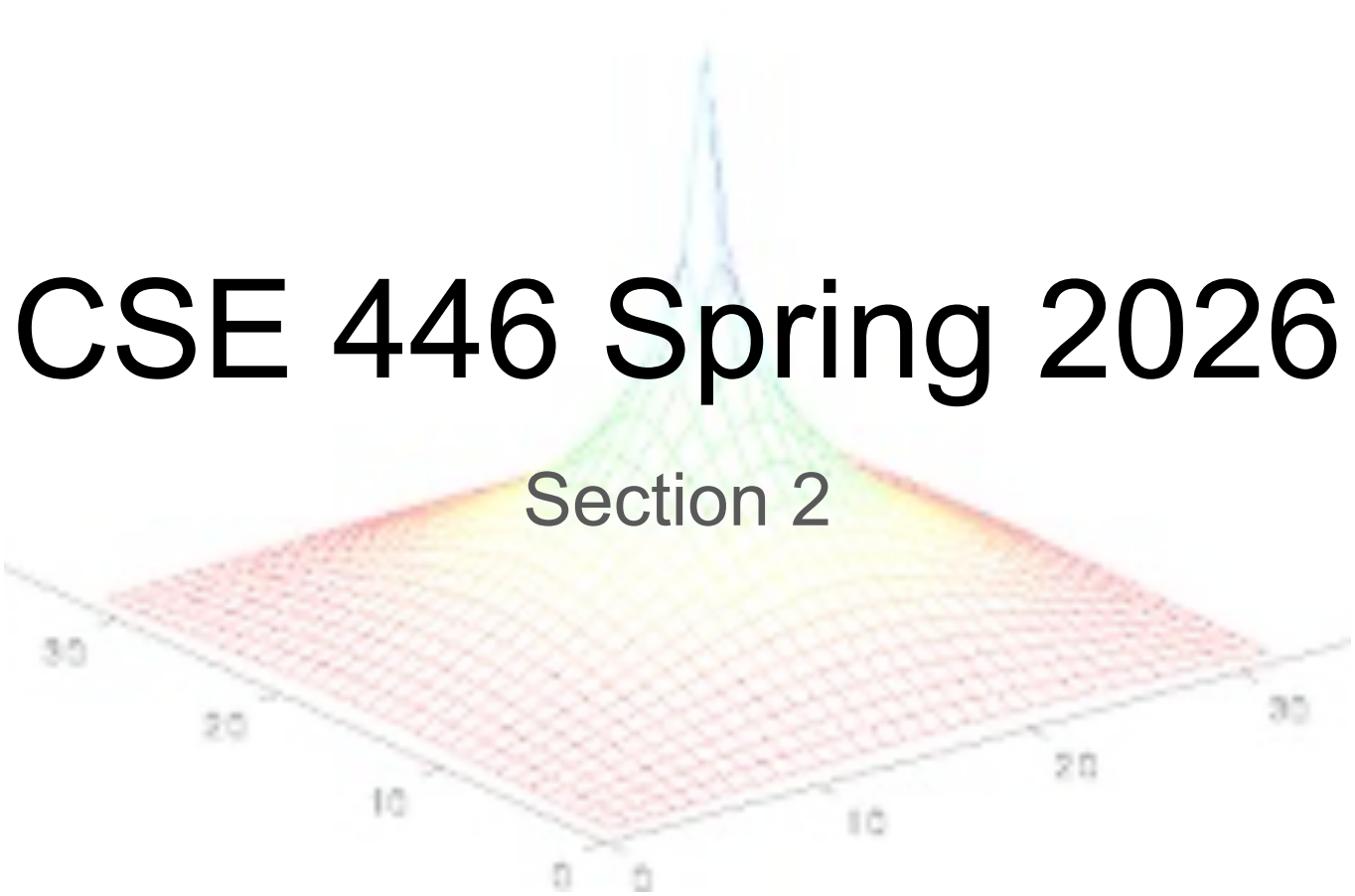
CSE 446 Spring 2026

Section 2



Reminders

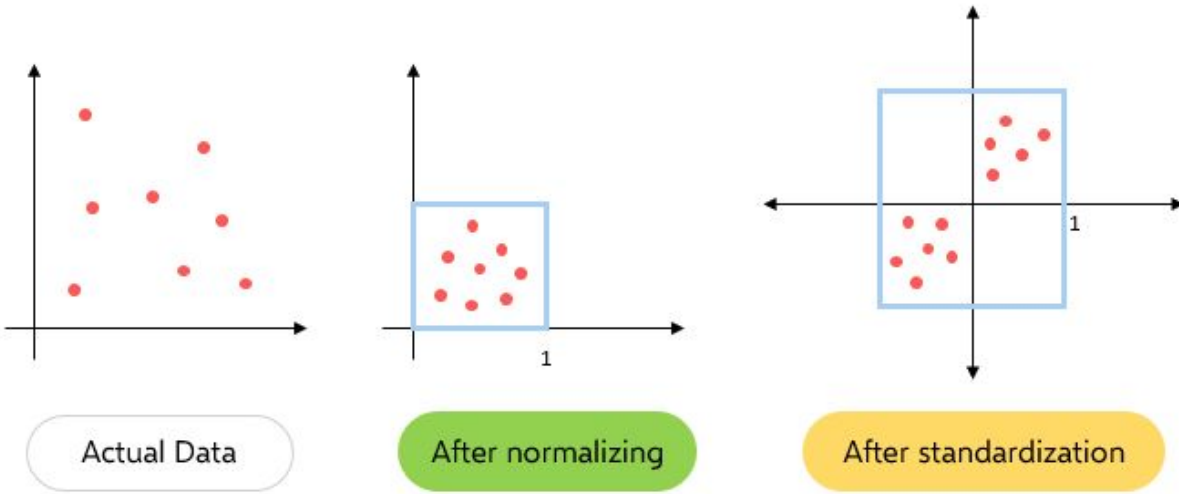
- **HW1** due **Wed, Apr 22**

Some tips:

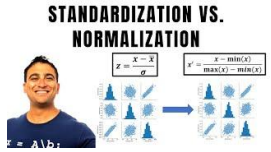
- Use office hours to your advantage
  - Student TA OH for homework questions
  - Professor OH for conceptual questions
- Skim the homeworks the day they are assigned and try one problem
  - Motivates you to get things done on time, starting an untouched assignment can be daunting
- Keep a tab open with the lecture slides while you do the homework for reference

a. Data Normalization/Standardization (1.1/1.2)
b. Train/Val/Test split (2.1)
c. MLE (3.b)
d. Review Linear Regression

Problem 1


1
1
Actual Data
After normalizing
After standardization

Example of Normalization/Standardization



