivariate calculus

1. Find the minimum value of the function $f(x, y) = x^2 + 2y^2 - xy + x - 4y$ over $\mathbb{R}^2$.

Solution: First observe that f grows to $+\infty$ as $|x| \to \infty$ and $|y| \to \infty$, so f must attain a minimum value at some point in $\mathbb{R}^2$. This global minimum will also be a local minimum, so it can be found using the first derivative test.

To prove the first statement, we can write f as:

$$\begin{aligned} f(x, y) &= x^2 + 2y^2 - xy + x - 4y \\ &= \left(\frac{x}{2} - y\right)^2 + \frac{3}{4}x^2 + y^2 + x - 4y \\ &\geq \frac{3}{4}x^2 + y^2 + x - 4y \end{aligned}$$

Since the quadratic terms have strictly positive coefficients, the function goes to $+\infty$ in all directions. Now, computing partial derivatives:

$$\begin{aligned} \frac{\partial f}{\partial x}(x, y) &= 2x - y + 1 \\ \frac{\partial f}{\partial y}(x, y) &= 4y - x - 4 \end{aligned}$$

Setting partial derivatives to 0,

$$\begin{aligned} 2x_0 - y_0 + 1 &= 0 \\ 4y_0 - x_0 - 4 &= 0 \end{aligned}$$

Solving these equations, we get $x_0 = 0$, $y_0 = 1$.

So far, we only know that this is a zero derivative point, it could be either a local maximum, or local minimum, or a saddle point. Normally we would compute the Hessian to resolve this. But since we argued earlier that the function must have a global minimum, and there is only one zero derivative point, the local minimum has to be the global minimum. Therefore the global minimum value of f is $f(x_0, y_0) = -2$.

2. Show that $f(x, y)$ is convex over $\mathbb{R}^2$ by showing that its Hessian is positive semi-definite.

Solution: Since the function is twice differentiable, it is convex if and only if its Hessian matrix is positive semi-definite. We therefore compute the Hessian matrix, which is defined as:

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}$$

Computing,

$$\begin{aligned} \frac{\partial f}{\partial x}(x,y) &= 2x - y + 1 \\ \frac{\partial f}{\partial y}(x,y) &= 4y - x - 4 \\ \frac{\partial^2 f}{\partial x^2} &= 2 \\ \frac{\partial^2 f}{\partial x \partial y} &= -1 \\ \frac{\partial^2 f}{\partial y^2} &= 4 \end{aligned}$$

Hence,

$$H = \begin{bmatrix} 2 & -1 \\ -1 & 4 \end{bmatrix}$$

It is easy to see that H is positive definite since it is diagonally dominant. Alternatively, we can compute its eigenvalues, λ_1 and λ_2:

$$\begin{aligned} \lambda_1 + \lambda_2 &= \text{trace}(H) = 6 \\ \lambda_1 \cdot \lambda_2 &= \det(H) = 7 \end{aligned}$$

Solving, $\lambda_1 = 3 + \sqrt{2}$ and $\lambda_2 = 3 - \sqrt{2}$. Since these are both positive, H is positive definite. Hence f is convex over $\mathbb{R}^2$.

Note: Here's a handy tool for quickly (sanity) check your matrix calculus `http://matrixcalculus.org/`.

1.5 Optimization and Gradient Descent

Grady Ent decides to train a single sigmoid unit using the following objective function:

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_i (\sigma(\boldsymbol{x}^{(i)} \cdot \boldsymbol{w}) - y^{(i)})^2 + \frac{\beta}{2}\sum_j w_j^2$$

where $\sigma(z) = 1/(1+e^{-z})$ is the sigmoid function. Note that $\boldsymbol{x}^{(i)} \cdot \boldsymbol{w}$ is the inner product between the vectors $\boldsymbol{x}^{(i)}$ and $\boldsymbol{w}$, where $(\boldsymbol{x}^{(i)}, y^{(i)})$ is the i-th training data point.

(a) Write an expression for $\partial E/\partial w_j$.

