Linear Regression

Week	THURS. Week 7.
Assignment Due	
Assignment Done	☐
Due Date	
Notes Done	☑

Class Notes

Linear Regression

In this course we're focusing on 3 main problems:

- **Classification** → given observable data and target, we want to build a system to classify new observations accurately (weeks 3-6)
- **Regression** → instead of trying to compute an enumerated class ID, we're trying to compute a continuous value

Linear regression is based on attempting to predict a ***continuous*** value:

- Linear comes from the computation being a ***linear combination*** of our features.
  - Each feature is weighted by some weight factor.
    - $g(\mathbf{x}) = w_1 x_1 + w_2 x_2 + \ldots w_D x_D + b$
      - We can say that w_j is the weight of the j^{th} feature and b is the bias.
        - NOTE: Think of the bias as the "offset", like the y-intercept of $y = mx + b$
    - As with our other linear funciton, we can write this via linear algebra as:
      - $g(\mathbf{x}) = \mathbf{x}\mathbf{w} + b$

- Similar conversation of when we talked about logistic regression → with this approach, we did also learn some weights
  - Although logisitic regression is *not technically regression*, it is close to linear regression, which why is why share a similar name.

$$\text{Logistic regression:} \quad g(\boldsymbol{x}) = \frac{1}{1+e^{-(\boldsymbol{x}\boldsymbol{w}+b)}}$$
$$\text{Linear regression:} \quad g(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{w} + b$$

- The training process attempts to learn the weights of each feature to best approximate the target value.
  - As with logistic regression, we get a **bias** term to consider as well.
- Again, linear regression is conceptually close to logistic regression → logistic regression has the linear regression term that we're considering.
  - Logistic regression just takes linear regression term and plugs it into sigmoid function to get value between 0 and 1 $[0, 1]$.
  - Linear regression is theoretically unbounded, able to take terms from negative infinity to positive infinity$(-\infty, +\infty)$.

In *logistic regression*, to find optimal weights, we start off with the likelihood, but noting that math is easier with taking the log of it:

- We then turn this likelihood into a log likelihood and then negate it, turning a maximization problem into a minimization problem.
  - Again, for logistic regression, since we want to use calculus to find values of $\mathbf{w}$ to minimize this log likelihood term, we want a differentiable function.
    - The process of finding weights that minimize log loss is often referred to as the **log loss estimation**.

With **linear regression**, we need a *different* objective function because our target value can be any floating point value, not just 1 or 0.

- For **linear regression**, we usually start thinking about the squared error, $J = (y - \hat{y})^2$
- This takes a measure of how good is the difference between the actual value and the expected value squared altogether.
- Over all of the observations, we can take the average of the squared errors for *each observation.*

$$J = \frac{1}{N}\sum_{i=1}^{N}\left(Y_i - \widehat{Y}_i\right)^2 = \frac{1}{N}\sum_{i=1}^{N}\left(Y_i - g(X_i)\right)^2$$

- While we may use a log loss objective function for *logistic regression*, for **linear regression**, the most common objective function to use is **least squares**.
- The process of finding weights that minimize squared error is often referred to as the **least squared estimate**.

Objective Function Optimization

Again, to find the optimal weights, we can look at *direct* and/or *iterative* (gradient-based) solutions.

- Whereas logistic regression did NOT have a direct solution, linear regression does!
  - We'll focus on that and then briefly touch on the iterative approach.
- What we want: Values of $\mathbf{w}$ that *minimize* our objective function