Normalization Vs. Standardization (Feature Scaling in Machine Learning)

Stock Prediction Dataset

RAW ORIGINAL DATASET

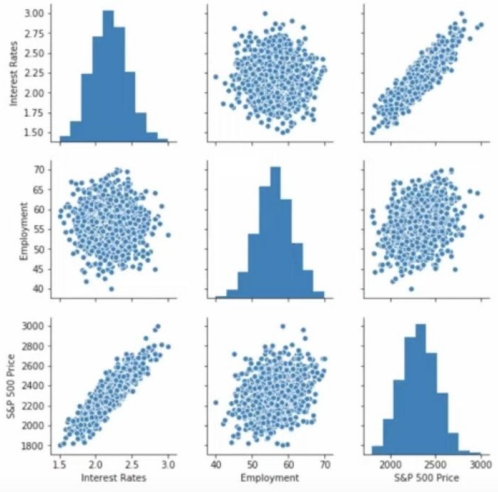
	Interest Rates	Employment	S&P 500 Price
0	1.943859	55.413571	2206.680582
1	2.258229	59.546305	2486.474488
2	2.215863	57.414687	2405.868337
3	1.977960	49.908353	2140.434475
4	2.437723	52.035492	2411.275663
5	2.143637	56.060598	2187.344909
6	2.148647	51.513208	2263.049249
7	2.176184	53.475909	2281.496374
8	2.125352	63.668422	2355.163011
9	2.225682	56.993396	2326.330337

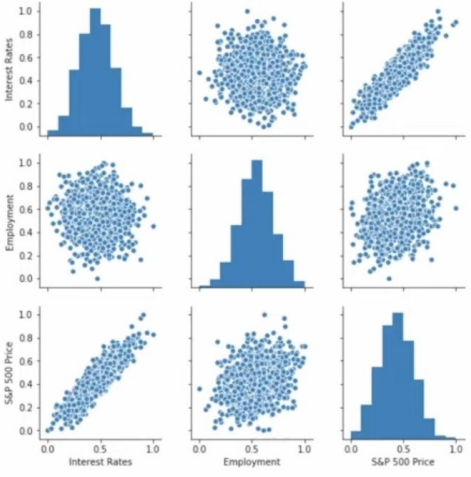
QUICK STATS!

	Interest Rates	Employment	S&P 500 Price
count	1000.00	1000.00	1000.00
mean	2.20	56.25	2320.00
std	0.24	4.86	193.85
min	1.50	40.00	1800.00
25%	2.04	53.03	2190.45
50%	2.20	56.16	2312.44
75%	2.36	59.42	2455.76
max	3.00	70.00	3000.00