Solution:

$$\begin{aligned}\frac{\partial E}{\partial w_j} &= \frac{\partial}{\partial w_j}\Big(\frac{1}{2}\sum_i (\sigma(\boldsymbol{x}^{(i)} \cdot \boldsymbol{w}) - y^{(i)})^2\Big) + \frac{\partial}{\partial w_j}\Big(\frac{1}{2}\beta\sum_j w_j^2\Big) \\ &= \sum_i \frac{\partial \ell^i}{\partial \hat{y}^{(i)}}\frac{\partial \hat{y}^{(i)}}{\partial z^{(i)}}\frac{\partial z^{(i)}}{\partial w_j} + \beta w_j \\ &= \sum_i (\hat{y}^{(i)} - y^{(i)})(\hat{y}^{(i)})(1-\hat{y}^{(i)})(x_j^{(i)}) + \beta w_j\end{aligned}$$

Where we use $\hat{y}^{(i)} := \sigma(\boldsymbol{x}^{(i)} \cdot \boldsymbol{w})$ to represent the predictions, $\ell^i := \frac{1}{2}(\hat{y}^i - y^i)^2$ as shorthand for the squared error on an individual point, and $z^{(i)} := \boldsymbol{x}^{(i)} \cdot \boldsymbol{w}$.

(b) Give the gradient descent update to weight w_j given a single training example $(\boldsymbol{x}, y)$. Your answer should be in terms of the training data and a learning rate.

Solution:

$$w_j := w_j - \alpha((\sigma(\boldsymbol{x} \cdot \boldsymbol{w}) - y)\sigma(\boldsymbol{x} \cdot \boldsymbol{w})(1 - \sigma(\boldsymbol{x} \cdot \boldsymbol{w}))x_j + \beta w_j),$$

where α is the learning rate.

(c) Is the following claim true? "Stochastic gradient descent steps always decrease the objective". Please provide a brief justification for your answer.

Solution: False. Stochastic gradient descent only provides an approximation of the gradient, as it does not use all training samples, and any individual sample may cause a step away from the optimum.

(d) Is the following claim true? "There are circumstances in which stochastic gradient descent is to be preferred to exact gradient descent". Please provide a brief justification for your answer.

Solution: True. (1) It may help avoid local minima, or even undesirable global minima. (2) It can sometimes improve computational efficiency.

2 Programming problems: Just for practice — do not turn in!

Do not use `for` loops in any of these solutions! (Some of these are tricky, but cool when you see it!).

2.1 Regularization

Given an $n \times n$ matrix C, add a scalar a to each diagonal entry of C.

Solution:

```python
import numpy as np

# Feel free to change the parameters
# n, C, and a below

n = 3
C = np.random.randn(n, n)
a = np.random.randn(1)
# Solution 1:
def regularization1(C, a):
    n = np.shape(C)[0]
    return C + a * np.eye(n)


# Solution 2:
def regularization2(C, a):
    n = np.shape(C)[0]
    idx = np.ravel_multi_index((range(n), range(n)), C.shape)
    C.flat[idx] += a
    return C


# While a little bit more complicated,
# solution 2 is much more efficient than solution 1 when $n$ is large.

print(regularization1(C, a))
print(regularization2(C, a))
```

2.2 Largest Off-diagonal Element

Given an $n \times n$ matrix A, find the value of the largest off-diagonal element.

Solution:

```
import numpy as np

# Feel free to change the parameters
# n and A below
n = 3
A = np.random.randn(n, n)

# Creating a mask for the off-diagonal indices
msk = np.eye(n) == 0
v = np.max(A[msk])
```

2.3 Pairwise Computation

Given a vector x of length m, and a vector y of length n, compute $m \times n$ matrices: A and B, such that $A(i,j) = x(i) + y(j)$, and $B(i,j) = x(i) \cdot y(j)$.

Solution:

```
import numpy as np

# feel free to change the parameters
# m, n, x, and y below

m = 3
n = 4
x = np.random.randn(m)
y = np.random.randn(n)
A = x[:, np.newaxis] + y[np.newaxis, :]
B = x[:, np.newaxis] * y[np.newaxis, :]
```

2.4 Pairwise Euclidean Distances

Given a $d \times m$ matrix X, and a $d \times n$ matrix Y, compute an $m \times n$ matrix D, such that $D(i,j) = \|x^i - y^j\|$, where x^i is the i-th column of X, and y^j is the j-th column of Y. Hint: You may

find the following decomposition (of the norm/distance squared) helpful for improving your code efficiency:

$$\left\|x^i - y^j\right\|^2 = \sum_{k=1}^{d} \left(x_k^i - y_k^j\right)^2 = \sum_{k=1}^{d} x_k^{i^2} + \sum_{k=1}^{d} y_k^{j^2} - \sum_{k=1}^{d} 2x_k^i y_k^j.$$