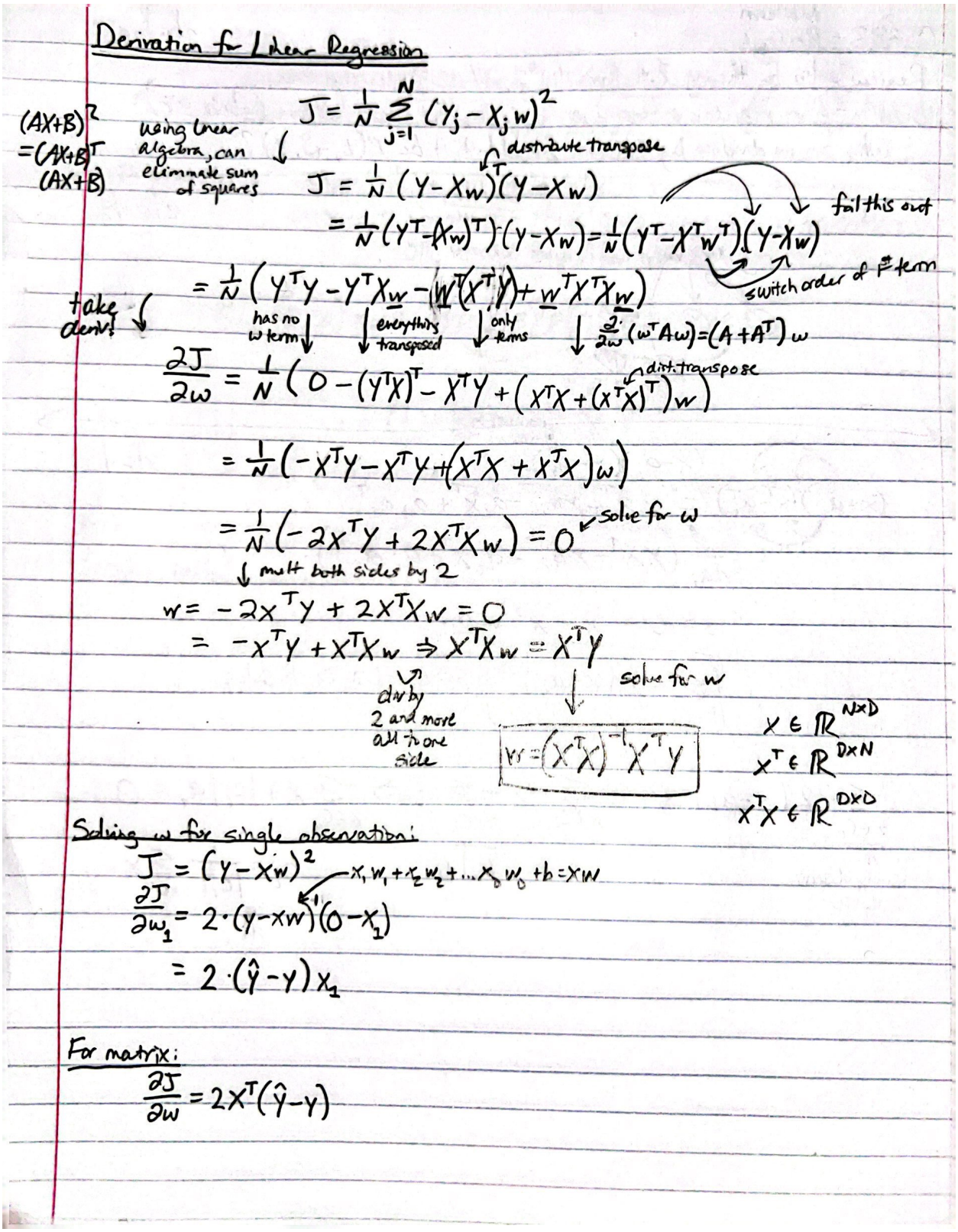
$$J = \frac{1}{N}\sum_{j=1}^{N}\left(Y_j - \widehat{Y}_j\right)^2 = \frac{1}{N}\sum_{j=1}^{N}\left(Y_j - \left(X_j\boldsymbol{w} + b\right)\right)^2$$

- Finding a direct solution would be easier if we can combine all the parameters into ***one variable/vector*** so that we don't need to take a derivative with respect to another variable, as b.

- A trick that we can do is add “dummy” feature” $x_0 = 1$ to our observations.
  - This makes our first parameter of w, w_0 is our bias b!
  - Now, we’re able to write our objective function as:

$$J = \frac{1}{N}\sum_{j=1}^{N}\left(Y_j - X_j w\right)^2$$

Below is the math undertaken to solve for a direct solution!


Derivation for Linear Regression
$J = \frac{1}{N}\sum_{j=1}^{N}(y_j - x_j w)^2$
$(AX+B)^2 = (AX+B)^T(AX+B)$
using linear algebra, can eliminate sum of squares
distribute transpose
$J = \frac{1}{N}(y - Xw)^T(y - Xw)$
$= \frac{1}{N}(y^T - (Xw)^T)(y - Xw) = \frac{1}{N}(y^T - X^T w^T)(y - Xw)$
foil this out
switch order of 2nd term
$= \frac{1}{N}(y^T y - y^T Xw - w^T X^T y + w^T X^T Xw)$
take deriv!
has no w term
everything transposed
only terms
$\frac{2}{2w}(w^T A w) = (A + A^T)w$
$\frac{2J}{2w} = \frac{1}{N}(0 - (y^T X)^T - X^T y + (X^T X + (X^T X)^T)w)$
dist. transpose
$= \frac{1}{N}(-X^T y - X^T y + (X^T X + X^T X)w)$
$= \frac{1}{N}(-2X^T y + 2X^T Xw) = 0$
solve for w
mult both sides by 2
$w = -2X^T y + 2X^T Xw = 0$
$= -X^T y + X^T Xw \Rightarrow X^T Xw = X^T y$
div by 2 and move all to one side
solve for w
$w = (X^T X)^{-1} X^T y$
$X \in \mathbb{R}^{N \times D}$
$X^T \in \mathbb{R}^{D \times N}$
$X^T X \in \mathbb{R}^{D \times D}$
Solving w for single observation:
$J = (y - Xw)^2$
$x_1 w_1 + x_2 w_2 + \ldots x_D w_D + b = Xw$
$\frac{2J}{2w_1} = 2 \cdot (y - Xw)^1 (0 - x_1)$
$= 2 \cdot (\hat{y} - y) x_1$
For matrix:
$\frac{2J}{2w} = 2X^T(\hat{y} - y)$

Example

EXAM1	EXAM2	EXAM3	FINAL
73	80	75	152
93	88	93	185
89	91	90	180
96	98	100	196
73	66	70	142
53	46	55	101
69	74	77	149
47	56	60	115
87	79	90	175
79	70	88	164
69	70	73	141
70	65	74	141
93	95	91	184
79	80	73	152
70	73	78	148
93	89	96	192
78	75	68	147
81	90	93	183
88	92	86	177
78	83	77	159
82	86	90	177
86	82	89	175
78	83	85	175
76	83	71	149
96	93	95	192

The three exams before the final are out of 100, while the final exam is out of 200.

