Machine Learning

Synthesis Questions

Linear Regression: Regression

1. What is SSE? Why do we want to minimize it?

Answer: SSE stands for Sum of Squared Errors, and it is one of the technique to make a good linear regression algorithm. The goal is to minimize the difference between the predicted values and the actual target values in the training data. It penalizes larger deviations more than the smaller ones by squaring the errors. Therefore, the smaller the SSE, the better fit the model is to the data, because it suggests its prediction is close to the actual observed values.

$$\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$$

2. What do the following symbols represent, and what are their shapes?

(Assume n is the dimensionality of the data and m is the number of datapoints)

- w: vector consisting of all the weights (coefficients) in the model

$$w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

- $\hat{w}$: vector consisting of the weights that minimize the SSE
- X: matrix holding all of m datapoints

$$X = \begin{bmatrix} x_1^\top \\ x_2^\top \\ \vdots \\ x_m^\top \end{bmatrix}$$

- y: vector holding all of actual values from the training set

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

3. Rewrite the Euclidean distance formula

Rewrite the Euclidean distance formula $d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$ using the L_2 norm $||x||$ given two vectors x and y. Do you notice any relationship between the L_2-norm and Euclidean distance?

$$||X||_2 = \sqrt{\sum_{i=1}^{n} X_i^2}$$

Let $X = x - y$

$$||x - y||_2 = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

$$||x - y||_2 = d(x,y)$$

- The **L2 norm** measures the *length* (magnitude) of a single vector.
- The **Euclidean distance** measures the *length of the difference vector* between two points.

4. In your own words, what are the differences between Ordinary, Ridge, and LASSO regressions?

Ordinary Regression: Not regularized regression, meaning that most of the time it is not generalized well and is too much tuned on its training data only.

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

To regularize the ordinary regression, we can use one of two methods: *Ridge Regression*, or *LASSO Regression*.

Using either the ridge regression or LASSO regression on $\hat{w}$, we get the regularized SSE:

$$\sum_{i=1}^{m}(y_i - \hat{y}_i)^2 + \lambda||\hat{w}||_{1/2}$$

The result gets larger if the elements of are large/unbalanced, so our goal is to minimize $||\hat{w}||_2$ or $||\hat{w}||_1$. Here, the λ term decides how much regularization we want. Finding the right value for λ is a balancing act.

Bonus: Solve for $\hat{w}$ in the Ridge regression setting.

Your answer should be the closed form solution for Ridge regression and the proof should be similar to the one given for ordinary linear regression. Here is the equation:

$$argmin_w \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda ||\hat{w}||_2^2$$

Logistic Regression: Classification

1. How does the sigmoid function play a role in making Logistic Regression a classification algorithm?

The sigmoid function result in a value from 0 to 1, indicating the probability of belonging to a certain class.

$$P(y = 1 \mid x) = \frac{1}{1 + e^{-(b + w^\top x)}}$$

We look at the result of linear equation $b + w^\top x$

- $b + w^\top x > 0 \implies$ value closer to 1 (likely belongs to class 1)
- $b + w^\top x < 0 \implies$ value closer to 0 (likely belongs to class 0)
- $b + w^\top x \sim 0 \implies$ value closer to 0.5 (likely belongs to any class)

So for the algorithm to say that the input is equally likely to be from either class, $b + w^\top x$ must have been a value near 0.

> While logistic regression is primarily used for binary classification, it can be extended to handle multiple classes through methods such as one-vs-rest (OvR). In this approach, we train multiple logistic regression models, each one responsible for distinguishing whether an input belongs to one particular class or any of the others. The model with the highest probability ultimately decides the final class label.

K-Means Clustering: Unsupervised Learning

1. Define "Unsupervised Learning" and how it is different than "Supervised Learning".

Supervised learning is when you train the model by giving the actual values data. *Unsupervised learning* is training the model by only giving them the unlabeled data to find patterns or structures.

2. What is a centroid? How is it calculated?

Centroid is the center of a cluster of similar data points. A good model has a minimum sum of squared distances between each data point and its cluster's *centroid* (usually a Euclidean distance). So the model repeats to find a new centroid by calculating the sum of squared distances, until convergence.

3. If the calculated centroids do not move very much after a few iterations, what does this indicate?

It means the model is stabilized, and therefore has found the minimum sum of squared distances between each data point and its cluster's *centroid*.

4. Find the lowest SSE

If you have n points to cluster, what value of k will give the lowest SSE no matter how the points are formed? (Hint: It's kind of a cheap, degenerate case!)

$$SSE = \sum_{j=1}^{k} \sum_{x_i \in C_j} ||x_i - \mu_j||^2$$

Without even looking at the equation, we can actually tell the answer is $k = n$, because the SSE will be zero! But of course, that is not a good model (it's extremely overfitting).

5. Why do you think k-means clustering is sensitive to centroid initialization?

Because the goal of this algorithm is to minimize distance between the points and the centroid. If the centroid was, for example, initially placed near the outlier, it will try to minimize the distance between the outlier, which is not really helpful. This means that a poor starting positions can lead the algorithm to converge on a local minimum instead of the global best one. Also, it doesn't always find the best centroids at the end, because it's not designed to escape local minima.