Solution:

```python
import numpy as np

# Feel free to change the dimension parameters
# d, m, n, X, and Y below
d = 3
m = 4
n = 5

X = np.random.randn(d, m)
Y = np.random.randn(d, n)
# Solution 1: two-fold for-loop

D = np.zeros((m, n))
for j in range(n):
    for i in range(m):
        D[i, j] = np.linalg.norm(X[:, i] - Y[:, j])


# Solution 2: one-fold for-loop
D = np.zeros((m, n))
for j in range(n):
    Z = (X.T - Y[:, j]).T
    D[:, j] = sum(Z**2)
D = np.sqrt(D)

# Solution 3: no for-loop (the best solution).
# Using the hint. There are three terms, each can be computed
# for all m  n pairs in batch without for-loop.

tx2 = np.sum(X**2, 0)
ty2 = np.sum(Y**2, 0)
Txy = np.dot(X.T, Y)
```

```
D = tx2[:, np.newaxis] + ty2[np.newaxis, :] - 2 * Txy
D = np.sqrt(D)

# Simple benchmark shows that solution 3 is 10x to 30x
# faster than solution 1 for moderate size problems.
```

2.5 Compute Mahalanobis Distances

The Mahalanobis distance is a measure of the distance between a point P and a distribution D, introduced by P. C. Mahalanobis in 1936. It is a multi-dimensional generalization of the idea of measuring how many standard deviations away P is from the mean of D. This distance is zero if P is at the mean of D, and grows as P moves away from the mean: Along each principal component axis, it measures the number of standard deviations from P to the mean of D. If each of these axes is rescaled to have unit variance, then Mahalanobis distance corresponds to standard Euclidean distance in the transformed space. Mahalanobis distance is thus unitless and scale-invariant, and takes into account the correlations of the data set (from `http://en.wikipedia.org/wiki/Mahalanobis_distance`). Given a center vector c, a positive-definite covariance matrix S, and a set of n vectors as columns in matrix X, compute the distances of each column in X to c, using the following formula:

$$D(i) = \left(x^i - c\right)^T S^{-1} \left(x^i - c\right). \tag{2}$$

Here, D is a vector of length n.

Solution:

```
import numpy as np

# Feel free to change the parameters
# n, c, S, and X below

n = 4
c = np.random.randn(n)
# we need a PD covariance matrix S,
# here's one (common) way of creating a random PD matrix
_ = np.random.randn(n, n)
S = (_) @ (_.T) + 1e-3 * np.eye(n)
X = np.random.randn(n, n)
```

```python
# Solution 1: naive solution as baseline
# Not good: inversing S for n times.
D = np.zeros(n)
for i in range(n):
    z = X[:, i] - c
    D[i] = np.dot(np.dot(z.T, np.linalg.inv(S)), z)

# Solution 2: do pre-computation
D = np.zeros(n)
invS = np.linalg.inv(S)
for i in range(n):
    z = X[:, i] - c
    D[i] = np.dot(np.dot(z.T, invS), z)

# Solution 3: vectorization
Z = X - c[:, np.newaxis]
invS = np.linalg.inv(S)
D = np.sum(Z.conj() * (np.dot(invS, Z)), axis=0)

# Solution 4: directly solving linear equations is
# more efcient than doing inverse.
Z = X - c[:, np.newaxis]
D = np.sum(Z.conj() * (np.linalg.solve(S, Z)), axis=0)
```

2.6 2-D Gaussian

Generate 1000 random points from a 2-D Gaussian distribution with mean $\mu = [4, 2]$ and covariance

$$\Sigma = \begin{pmatrix} 1 & 1.5 \\ 1.5 & 3 \end{pmatrix} \tag{3}$$

Plot the points so obtained, and estimate their mean and covariance from the data. Find the eigenvectors of the covariance matrix and plot them centered at the sample mean.