Normalization: Range 0 - 1

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



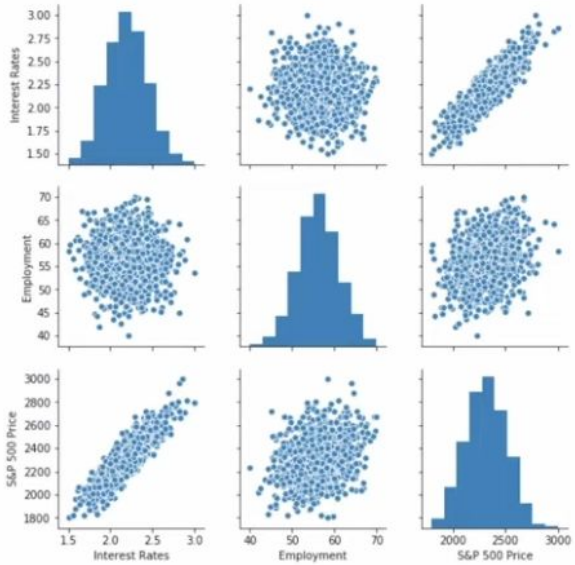


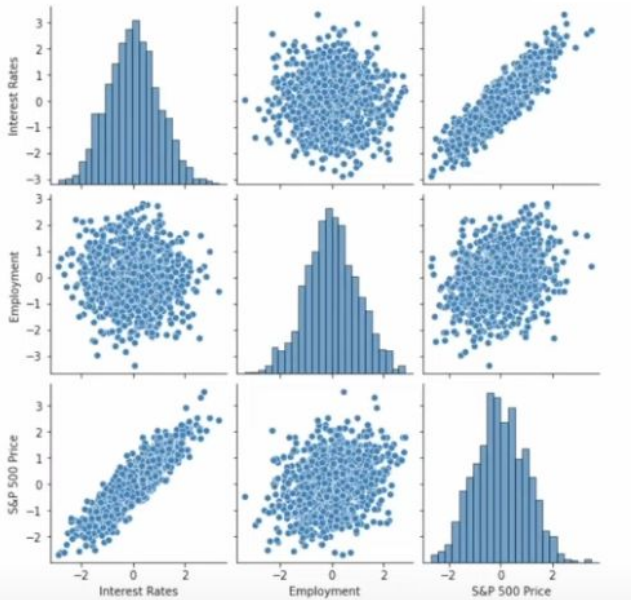
	Interest Rates	Employment	S&P 500 Price
count	1000.00	1000.00	1000.00
mean	2.20	56.25	2320.00
std	0.24	4.86	193.85
min	1.50	40.00	1800.00
25%	2.04	53.03	2190.45
50%	2.20	56.16	2312.44
75%	2.36	59.42	2455.76
max	3.00	70.00	3000.00

	Interest Rates	Employment	S&P 500 Price
count	1000.00	1000.00	1000.00
mean	0.46	0.54	0.43
std	0.16	0.16	0.16
min	0.00	0.00	0.00
25%	0.36	0.43	0.33
50%	0.47	0.54	0.43
75%	0.57	0.65	0.55
max	1.00	1.00	1.00

Standardization: 0 Mean, 1 Standard Deviation

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



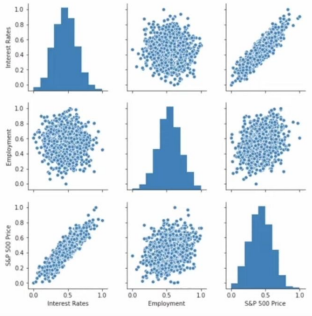


	Interest Rates	Employment	S&P 500 Price
count	1000.00	1000.00	1000.00
mean	2.20	56.25	2320.00
std	0.24	4.86	193.85
min	1.50	40.00	1800.00

	Interest Rates	Employment	S&P 500 Price
count	1000.00	1000.00	1000.00
mean	0.00	0.00	-0.00
std	1.00	1.00	1.00
min	-2.88	-3.34	-2.68

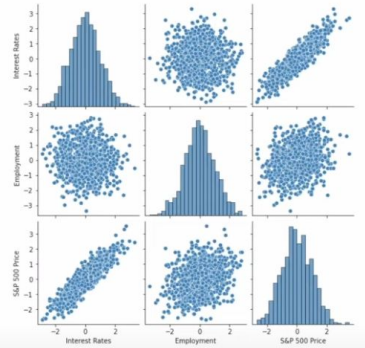
STANDARDIZATION Vs. NORMALIZATION?

- Generally speaking, there is no right or wrong answer!
- In case of neural networks, normalization is preferred since we don't assume any data distribution.
- Standardization is preferred when data follows gaussian distribution
- Standardization is preferred over normalization when there are a lot of outliers.



	Interest Rates	Employment	S&P 500 Price
count	1000.00	1000.00	1000.00
mean	0.46	0.54	0.43
std	0.16	0.16	0.16
min	0.00	0.00	0.00
25%	0.36	0.43	0.33
50%	0.47	0.54	0.43
75%	0.57	0.65	0.55
max	1.00	1.00	1.00

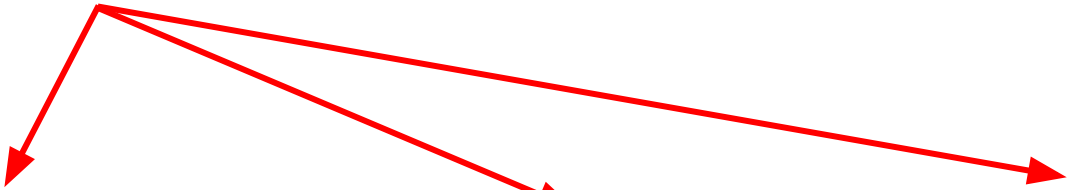
Task/Model Type	Prefer...	Why?
Neural Networks	Normalization	Helps convergence, bounded inputs
KNN, K-Means	Normalization	Distance-based, scale matters
Linear Models (LR, SVM)	Standardization	Assumes normally distributed features
PCA / LDA	Standardization	Sensitive to variance



	Interest Rates	Employment	S&P 500 Price
count	1000.00	1000.00	1000.00
mean	0.00	0.00	-0.00
std	1.00	1.00	1.00
min	-2.88	-3.34	-2.68

What do you never ever ever ever ever ever ever ever ever do?