- We'd like to predict the final exam score based on the scoring of the 3 previous exams.
    - We'll use the first 8 observations for validation, and use the remaining for training.

Since our direct solution involves calculating $X^T X$, we can see that this will end up being a 3×3 matrix, which is easy for us to invert.

- Using the weights that we calculate, we can turn the result of our model $\mathbf{w} = (X^T X)^{-1} X^T Y$ into a linear combination equation.
    - $\mathbf{w} = \begin{bmatrix} -9.96 \\ 0.38 \\ 0.57 \\ 1.19 \end{bmatrix}$
    - $FinalExam = 0.38 * Exam1 + 0.57 * Exam2 + 1.19 * Exam3 - 9.96$

        - We can extract some significance from the model itself from the values calculated from our closed-form solution.
            - The highest weight outside of the bias is the 1.19, pertaining to exam 3.
                - This may allow us to conclude that the most relevant feature to predicting a final exam is the third exam → makes sense in how people do on later exams indicate how they're going to do on the final exam

In computing statistics on our model with the training and validation data, we can see the following:

- **Training RMSE** → 2.39
- **Validation RSME** → 2.82
- **Training SMAPE** → 0.08
- **Validation RSME** → 0.06
    - Validation didn't do as well as training did, but it was not completely worse.
        - This means that the model hasn't not generalized so badly.

Iterative (Gradient-Based) Least Squares Approach

For an iterative approach, we can typically start with a single observation then batch as needed:

- $J = (y - \mathbf{x}\mathbf{w})^2$
- We'd want to start by finding $\frac{\partial J}{\partial w_i}$ (for a single observation) and then try to group this as $\frac{\partial J}{\partial \mathbf{w}}$ (for a batch):
    - For the $\frac{\partial J}{\partial w_i}$, we should arrive at:
        - $\frac{\partial J}{\partial w_i} = 2x^T(\hat{y} - y)$
    - For $\frac{\partial J}{\partial \mathbf{w}}$, we should arrive at:
        - $\frac{\partial J}{\partial \mathbf{w}} = \frac{2}{N} X^T(\hat{Y} - Y)$

- From here, we'd do a similar process to gradient learning in trying to get closer and closer to our solution.
- We would do this approach if there are too many features that doing the direct approach is infeasible.

Example

EXAM1	EXAM2	EXAM3	FINAL
73	80	75	152
93	88	93	185
89	91	90	180
96	98	100	196
73	66	70	142
53	46	55	101
69	74	77	149
47	56	60	115
87	79	90	175
79	70	88	164
69	70	73	141
70	65	74	141
93	95	91	184
79	80	73	152
70	73	78	148
93	89	96	192
78	75	68	147
81	90	93	183
88	92	86	177
78	83	77	159
82	86	90	177
86	82	89	175
78	83	85	175
76	83	71	149
96	93	95	192

In doing our example again with gradient descent, we make a few design decisions:

- Z-score our features since gradient learning is typically more stable if we z-score
  - This leads to faster convergence.
- Initialize each of the parameters to some random number in the range 10^{-4}
- Use all samples for each epoch (full batch)
- Make our η value = 4
- Run 100,000 epochs.

In running this type of batching, we see that our model learns the following weights:

$$\mathbf{w} = \begin{bmatrix} 166.53 \\ 3.31 \\ 4.87 \\ 10.88 \end{bmatrix}$$

- We can note here that the weights are quite different than from our direct solution because of the z-scoring.
    - We can notice the bias is large due to z-scoring in order to get to 0 to 200 range for final solution.
    - The evaluation matrix are effectively the same as when we evaluated above:
    - **Training RMSE** → 2.40
    - **Validation RSME** → 2.84
    - **Training SMAPE** → 0.084
    - **Validation RSME** → 0.06

Beyond Linear Regression

Linear regression can be considered to be somewhat *constrained* with making predictions just as a linear combination of the features:

- There can be non-linear models that exist to help with regression other than linear regression!
    - OR, we could do something similar to what we did with SVMs where we change our data's representation to include non-linear terms!
        - Ex: If we have only a single feature $\mathbf{x} = [x_1]$ and we wanted to model the equation $y = ax^2 + bx + c$, we can change the data to have three features instead.
            - We can now make $\mathbf{x} = [x_1^2, x_1, 1]$
            - Now, the weights learned are essentially learning the $y = ax^2 + bx + c$ formula.

- Augmenting the observations to have non-linear terms without them would allow us to learn non-linear ideas even with linear regression!