

Section 02: Train/Test Split, MLE, Data Normalization/-Standardization, Linear Algebra

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

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

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 ?

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.

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

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)
```

2.2. Program 2

```
1 # We are given: 1) dataset X with n=1000 samples and 50 features and 2) a vector y of length 1000 with labels.
2 # Consider the following code to train a model, using cross validation to perform hyperparameter tuning.
3
4 def fit(Xin, Yin, _lambda):
5     w = np.linalg.solve(np.dot(Xin.T, Xin) + _lambda * np.eye(Xin.shape[1]), np.dot(Xin.T, Yin))
6     b = np.mean(Yin) - np.dot(w, mu)
7     return w, b
8
9 def predict(w, b, Xin):
10    return np.dot(Xin, w) + b
11
12 idx = np.random.permutation(1000)
13 TRAIN = idx[0:800]
14 VAL = idx[800:900]
15 TEST = idx[900:]
16
17 ytrain = y[TRAIN]
18 Xtrain = X[TRAIN, :]
19 yval = y[VAL]
20 Xval = X[VAL, :]
21
22 # demean data
23 mu = np.mean(Xtrain, axis=0)
24 Xtrain = Xtrain - mu
25 Xval = Xval - mu
26
27 # use validation set to pick the best hyper-parameter to use
28 lambdas = [10 ** -5, 10 ** -4, 10 ** -3, 10 ** -2]
29 err = np.zeros(len(lambdas))
30
31 for idx, _lambda in enumerate(lambdas):
32     w, b = fit(Xtrain, ytrain, _lambda)
33     yval_hat = predict(w, b, Xval)
34     err[idx] = np.mean((yval_hat - yval)**2)
35
36 lambda_best = lambdas[np.argmin(err)]
37
38 Xtot = np.concatenate((Xtrain, Xval), axis=0)
39 ytot = np.concatenate((ytrain, yval), axis=0)
40
41 w, b = fit(Xtot, ytot, lambda_best)
42
43 ytest = y[TEST]
44 Xtest = X[TEST, :]
45
46 # demean data
47 Xtest = Xtest - mu
48
49 ytot_hat = predict(w, b, Xtot)
50 train_error = np.mean((ytot_hat - ytot) **2)
51 ytest_hat = predict(w, b, Xtest)
52 test_error = np.mean((ytest_hat - ytest) **2)
53
54 print('Train error = ', train_error)
55 print('Test error = ', test_error)
```

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 $i = 1, 2, \dots, m$. 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

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

- (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 .
- (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.
- (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 .

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 row space of X is the column space of X^\top , and vice versa.
- (b) The nullspace of X and the row space 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 $\text{Range}(X^\top) = \text{Null}(X)^\perp$.
- (c) The nullspace of X^\top is orthogonal to the column space of X . This can be written in shorthand as $\text{Null}(X^\top) = \text{Range}(X)^\perp$.

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?
- (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 $\text{Null}(X) = \text{Null}(X^\top X)$.
- (c) Is it true that $X^\top X$ is always invertible?
- (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 .
- (e) The column space and row space of $X^\top X$ are the same, and are equal to the row space of X . (Hint: Use the relationship between nullspace and row space.)