Tune your model on your test set!



Training parameters

Easy to remember…

Choosing hyperparameters

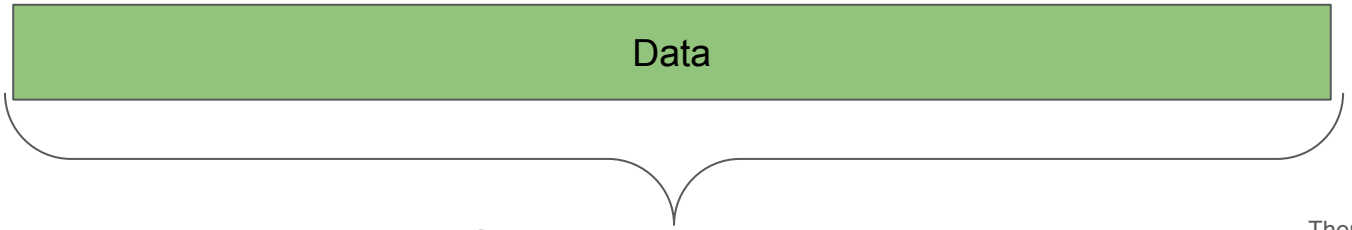
Easy-ish to remember…

Preprocessing, small changes, etc…

Hard to remember

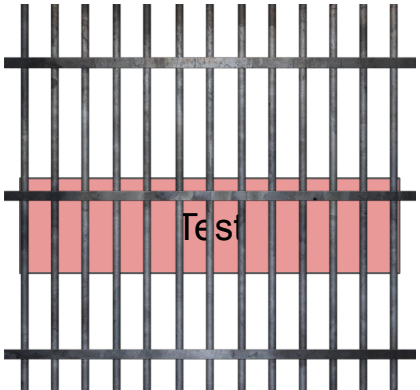
Very sinister!

Easiest way to combat this?

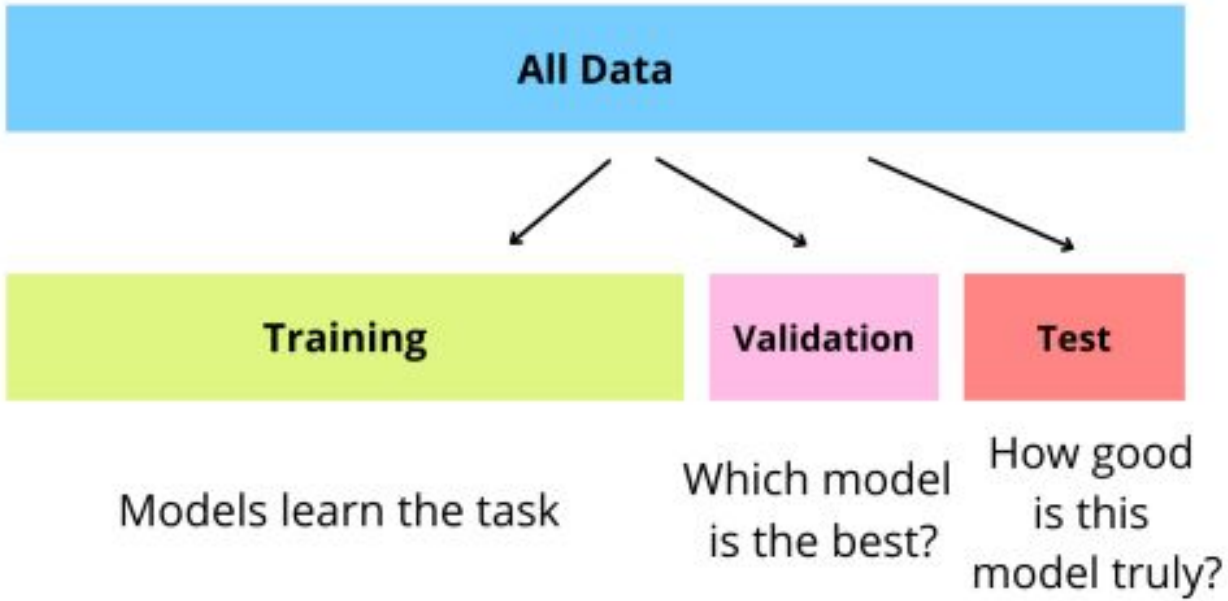


Only touch this until
everything is done

Then you can free him



Train/Val/Test Splits



Stock Prediction Dataset (E.g. train/val split)

Training

Validation

	$X^{(1)}_1$	$X^{(2)}_1$	Y
	Interest Rates	Employment	S&P 500 Price
0	1.943859	55.413571	2206.680582
1	2.258229	59.546305	2486.474488
2	2.215863	57.414687	2405.868337
3	1.977960	49.908353	2140.434475
4	2.437723	52.035492	2411.275663
5	2.143637	56.060598	2187.344909
6	2.148647	51.513208	
7	2.176184	53.475909	
8	2.125352	63.668422	
9	2.225682	56.993396	

$$\hat{y}_i = \mathbf{x}_i^\top \mathbf{w} + b$$

Problem 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?