K-Means vs K-Medians Clustering

Give a concrete, real-world example of a situation where...

1. k-means clustering is a better choice than k-medians clustering

Comparing different breed of dogs based on their height, weight, and ear length. It doesn't need a *k-medians clustering* because every breed has distinct feature, and there wouldn't be much of outliers. *K-means clustering* will work the best when features are continuous, and dog breeds have measurable, averageable traits — like average weight, height, and ear length — that make sense to represent as means.

2. k-medians clustering is a better choice than k-means clustering

Grouping people to different major departments based on the percentage of courses taken in each subject area. For example, percentage of class in biology, chemistry, arts, and stuff like that. There will be A LOT of outliers (student like me, who took bunch of math and computer science courses but it actually a neuroscience major), so I believe *k-medians clustering* is the best. But I think it can still find some patterns.

3. neither of these algorithms will work well

Classifying genres of music based on rhythm and tempo. This is because many genres have overlapping tempo and rhythms, and there will be so many outliers.

Other ML Concepts

1. What does "expressivity" of a function mean?

An *expressivity* of a linear model is the indication of how well the model fits to the data. A linear model will have a low expressivity, and a higher-degree polynomial model will have a high expressivity.

2. How do over and under-fitting relate to bias and variance?

If the model has high bias, it will be very much under-fitting. If the variance is high, it will be over-fitting to the training set, and underfitting to any other data.

3. If a model gives low bias and variance while fitting to a dataset, is it a good choice?

Yes! Low bias means the model fits the training data well, and low variance means the model also generalizes well to new, unseen data. Low bias + low variance = accurate and stable predictions

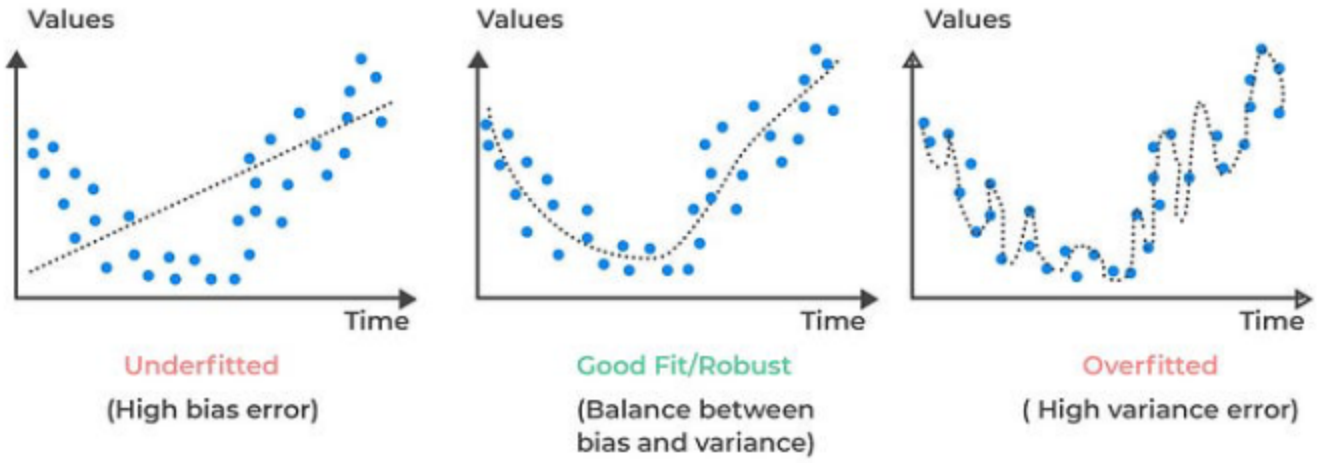

image-127.png

4. Why is knowing that your optimization problem is convex useful/good?

Because we don't need to worry about any "if" that our model can be further be optimized. The minimum SSE we get will always be both the local and the global minimum.

> If you are sure that the function you are optimizing is convex, then you can rest assured that you have found an optimal set of weights.

Bonus: Are either k-means or k-medians convex optimization problems? Give reasoning (does not need to be in proof form)

I don't think k-means and k-medians are not convex optimization problems. They are so sensitive to the initialization by assigning points to the nearest centroid, meaning they wouldn't try to find any other minimum. This means the algorithm can converge to different local minima depending on how the centroids are initialized, and there's no guarantee it will find the global minimum of the SSE.

Conclusion

> This section covered the fundamental concepts of machine learning, with a focus on supervised learning (linear regression) and unsupervised learning (k-means clustering). We also introduced important concepts like regularization (L1 and L2), the bias-variance tradeoff, and convexity. Additionally, we discussed the differences between k-means and k-medians clustering and clarified why logistic regression is a classification algorithm. Understanding these basics is crucial as machine learning continues to influence more of our lives. These concepts are also the most applicable in day-to-day ML, the backbone of modern systems. More often than not, Occam's Razor holds true: Don't use more than necessary and overcomplicate things.