Section 02: Solutions

In this section, we go over data standardization and normalization, explore maximum likelihood estimation with more examples of noise densities, review train/val/test split, and also review some basics about subspaces in linear algebra.

Definitions, again!

Norms
We haven't covered norms in much detail, but they are instrumental and show up quite often! We will cover them later in more detail, but for now, it is sufficient to get yourself familiar with definitions for some of the most widely used norms! For any vector v that is n-dimensional, i.e. $v \in \mathbb{R}^n$, we have the following

(a) **One-norm (ℓ_1)**: $||v||_1 = \sum_{i=1}^{n} |v_i|$

(b) **Two-norm (ℓ_2)**: $||v||_2 = \sqrt{v^T v} = \sqrt{\sum_{i=1}^{n} v_i^2}$

(c) **∞-norm**: $||v||_\infty = \max_i |v_i|$

Symmetric Matrices and the Quadratic Form
Let us define a matrix $A \in \mathbb{R}^{n \times n}$.

(a) We have that the matrix A is symmetric iff $A = A^T$

(b) The quadratic form is defined to be $x^T Ax$ for any vector $x \in \mathbb{R}^n$. The matrix A is said to be positive semi-definite if $x^T Ax \geq 0$

Closure of the Normal and Laplacian under scale and shift

(a) **Normal:** If $X \sim \mathcal{N}(\mu_x, \sigma_x^2)$, then we have that $Y = aX + b \sim \mathcal{N}(a\mu_x + b, a^2\sigma_x^2)$.

(b) **Laplacian:** If $X \sim \text{Laplace}(\mu, b)$ then $kX + c \sim \text{Laplace}(k\mu + c, |k|b)$

1. Data Normalization/Standardization

Sometimes, our features have very different ranges of values. This is not ideal and can lead to numerical issues (e.g., overflow) and optimization difficulties.

There are two ways to take care of this issue. One is called data normalization, and the other is called data standardization. Sometimes these terms are used interchangeably, but it is important to understand the difference.

Below, $x_i^{(j)}$ represents the value of the i^{th} feature of the j^{th} data point.

1.1. Data Standardization

Data standardization is the task of transforming each feature in our dataset to have mean 0 and variance 1. The typical way to do this is using the Z-Score, which is defined as below:

$$\tilde{x_i}^{(j)} = \frac{x_i^{(j)} - \mu_i}{\sigma_i}$$

Where μ_i is the mean of each feature and σ_i is the standard deviation of each feature, which are empirically calculated from the data.

Question: what should you do when $\sigma_i = 0$ for some i?

Solution:

Having $\sigma_i = 0$ for some feature means that the value for that feature is constant in our dataset. If we leave it as 0, we will encounter a divide by 0 error. Since the feature is constant, once we subtract the mean, the new value for the feature will be 0, so we can divide by anything except 0 to avoid this error.

Having $\sigma_i = 0$ is rare, and may be a sign something is wrong with your data or code. One specific case to watch out for is appending your bias column of ones before standardizing.

1.2. Data Normalization

Data normalization refers to the task of rescaling each feature in our dataset to have range [0, 1].

One such method to achieve this is min-max scaling:

$$\tilde{x_i}^{(j)} = \frac{x_i^{(j)} - x_i^{min}}{x_i^{max} - x_i^{min}}$$

Where x_i^{min}, x_i^{max} are the minimum and maximum values of feature i in our dataset, respectively.

When training and evaluating your model, you should calculate the parameters for your normalization or standardization function on the training set **ONLY**!

In other words, if we were using Z-Score, we'd calculate our μ_i, σ_i on the training set, and use those same values when standardizing our validation/test data. The same applies to normalization methods, such as min-max scaling, where we want to determine x_i^{min}, x_i^{max} from training data. This will likely mean that we may get some values outside [0, 1] after rescaling new data, but, given that our training dataset is large enough, this shouldn't be much of an issue (so unseen data likely won't be wildly outside of [0, 1]).

Question 1: Should we always choose x_i^{min} and x_i^{max} based on train data? Can we sometimes do better? Think about cases when we have some underlying information about data.

Solution:

Consider RGB images. These are typically encoded as arrays of shape (3, height, width), with each value being an integer in range [0, 255]. In this case we should just use $x_i^{max} = 255$ and $x_i^{min} = 0$ to normalize the data.

In general there can be many cases in which we will know max and/or min values of distribution. **Always examine and visualize data** before transforming it.

Question 2: When can values outside of [0, 1] range in test set cause issues?

Solution:

This might lead to an issue if our model performs any transformations on data that have a limited domain. Consider a model $f(x) = \log(x)^T w$. In this case if test datapoint have a value below 0, this code will fail, as log has domain $[0, \infty)$.

In general, after you visualize the data, think about what transforms are needed for it to be well behaved. Always pay attention to domains and ranges of each transform since these may lead to NaNs.