Constant feature.
Likely remove the feature before standardization and and add it back to the dataset to avoid divide by zero error.

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.

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.

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

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 $[0, \infty)$.

Problem 2.1

Program 1

```
# Given dataset of 1000-by-50 feature
# matrix X, and 1000-by-1 labels vector

mu = np.mean(X, axis=0)
X = X - mu

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

ytrain = y[TRAIN]
Xtrain = X[TRAIN, :]

# solve for argmin_w ||Xtrain*w - ytrain||_2
w = np.linalg.solve(np.dot(Xtrain.T, Xtrain), np.dot(Xtrain.T, ytrain))

b = np.mean(ytrain)

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

train_error = np.dot(np.dot(Xtrain, w)+b - ytrain,
              np.dot(Xtrain, w)+b - ytrain ) / len(TRAIN)
test_error = np.dot(np.dot(Xtest, w)+b - ytest,
              np.dot(Xtest, w)+b - ytest ) / len(TEST)

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

mu is calculated from the **entire** data (train + test), intertwining them!

This is bad!

Correct procedure:

- **Split into train and test**
- **Compute the mean of the train data (μ_{train})**
- **De-mean both train and test data using μ_{train}**

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)
```

The final model is trained on BOTH the training and validation sets.

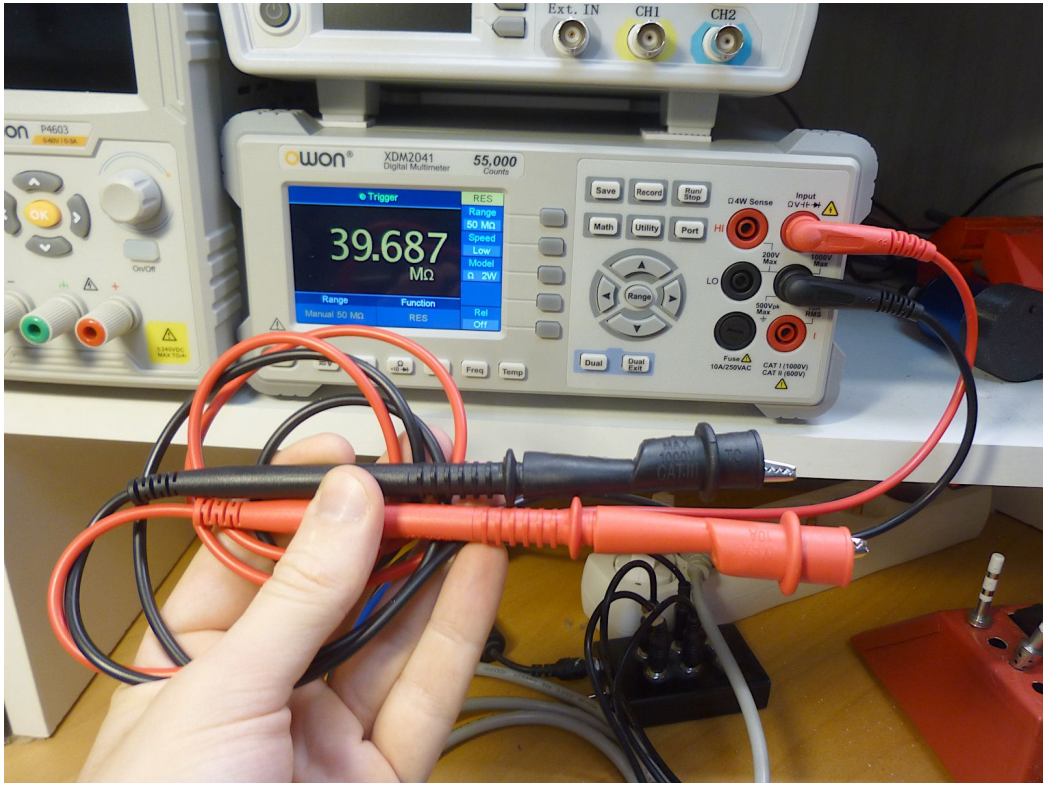
This is… eh…

Your hyperparameters selected on just the train data may not hold for train + val:

- **More data is good but you should ensure that the hyperparameters you tuned do not depend on the number of elements.**
- **Tradeoff between more data and better test error estimate**