Solution:

```python
import matplotlib.pyplot as plt
import numpy as np
```

```
N = 1000
mu = [4, 2]
sigma = [[1, 1.5], [1.5, 3]]


R = np.random.multivariate_normal(mu, sigma, N)
plt.plot(*(zip(*R)), marker=".", ls="")
plt.show()
muhat = np.mean(R, 0)
R_0 = R - muhat[np.newaxis, :]
# Why is the denominator (N-1) instead of (N)? Check
# out Bessel's correction
SIGMA_hat = np.dot(R_0.T, R_0) / (N - 1)

L, Q = np.linalg.eig(SIGMA_hat)

plt.arrow(
    muhat[0],
    muhat[1],
    Q[0, 0],
    Q[1, 0],
    shape="full",
    lw=3,
    length_includes_head=True,
    head_width=0.01,
)
plt.arrow(
    muhat[0],
    muhat[1],
    Q[0, 1],
    Q[1, 1],
    shape="full",
    lw=3,
    length_includes_head=True,
    head_width=0.01,
)

plt.plot(*(zip(*R)), marker=".", ls="")
plt.axis([1, 6, -2, 6])
```

```
plt.show()
```

2.7 Tournament fun

A tennis tournament starts with sixteen players. Let's call them $h_i, i = 1, 2, \ldots 16$ (human i, to avoid the potentially confusing notation p_i). The first round has eight games, randomly drawn/paired; i.e., every player has an equal chance of facing any other player. The eight winners enter the next round.

As an enthusiastic tennis and data fan, you have an internal model of these 16 players based on their past performance. In particular, you view each player h_i as having a performance index score $s_i \sim \text{Gaussian}(\theta_i, \sigma_i^2)$. The mean θ_i roughly captures the player's 'intrinsic ability' and the variance σ_i^2 roughly captures the player's performance reliability (accounting for recent injuries etc.). In a match between h_i and h_j, player h_i wins if $s_i > s_j$.

Based on your model, what's the probability that your "top seed player" (the one with the highest θ) enters the next round? Run 10,000 simulations to check if it agrees with your answer.

Solution: Suppose the top seed player is h_1. For h_1 to win against an opponent h_j, we need the event $s_j - s_1 < 0$. Since s_i and s_j are independent normal distributions, their difference is also a normal distribution, with mean $\theta_j - \theta_1$ and variance $\sigma_j^2 + \sigma_1^2$.

Because h_1 has a $1/15$ chance of facing any one (and only one) other player, we have 15 such disjoint events. So the total probability of h_1 entering the next game is simply:

$$\frac{1}{15} \sum_{j=2}^{15} p(s_j - s_1 < 0)$$

where $(s_j - s_1) \sim \text{Gaussian}(\theta_j - \theta_1, \sigma_j^2 + \sigma_1^2)$. Solution code below.

```
import numpy as np
from scipy.stats import norm

# Feel free to change the parameters below
theta = np.linspace(3, 16, 16)
sigma = np.linspace(1, 2, 16)

# We'll start by getting our CDF solution

# get our top seed player's parameters
top_seed_index = np.argmax(theta)
top_seed_theta = theta[top_seed_index]
```

```
top_seed_sigma = sigma[top_seed_index]
# get the fifteen 'difference' random variable parameters
all_other_theta = np.delete(theta, top_seed_index)
all_other_sigma = np.delete(sigma, top_seed_index)
x_mean = all_other_theta - top_seed_theta
x_var = top_seed_sigma**2 + all_other_sigma**2
# x is the array holding the prob. of top seed player
# winning against each of the 15 opponents
x = [norm.cdf(0, loc=i, scale=np.sqrt(j)) for (i, j) in zip(x_mean, x_var)]
ans = np.sum(x) / 15
print(f"Top seed player's chance of winning is {ans}")

# We then run some simulations to see if the proportion of
# wins agree with the solution above
M = int(1e5)
count = 0


def one_simulation(all_other_theta, all_other_sigma):
    # choose a random opponent index
    j = np.random.choice(range(15))
    sj = norm.rvs(all_other_theta[j], all_other_sigma[j])
    top_seed_s = norm.rvs(top_seed_theta, top_seed_sigma)
    if top_seed_s < sj:
        return False
    return True


for i in range(M):
    if one_simulation(all_other_theta, all_other_sigma):
        count += 1
print(f"Top seed player wins {count/M} of the total simulated games.")
```