CS188–Winter 2020 — Homework 3 Solutions

Eugene Choi, SID 905368197

Collaborators: Luca Matsumoto (UID :204726167), Trevor Holt (UID: 204794180)

1.Overfitting

(a) You can tell if your trained model is overfitting when you get really high accuracy using the data that you trained with, but the accuracy drops significantly when using new data.

(b) It will be bad to deploy a model that has been overfitted because the purpose of a model is usually to predict new data with high accuracy which an overfitted model cannot do correctly.

(c) Overfitting can occur when you use an excess amount of information from your data or from your prior knowledge. What ends up happening is that your model predicts data using too much specificity which can lead to incorrect predictions when slight variations in the data occur. (model sucks at predicting when small changes are introduced)

(d) Weight Regularization: Weight regularization adds penalty to the model during training according the magnitude of the weights. When a model has larger weights, it is likely a sign that it is most likely overfitting. By adding penalty, we encourage our model to keep the weights small which helps manage overfitting by reducing generalization error and allowing our model to ignore less relevant input data.

Dropout: The dropout method is where randomly selected neurons are ignored during the training process. With some neurons being "dropped", the neural network must use its remaining neurons to handle the the responsibilities of the missing neurons when it comes to predicting. This helps to manage overfitting by making the network less sensitive to the specific weights of neurons which leads to better generalization and less impact to the accuracy of the predictions when new data is introduced.

2.Overfitting Mitigations

(a) Using a smaller training dataset: Does not mitigate overfitting. Overfitting often occurs when our model adjusts too much to the data. In the case of a smaller training dataset, there is not enough data to establish a common pattern between the data that we can use as part of the prediction. The common pattern that a data scientist may think he/she sees may actually be too specific of a feature which leads to overfitting.

(b) Restricting the maximum value any parameter can take on: Does mitigate overfitting. An example of this is the regularization method known as weight constraint which limits the magnitude of weights from exceeding a certain threshold. This leads to less of a negative impact on accuracy when new data is introduced because of better generalization.

(c) Training your neural network for longer (more iterations): Does not mitigate overfitting. Usually, if you want to prevent overfitting by altering the number of interations, you would usually stop it early instead of training longer in a method called early stopping (a method of regularization).

(d) Training a model with more parameters: Does not mitigate overfitting. This is one of the reasons for models overfitting! Rather than generalizing, the extra parameters causes the model to predict with more specifics.

(e) Randomly setting the outputs of 50 percent of the nodes in your neural network to zero: Does mitigate overfitting. This is known as the regularization method known as dropout which I described earlier above in question 1.

(f) Incorporating additional sparse features into your model: Does not mitigate ovfitting. Adding additional features in general will cause the model to lose its ability to generalize which would lead to overfitting.

(g) Initializing your parameters randomly instead of to zero: Does mitigate overfitting. Techniques like gradient descent will stop at zero so if you randomly set all values to zero, then you would not be able to fit anything meaning that overfitting would never occur.

(h) Training your model on a graphics processing unit or specialized accelerator chip instead of a CPU: Does not mitigate overfitting. This would just improve the performance speed of the model, but would not actually solve the issue of overfitting.

3.K-Nearest Neighbors

(a) When your data is not normalized, the range used to represent the data may be too large which if the actual distance between the data point coordinates is small, will squish the data points much closer together. If a sample is chosen, we will not be able to really tell which class that sample is associated with because the euclidean distance between all the points will appear way to small. Normalization fixes that issue by making sure that the range that is used to represent the data points is proportional to the data so that we will be able to see the euclidean distance and accurately classify which class a sample would belong to.

(b) We choose an odd value for k to avoid having tied votes (two classes achieving the same score).

(c) To normalize the data, I chose to divide the y values by 760 which was the largest y value so that the values in the y column lied between [0 - 1]. I then divided the x values by 1.2.
Training Set
x.....y.....L
0.17 0.46 0
0.08 0.99 0
0.25 0.92 0
0.08 0.66 0
0.17 0.66 0
1.00 0.53 1
0.75 0.54 1
0.92 0.51 1
0.08 1.00 1

Test Set
x.....y.....L(pred)
0.08 1.00
0.17 0.92
0.50 0.26

(d) When k = 3
x.....y.....L(pred)
0.08 1.00 0
0.17 0.92 0
0.50 0.26 1

x	y	L		x	y	L(pred)	euc dist (0.50, 0.26)	euc dist (0.08,1)	euc dist (0.17,0.92)
0.17	0.46	0		0.08	1		0.3858756276	0.5474486277	0.46
0.08	0.99	0		0.17	0.92		0.8421995013	0.01	0.1140175425
0.25	0.92	0		0.5	0.26		0.7057619995	0.1878829423	0.08
0.08	0.66	0					0.58	0.34	0.2751363298
0.17	0.66	0					0.5185556865	0.3517101079	0.26
1	0.53	1					0.5682429058	1.033102125	0.9170605214
0.75	0.54	1					0.3753664876	0.8127115109	0.6933974329
0.92	0.51	1					0.4887739764	0.9724710793	0.8547514259
0.08	1	1					0.8508818954	0	0.1204159458

Figure 1: K = 3

(e) When k = 1
x.....y.....L(pred)