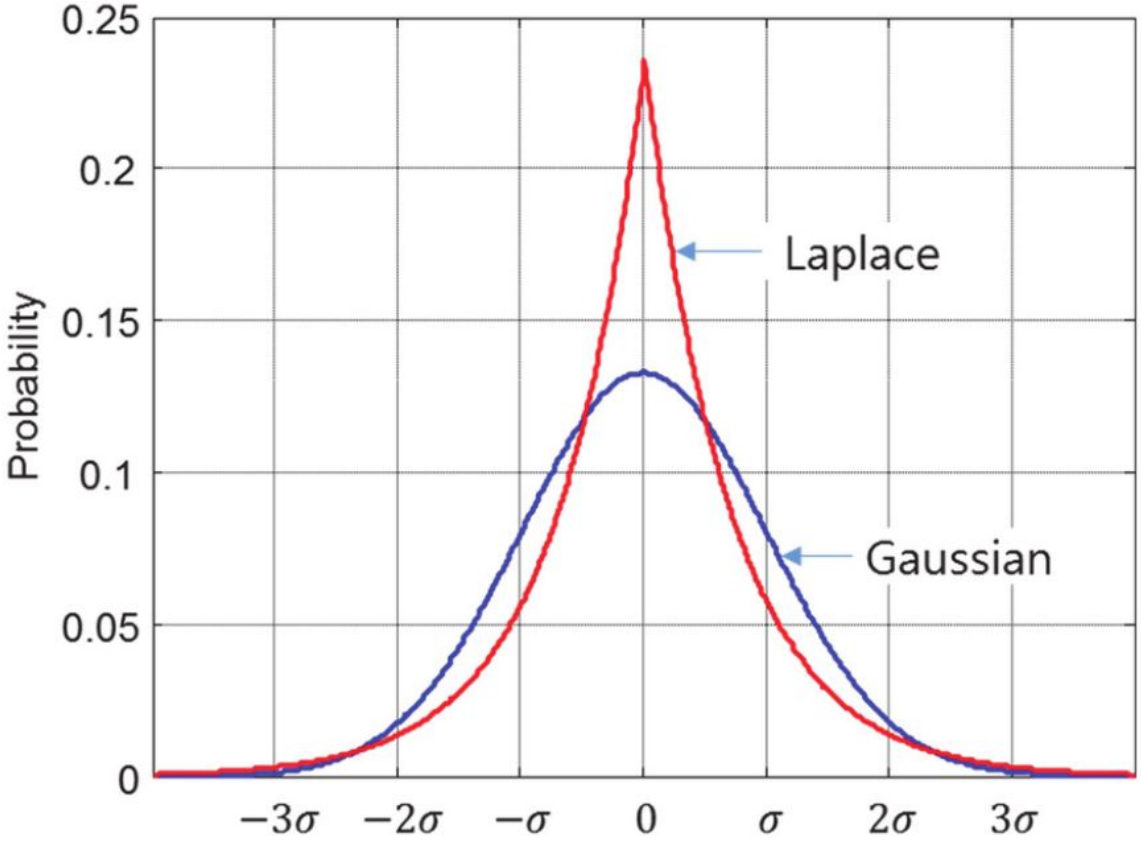
Problem 3b

Estimating from noisy measurements



- E.g. If your hand is not perfectly still, you introduce some noise
- The noise follows some distribution if studied
  - [e.g. more likely that your hand drops lower than goes higher]

Two Common Noise Distributions In The Class



Problem 3b - Context

Linear model with noise: $y_i = x_i^T w + v_i$

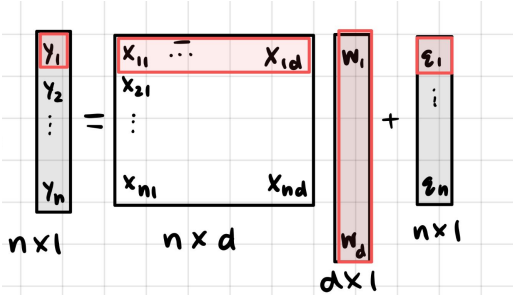
Rewrite as: $v_i = y_i - x_i^\top w$

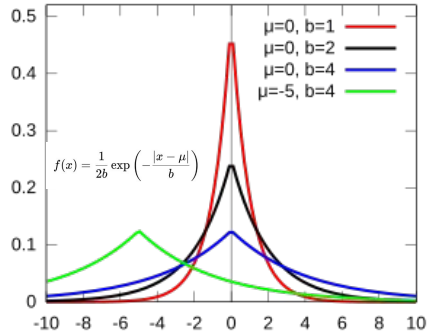
Noise has density p(z=v): $p(z) = (1/2a)\exp(-|z|/a)$

Then:

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

ie. the probability of observing the data





mu = 0 and b = a

We will use MLE to maximize the likelihood of seeing the data by finding $\hat{w}_{MLE}$

(which color curve best matches the noisy data distribution?)

