CS 461 Notes Jan. 29 2026

Iris Martinez

January 29, 2026

Contents

1 Notes

1.1 Part 1 - Regression

- Regression lets us a function to predict a value.
- It is a broad collection of tasks in numerical predictions:
  - Predicting stocks
  - predicting home prices
  - predicteding "aggregate outcomes" of economic conditions (such as unemployment rates or sales volume)
- Essentially, **regression** refers to regressing *from* the answer to the data in the backwards pass of the algorithm, so that then we can *predict towards* the outcome given only data in a forward pass.

1.2 Linear regression, formalized

- In linear regression, we use a matrix X which is made up of i feature vectors.
- Row X_i gives the feature vector for data point i.
- Column $X - j$ gives the vector of all i observations of feature j
- The target that we regress against is a given vector y, which is the expected output.
- $\hat{y}$ is a vector of the output feature y across all i data points
- This allows us to make this regression equation: $\hat{y} = X\hat{w}$
- We can then find what weight vector $\hat{w}$ will operate on the data x to give the correct prediciton
- This boils down to solving a very large system of equations to find a relative importance for each feature that lets us predict the output value.
- We often include a "bias column" at the beginning of x.
- This is a column of all ones which allows us to skew the entire distribution of the system up or down, in effect giving the output data a "center" for its spread

1.3 Solving systems

- Programatically, there are simple library impementations of this. To program this out, we just need to use numpy and some matrix multiplications.

- Mathematically:

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

- Programatically(numpy):

```
X_transpose = X.T

XTX = X_transpose @ X

XTX_inv = np.linalg.inv(XTX)

XTy = X_transpose @ y

w = XTX_inv @ XTy
```

1.4 Constraining Problems

- A problem may have exist on an extremely high dimensionaly space. In other words, the output of a given thing may be defined by an unlimited number of features and corresponding linear equations.
- The output may be the linear combination of any number of features and their relative importances.
- If we model a problem with *too many* features, we risk overfitting. In other words, giving ourselves an abundance of features that allow us to directly correspond our weights to predict the exact training outputs.
- This is bad, we no longer model the trend, but instead embed into our weights all the necessary information to get out output values we already saw historically.
- Instead, we want to find the line where we just underfit, which lets us boil down the problem into a lower dimensional space that is actually following the trend rather than embedding the data.

2 Calculating error (loss)

2.1 Least Squares

We can calculate the total loss over all data points n like this:

$$Error = 1/2 \sum_{n=1}^{N} (y_n - \hat{w}^T \hat{x_n})^2$$

Which is the same as a single step matmul:

$$Error = 1/2 (X\hat{w} - \hat{y})^T (X\hat{w} - \hat{y})$$

The gradient of the error is:

$$X^T X\hat{w} - X^T\hat{y}$$

The minimum when the gradient is zero:

$$X^T X\hat{w} = X^T\hat{y}$$

2.2 Singular Value Decomposition

- We can write any transformation with a series of operations like this USV^T
- Geometrically, we *factor a complex matrix* into a rotation, a scalar multplication, and another rotation.
- This is useful because we can use this to reduce our inference of data from the weights. We can take the data matrix and write it in this form.
- As a result, we can simplify the operations for getting from weights to predicted values. This lets us expand the information embedded into the weights back into predictions.
- This also lets us easily invert the operation, letting us go from data to weights without calculating a difficult inverse.
- Instead, we describe this inverse as the opposite multiplication of the SVD. In other words we can do $V^T SU$
- This is called the **pseudo-inverse**
- This approximates the inverse and, which allows us to avoid overfitting and (somehow) helps us optimize for least squares loss.

3 Conclusion

Regression is about making quantitative predictions. Linear regression is compiling things down into a vector of weights.

4 Questions

- Is my understanding of what SVD is/does/is used for correct?
- Am i understanding the overfitting correctly?
- How does SVD optimize least squares? How does it let you use more features in your model?
- Is my interpretation of the effect of bias terms correct?