0.08 1.00 1
0.17 0.92 0
0.50 0.26 1

x	y	L		x	y	L(pred)	euc dist (0.50, 0.26)	euc dist (0.08,1)	euc dist (0.17,0.92)
0.17	0.46	0		0.08	1		0.3858756276	0.5474486277	0.4(
0.08	0.99	0		0.17	0.92		0.8421995013	0.01	0.114017542
0.25	0.92	0		0.5	0.26		0.7057619995	0.1878829423	0.0
0.08	0.66	0					0.58	0.34	0.275136329
0.17	0.66	0					0.5185556865	0.3517101079	0.2(
1	0.53	1					0.5682429058	1.033102125	0.917060521
0.75	0.54	1					0.3753664876	0.8127115109	0.693397432
0.92	0.51	1					0.4887739764	0.9724710793	0.854751425
0.08	1	1					0.8508818954	0	0.120415945

Figure 2: K = 1

(f) When k = 5
x.....y.....L(pred)
0.08 1.00 0
0.17 0.92 0
0.50 0.26 1

x	y	L		x	y	L(pred)	euc dist (0.50, 0.26)	euc dist (0.08,1)	euc dist (0.17,0.92)
0.17	0.46	0		0.08	1		0.3858756276	0.5474486277	0.46
0.08	0.99	0		0.17	0.92		0.8421995013	0.01	0.1140175425
0.25	0.92	0		0.5	0.26		0.7057619995	0.1878829423	0.08
0.08	0.66	0					0.58	0.34	0.2751363298
0.17	0.66	0					0.5185556865	0.3517101079	0.26
1	0.53	1					0.5682429058	1.033102125	0.9170605214
0.75	0.54	1					0.3753664876	0.8127115109	0.6933974329
0.92	0.51	1					0.4887739764	0.9724710793	0.8547514259
0.08	1	1					0.8508818954	0	0.1204159458

Figure 3: K = 5

(g) A potential issue with using a low k for KNN is that it will be much noisier and have a larger influence on the classification result.

(h) A potential issue with selecting a k that is too high is that bias increases along with the computational expense.

(i) If I had a dataset and wanted to understand how well KNN with a k of 3 performed on it, I would use f1-score, recall, and precision to quantify the performance. We can get a baseline for our models predictive power by finding the accuracy and fine tune our model using f1-score, recall, and precision to look for parts of our model that we can change.

(j) If I had a dataset consisting of 200 samples, I would select the most optimal k by calculating the accuracy of my model. I would do this by running a loop that will calculate the accuracy of my model with n neighbors that we got when we created a list of neighbors.

4.Principal Component Analysis

(a) Data with a linear structure: PCA works well with linear structures because PCA is a type of feature extraction that combines input variables in a way that allows us to drop the variable with least importance. The new variable values that we get out of PCA are all independent from one another which is also true for linear models which is required for independent variable to be independent from one another – making linear structures work well with PCA.

(b) Data lying on a hyperbolic plane: Would not work well with PCA because of the required use of a kernel transformation to make the data linear.

(c) A dataset containing non-normalized features: Would not work well with PCA because PCA tries to maximize variance which is best done when using normalized features.

(d) A dataset where each feature is statistically independent of all others: Does not work well with PCA because PCA creates uncorrelated variables from correlated variables. When this dataset us passed into PCA, it would not do much because the values are already independent from each other.

5.Artificial Neural Networks

(a)

a1	a2	a3 (ypred)	b0	b1	b2	w0	w1	w2	w3	w4
0.7310585786	0.002472623157	0.07885604513	1	-6	-3.93	3	4	6	5	2
0.9996647254	0.9933071866	0.8852943595	1.000225236	-5.999994349	−3.92942720728	3	4	6	5	2.000418745
0.02360761314	0.2689312083	0.02689312083	1.000224455	−6.00002532369	−3.93059202306	2.999999219	3.999999219	5.999969025	4.999969025	1.99925432
			1.000194602	-6.001792619	-3.932760794	2.999999274	3.999969366	5.999971201	4.99820173	1.997100581

w5
4
4.000001416
3.998844396
3.998320431

The updated weights and biases are at the bottom of each column.

(b)

(c)

(d)

(e)

(f)

(g)

6.Reinforcement Learning (REMOVED)

7.(Money)ball So Hard

(a) The data science premise of Moneyball is how Billy Beanie – general manager of the Oakland A's – decided to take a different approach to baseball by using data analytics after losing to the Yankee's in the 2001 American League Championship Series and having many of the star players depart from the team. On a limited budget of $41 million, he performed data mining on a large pool of players to look for two main features that he used to predict a player's ability: slugging percentage and the on-base percentage. If the player met both quotas, he would be considered a good player and be scouted onto the team. Instead of competing for expensive high batting average players, he looked more for players with high on-base percentages because players with higher on-base percentages would most likely generate runs for the team.

The first step that I would take to solving the problem at hand is collect data by web scraping player statistics off of mlb.com as well as other MLB affiliated websites. From the data that I collected, I would look for players that do not meet the certain threshold values for the desired features that i am looking for and remove them from the collected data. Removing them would clean my data which would allow me to preprocess and then normalize my data to later be fit to a certain ML model that will tell me whether or not to sign the player. The ML model that I would most likely use is a classification model because all I need is a yes or no.

(b) An issue that can arise from the practice of this method is the misrepresentation of a player. If the player has not played many games, then their overall performance as a player is still yet to be credible because it can drastically change in the span of a season. For example, a rookie player may have a higher batting average (80%) then a seasoned player (60%), but that would only be because the rookie only played one game where he hit 3/4 balls while the seasoned player has played has hit 60% of the balls pitched over the duration of the 70 games he has played. This method of looking at on-base percentage and slugging percentage can be improved by adding in a feature that incorporates how many games the player has actually been a part of in the major leagues to determine the experience that the player has had as a professional.