Maximum Likelihood Estimation (MLE)

$$P(w \mid X, Y) = \frac{P(X, Y \mid w)\, P(w)}{P(X, Y)}$$

$$\text{posterior} = \frac{\text{likelihood} \cdot \text{prior}}{\text{evidence}}$$

$$\hat{w}_{MLE} = \arg\max_{w} p(X, Y|w)$$

Suppose we have data $\mathcal{D} = \{x^{(i)}\}_{i=1}^{N}$

$$\boldsymbol{\theta}^{\text{MLE}} = \underset{\boldsymbol{\theta}}{\text{argmax}} \prod_{i=1}^{N} p(\mathbf{x}^{(i)}|\boldsymbol{\theta})$$

Maximum Likelihood Estimate (MLE)

$$\hat{w}_{MLE} = \arg\max_{w} \log p_w((x_1, y_1), \cdots, (x_m, y_m))$$

$$\boldsymbol{\theta}^{\text{MLE}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \prod_{i=1}^{N} p(\mathbf{x}^{(i)}|\boldsymbol{\theta})$$

$$= \arg\max_{w} \log \prod_{i=1}^{m} p(y_i - x_i^\top w)$$

	$X^{(1)}_1$ Interest Rates	$X^{(2)}_1$ Employment	Y S&P 500 Price
0	1.943859	55.413571	2206.680582
1	2.258229	59.546305	2486.474488
2	2.215863	57.414687	2405.868337
3	1.977960	49.908353	2140.434475
4	2.437723	52.035492	2411.275663
5	2.143637	56.060598	2187.344909

Let's look at the steps *before* considering the distribution:

3b) Laplacian - Work in pairs for 3 minutes for the first step

(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.

$$\hat{w}_{MLE} = \arg\max_w \log p_w((x_1, y_1), \cdots, (x_m, y_m))$$

$$= \arg\max_w \log \prod_{i=1}^{m} p(y_i - x_i^\top w)$$

	X_1 Interest Rates	X_2 Employment	Y S&P 500 Price
0	1.943859	55.413571	2206.680582
1	2.258229	59.546305	2486.474486
2	2.215863	57.414687	2405.868337
3	1.977960	49.908353	2140.434475
4	2.437723	52.035492	2411.275863
5	2.143637	56.060598	2187.344909

$$= \arg\max_w \sum_{i=1}^{m} \log p(y_i - x_i^\top w) \quad [\log(ab) = \log a + \log b]$$

What do we plug in here?

$$= \arg\max_w \sum_{i=1}^{m} \log\left(\frac{1}{2a} \cdot e^{\frac{-|y_i - x_i^T w|}{a}}\right)$$

$$= \arg\max_w \sum_{i=1}^{m} \log\left(\frac{1}{2a}\right) - \frac{|y_i - x_i^T w|}{a}$$

$$= \arg\max_w \sum_{i=1}^{m} -\frac{|y_i - x_i^T w|}{a}$$ constant offset doesn't affect opti

$$= \frac{1}{a} \cdot \arg\max_w \sum_{i=1}^{m} -|y_i - x_i^T w|$$

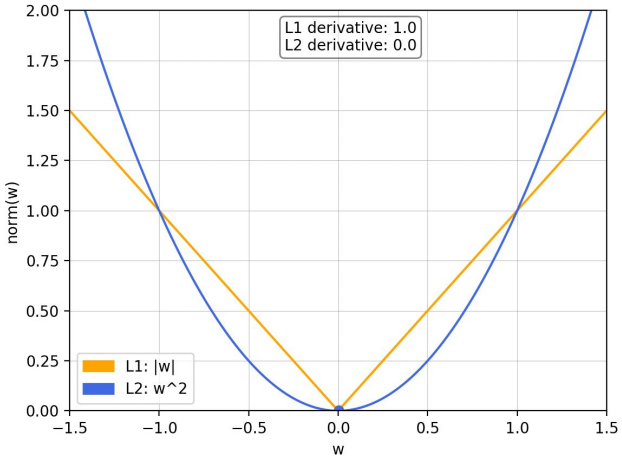
$$= \arg\max_w \sum_{i=1}^{m} -|y_i - x_i^T w|$$ constant offset doesn't affect opti

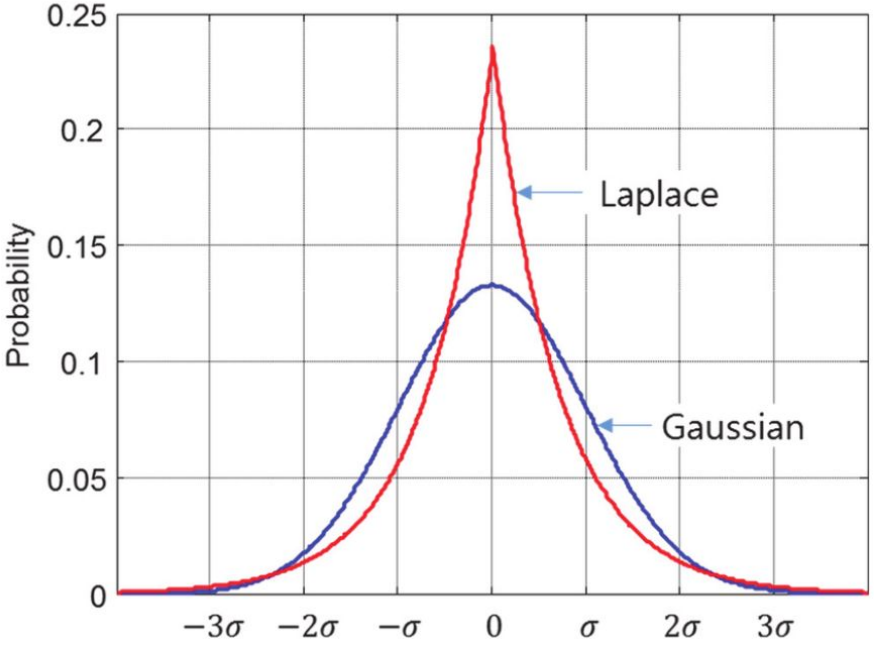
$$= \arg\min_w \sum_{i=1}^{m} |y_i - x_i^T w| = \arg\min_w \|Xw - Y\|_1$$