2. Biased Test Error

Is the test error unbiased for these programs? If not, how can we fix the code so it is?

2.1. Program 1

```
1  # Given dataset of 1000-by-50 feature
2  # matrix X, and 1000-by-1 labels vector
3
4  mu = np.mean(X, axis=0)
5  X = X - mu
6
7  idx = np.random.permutation(1000)
8  TRAIN = idx[0:900]
9  TEST = idx[900::]
10
11 ytrain = y[TRAIN]
12 Xtrain = X[TRAIN, :]
13
14 # solve for argmin_w ||Xtrain*w - ytrain||_2
15 w = np.linalg.solve(np.dot(Xtrain.T, Xtrain), np.dot(Xtrain.T, ytrain))
16
17 b = np.mean(ytrain)
18
19 ytest = y[TEST]
20 Xtest = X[TEST, :]
21
22 train_error = np.dot(np.dot(Xtrain, w)+b - ytrain,
23               np.dot(Xtrain, w)+b - ytrain ) / len(TRAIN)
24 test_error = np.dot(np.dot(Xtest, w)+b - ytest,
25               np.dot(Xtest, w)+b - ytest ) / len(TEST)
26
27 print('Train error = ', train_error)
28 print('Test error = ', test_error)
```

Solution:

The error is at the beginning of the program on lines 4 and 5. Notice how μ is a function of both the train and test data. By de-meaning the entire dataset before splitting, we are intertwining the train and test data. The correct procedure is:

- Split into train and test

- Compute the mean of the train data, μ_{train}
- De-mean both the train and test data with μ_{train}

2.2. Program 2

```
# We are given: 1) dataset X with n=1000 samples and 50 features and 2) a vector y of length 1000 with labels.
# Consider the following code to train a model, using cross validation to perform hyperparameter tuning.

def fit(Xin, Yin, _lambda):
    w = np.linalg.solve(np.dot(Xin.T, Xin) + _lambda * np.eye(Xin.shape[1]), np.dot(Xin.T, Yin))
    b = np.mean(Yin) - np.dot(w, mu)
    return w, b

def predict(w, b, Xin):
    return np.dot(Xin, w) + b

idx = np.random.permutation(1000)
TRAIN = idx[0:800]
VAL = idx[800:900]
TEST = idx[900::]

ytrain = y[TRAIN]
Xtrain = X[TRAIN, :]
yval = y[VAL]
Xval = X[VAL, :]

# demean data
mu = np.mean(Xtrain, axis=0)
Xtrain = Xtrain - mu
Xval = Xval - mu

# use validation set to pick the best hyper-parameter to use
lambdas = [10 ** -5, 10 ** -4, 10 ** -3, 10 ** -2]
err = np.zeros(len(lambdas))

for idx, _lambda in enumerate(lambdas):
    w, b = fit(Xtrain, ytrain, _lambda)
    yval_hat = predict(w, b, Xval)
    err[idx] = np.mean((yval_hat - yval)**2)

lambda_best = lambdas[np.argmin(err)]

Xtot = np.concatenate((Xtrain, Xval), axis=0)
ytot = np.concatenate((ytrain, yval), axis=0)

w, b = fit(Xtot, ytot, lambda_best)

ytest = y[TEST]
Xtest = X[TEST, :]

# demean data
Xtest = Xtest - mu

ytot_hat = predict(w, b, Xtot)
train_error = np.mean((ytot_hat - ytot) **2)
ytest_hat = predict(w, b, Xtest)
test_error = np.mean((ytest_hat - ytest) **2)

print('Train error = ', train_error)
print('Test error = ', test_error)
```

Solution:

We are adding the validation data back into training (creating `Xtot`), and then retraining the whole model on this data. However, optimal value of λ **depends** on size of the training dataset, so by adding more data we are using sub-optimal value in final `fit` call. In general, models get better the more data you give them, but only add the validation set back in if you are confident the hyperparameter doesn't depend on the number of elements, and that you aren't allowing your model access to the test set. Note: it is not incorrect to add validation data back into training. It's a trade-off between having more training data and having a reliable estimate of test performance. See offering au24 slide 39 for reference.

3. Maximum Likelihood Estimation

In this section, we formulate maximum likelihood estimation for different noise densities as different minimization problems. Specifically, we'll see how each noise distribution corresponds to a specific objective function.

We consider the linear measurement model (parameterized by w), $y_i = x_i^\top w + v_i$ for $i = 1, 2, \ldots, m$. The noise v_i for different measurements (x_i, y_i) are all independent and identically distributed. Under our assumption of a linear model, $v_i = y_i - x_i^\top w$. Note Per the principle of maximum likelihood estimation, we seek to maximize

$$\log p_w((x_1, y_1), \cdots, (x_m, y_m)) = \log \prod_{i=1}^{m} p(y_i - x_i^\top w).$$

(a) Show that when the noise measurements follow a Gaussian distribution ($v_i \sim \mathcal{N}(0, \sigma^2)$), the maximum likelihood estimate of w is the solution to $\min_w \|Xw - Y\|_2^2$. Here each row in X corresponds to a x_i, and each row in Y to y_i.