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

Cool Connection

Visual similarities exist between the L1 norm vs. the Laplacian, and the L2 norm vs. the Gaussian





Using Laplacian (sharp) as noise results in the L1 norm (sharp) in the optimization equation

Using Gaussian (smooth) as noise results in the l2 norm (smooth) in the optimization equation

Linear Regression Review

We will continue the optimization problem for the standard linear regression (gaussian noise assumption)

Linear Regression

$$\hat{\mathbf{w}} = \underset{w}{argmin} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$ Find $\mathbf{w}$ that minimizes the SSE. This is $\hat{\mathbf{w}}$.

$$= \underset{w}{argmin} \sum_{i=1}^{n} (y_i - (\mathbf{x}_i^\top \mathbf{w} + b))^2$$ Plug in $\hat{y}_i = \mathbf{x}_i^\top \mathbf{w} + b$.

$$= \underset{w}{argmin} \sum_{i=1}^{n} (y_i - (\mathbf{x}_i^\top \mathbf{w}))^2$$ Disregard b, since an intercept can be represented by appending a 1 to $\mathbf{x}$.

$$0 = \frac{\partial}{\partial w} \sum_{i=1}^{n} (y_i - (\mathbf{x}_i^\top \hat{\mathbf{w}}))^2$$ Find $\hat{\mathbf{w}}$ by taking the partial derivative of the argmin term and setting it equal to 0.

$$= \sum_{i=1}^{n} \frac{\partial}{\partial w} (y_i - (\mathbf{x}_i^\top \hat{\mathbf{w}}))^2$$ Property of derivatives.

$$= \sum_{i=1}^{n} 2(y_i - \mathbf{x}_i^\top \hat{\mathbf{w}})(-\mathbf{x}_i)$$ Derive.

Linear Regression

$$= \sum_{i=1}^{n} (y_i - \mathbf{x}_i^\top \hat{\mathbf{w}})(\mathbf{x}_i) \qquad \text{Divide by -2}$$

$$= \sum_{i=1}^{n} (\mathbf{x}_i)(y_i - \mathbf{x}_i^\top \hat{\mathbf{w}}) \qquad \text{The term } y_i - \mathbf{x}_i^\top \hat{\mathbf{w}} \text{ is a scalar. } c\mathbf{x} = \mathbf{x}c$$

$$= \sum_{i=1}^{n} (\mathbf{x}_i y_i - \mathbf{x}_i \mathbf{x}_i^\top \hat{\mathbf{w}}) \qquad \text{Distribute}$$

$$= \sum_{i=1}^{n} \mathbf{x}_i y_i - \sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top \hat{\mathbf{w}} \qquad \text{Distribute}$$

$$(\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top)\hat{\mathbf{w}} = \sum_{i=1}^{n} \mathbf{x}_i y_i \qquad \text{Move second term to LHS.}$$

$$(\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top)\hat{\mathbf{w}} = \sum_{i=1}^{n} \mathbf{x}_i y_i$$

$$X^T X = [x_1\ x_2\ \cdots\ x_n] \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{bmatrix}$$

$$X^T X = x_1 x_1^T + x_2 x_2^T + \cdots + x_n x_n^T$$

$$= \sum_{i=1}^{n} x_i x_i^T$$

$$X^T y = [x_1\ x_2\ \cdots\ x_n] \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$$X^T y = y_1 x_1 + y_2 x_2 + \cdots + y_n x_n$$

$$= \sum_{i=1}^{n} x_i y_i$$

$$(X^\top X)\hat{\mathbf{w}} = X^\top \mathbf{y}$$

Convert from vector summation form to matrix form

$$(X^\top X)^{-1}(X^\top X)\hat{\mathbf{w}} = (X^\top X)^{-1} X^\top \mathbf{y}$$

Left multiply by $(X^\top X)^{-1}$

$$\hat{\mathbf{w}} = (X^\top X)^{-1} X^\top \mathbf{y}$$

Cancel

Problem 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?

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 $\text{Null}(X) = \text{Null}(X^\top X)$.

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?

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.

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.)

$X^\top X$ 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: $\text{Row}(X^\top X) = \text{Null}(X^\top X)^\perp = \text{Null}(X)^\perp = \text{Row}(X)$.

Questions?