Solution:

When $v_i \sim \mathcal{N}(0, \sigma^2)$, the density is given by the expression $p(v) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-v^2/2\sigma^2}$. This implies that the MLE of parameter is

$$\begin{aligned}
\hat{w}_{MLE} &= \arg\max_w \log p_w((x_1, y_1), \cdots, (x_m, y_m)) \\
&= \arg\max_w \log \prod_{i=1}^{m} p(y_i - x_i^\top w) \\
&= \arg\max_w \sum_{i=1}^{m} \log p(y_i - x_i^\top w) \quad [\log(ab) = \log a + \log b] \\
&= \arg\max_w \sum_{i=1}^{m} \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y_i - x_i^\top w)^2/2\sigma^2} \right] \\
&= \arg\max_w m \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right) + \sum_{i=1}^{m} -\frac{(y_i - x_i^\top w)^2}{2\sigma^2} \\
&= \arg\max_w \sum_{i=1}^{m} -\frac{1}{2\sigma^2} (y_i - x_i^\top w)^2 \quad \text{(constant offset doesn't affect results)} \\
&= \arg\max_w \sum_{i=1}^{m} -(y_i - x_i^\top w)^2 \quad \text{(positive scalar scaling doesn't affect results)} \\
&= \arg\min_w \sum_{i=1}^{m} (y_i - x_i^\top w)^2 = \arg\min_w \|Xw - Y\|_2^2
\end{aligned}$$

Therefore, the maximum likelihood estimate is given by $\arg\min \|Xw - Y\|_2^2$, as claimed.

(b) When the noise measurements follow a Laplacian distribution ($p(z) = (1/2a) \exp(-|z|/a)$), what is the maximum likelihood estimate of w? Express your answer as the solution to an optimization problem such as in the

previous part.

Solution:

For $a > 0$, with density $p(z) = (1/2a)\exp(-|z|/a)$, we follow a similar procedure as in part (a)

$$\begin{aligned}
\hat{w}_{MLE} &= \arg\max_w \log p_w((x_1, y_1), \cdots, (x_m, y_m)) \\
&= \arg\max_w \log \prod_{i=1}^{m} p(y_i - x_i^\top w) \\
&= \arg\max_w \sum_{i=1}^{m} \log p(y_i - x_i^\top w) \quad [\log(ab) = \log a + \log b] \\
&= \arg\max_w \sum_{i=1}^{m} \log\left(\frac{1}{2a} \cdot e^{\frac{-|y_i - x_i^T w|}{a}}\right) \\
&= \arg\max_w \sum_{i=1}^{m} \log\left(\frac{1}{2a}\right) - \frac{|y_i - x_i^T w|}{a} \\
&= \arg\max_w \sum_{i=1}^{m} -\frac{|y_i - x_i^T w|}{a} && \text{constant offset doesn't affect optimization} \\
&= \frac{1}{a} \cdot \arg\max_w \sum_{i=1}^{m} -|y_i - x_i^T w| \\
&= \arg\max_w \sum_{i=1}^{m} -|y_i - x_i^T w| && \text{constant offset doesn't affect optimization} \\
&= \arg\min_w \sum_{i=1}^{m} |y_i - x_i^T w| = \arg\min_w \|Xw - Y\|_1
\end{aligned}$$

Therefore the maximum likelihood estimate of w is $\widehat{w} = \arg\min_w \|Xw - Y\|_1$.

(c) When the noise measurements follow a uniform distribution ($p(z) = (1/2a)$ on $[-a, a]$), what is the maximum likelihood estimate of w? Express your answer as a condition to be satisfied by some function of w.

Solution:

For uniformly distributed v_i on $[-a, a]$, the density function is $p(z) = \frac{1}{2a}$, we have that

$$\begin{aligned}
\hat{w}_{MLE} &= \arg\max_w \log p_w((x_1, y_1), \cdots, (x_m, y_m)) \\
&= \arg\max_w \log \prod_{i=1}^{m} p(y_i - x_i^\top w) \\
&= \arg\max_w \sum_{i=1}^{m} \log p(y_i - x_i^\top w) \quad [\log(ab) = \log a + \log b]
\end{aligned}$$

Using an indicator function, we have that this can be simplified into

$$\begin{aligned}\hat{w}_{MLE} &= \arg\max_w \sum_{i=1}^{m} \log\left(\frac{1}{2a} \cdot \mathbf{1}\{-a \le y_i - x_i^T w \le a\}\right) \\ &= \arg\max_w \sum_{i=1}^{m} \log\left(\frac{1}{2a}\right) + \log\left(\mathbf{1}\{-a \le y_i - x_i^T w \le a\}\right) \\ &= \arg\max_w \sum_{i=1}^{m} \log\left(\mathbf{1}\{-a \le y_i - x_i^T w \le a\}\right) \qquad \text{Constant factor doesn't affect optimization}\end{aligned}$$

We now simplify this to be

$$\hat{w}_{MLE} = \arg\max \begin{cases} 0 & -a \le y_i - x_i^T w \le a \\ -\infty & \text{otherwise} \end{cases}$$

We further simplify this to

$$\hat{w}_{MLE} = \arg\max \begin{cases} 0 & ||Y - Xw||_\infty \le a \\ -\infty & \text{otherwise} \end{cases}$$

Therefore the maximum likelihood estimate is given by any w that satisfies $|Y - Xw||_\infty \le a$ which is the same as $||Xw - Y||_\infty \le a$.

4. Linear Algebra Review Part 2

Let $X \in \mathbb{R}^{m \times n}$. X may not have full rank. We explore properties about the four fundamental subspaces of X.

Please refer to section 3 of the supplementary material for more info on matrix properties and subspaces.

4.1. Connections between subspaces of X

Check the following facts.

(a) The rowspace of X is the columnspace of $X^\top$, and vice versa.

Solution:

The matrix $X^\top$ is $\begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}$. The rows of X are the columns of $X^\top$, and vice versa.

(b) The nullspace of X and the rowspace of X are orthogonal complements. This can be written in shorthand as $\text{Null}(X) = \text{Range}(X^\top)^\perp$. This is further equivalent to saying $\text{Range}(X^\top) = \text{Null}(X)^\perp$.

Solution:

A vector $v \in \text{Null}(X)$ if and only if $Xv = 0$, which is true if and only if for every row X_i of X, $\langle X_i, v \rangle = 0$. This is precisely the condition that v is perpendicular to each row of X, which is the stated claim.

(c) The nullspace of $X^\top$ is orthogonal to the columnspace of X. This can be written in shorthand as $\text{Null}(X^\top) = \text{Range}(X)^\perp$.

Solution:

This is seen by applying the previous result to $X^\top$.

4.2. Linear algebra facts for linear regression

We saw in lecture on Linear Regression that the closed form expression for linear regression without an offset involves the term $(X^\top X)^{-1}$.

(a) Is it true that the matrix $X^\top X$ is always symmetric and positive semidefinite?

Solution:

Yes. Symmetry can be checked by computing the transpose. For any vector u, we have $u^\top X^\top X u = \|Xu\|_2^2 \geq 0$.

(b) State and prove the connection between the nullspace of X and the nullspace of $X^\top X$. That is, your statement should look like one of the following: $\text{Null}(X) \subseteq \text{Null}(X^\top X)$, or $\text{Null}(X) \supseteq \text{Null}(X^\top X)$ or $\text{Null}(X) = \text{Null}(X^\top X)$.

Solution:

We have, $\text{Null}(X) = \text{Null}(X^\top X)$. Let $v \in \text{Null}(X)$. Then, one can check that $X^\top X v = 0$, leading to $v \in \text{Null}(X^\top X)$, which proves $\text{Null}(X) \subseteq \text{Null}(X^\top X)$. For the other direction, let $0 \neq v \in \text{Null}(X^\top X)$. Then, $0 = v^\top X^\top X v = \|Xv\|_2^2$, which implies $v \in \text{Null}(X)$. Therefore, $\text{Null}(X^\top X) \subseteq \text{Null}(X)$, which finishes the proof.

(c) Is it true that $X^\top X$ is always invertible?

Solution:

No, this isn't always the case. Since $\text{Null}(X) = \text{Null}(X^\top X)$ (see the answer to the previous question), the matrix $X^\top X$ is not invertible if X has a non-empty nullspace.

(d) Based on the above fact about the connection between the nullspaces of X and $X^\top X$ and the expression for linear regression without an offset (that we referred to two problems above), justify the use of "tall skinny" data matrix X as opposed to a "short wide" matrix X.

Solution:

If X is "short and wide", it has a non-empty nullspace. Therefore, $X^\top X$ is not invertible.

(e) The columnspace and rowspace of $X^\top X$ are the same, and are equal to the rowspace of X. (Hint: Use the relationship between nullspace and rowspace.)

Solution:

$X^\top X$ is symmetric, and from the previous parts, we have $\text{Row}(X^\top X) = \text{Col}((X^\top X)^\top) = \text{Col}(X^\top X)$. By previous parts again, we have: $\text{Row}(X^\top X) = \text{Null}(X^\top X)^\perp = \text{Null}(X)^\perp = \text{Row}(X)$.