CS188–Winter 2020 — Homework 3 Solutions

Hermmy Wang, SID 704978214

Collaborators: None

2. Overfitting

(a) A model is overfitting when the model fits the training data and even its outliers so that it cannot be generalized to new data. In other words, the training error is extremely low, while the testing error is high.

(b) The model could not be generalized to other datasets. When the model is applied on new data, the performance will be inaccurate and poor.

(c) Overfitting occurs when the model captures the noises of the training data. This might be due to that the training time (number of iterations) is too long on the dataset. Or the model is initialized to be complex and gets excessively complicated during training.

(d) We can use dropout to randomly drop units along with their connections during training. Each unit retrained with fixed probability p, independent of other units. We need to choose a hyper-parameter p. Another way is to use weight decay, introducing a regularization term to the objective function to penalize big weights. It will determine how dominant regularization is during gradient computation. Bigger weights will have bigger penalty and smaller weights will have smaller penalty.

3. Overfitting mitigations

(a) No. We should use a larger dataset to avoid overfitting.

(b) Yes. If restricting a maximum number on parameters can help remove outliers and set a reasonable bound on the feature, then it is an effective way to prevent learning the noises of the training data.

(c) No. Training your neural network for longer (more iterations) increases chance of overfitting.

(d) No. We should select only relevant features to feed into the model to prevent the model getting excessively complicated.

(e) Yes. Dropout is a regularization technique to prevent overfitting.

(f) No. Additional sparse features can cause overfitting.

(g) Yes. Initializing parameters randomly increases the information of a dataset, which helps reduce overfitting. Zero-initialization is not a recommended practice in most cases. The model will not likely to be trained.

(h) No. Using accelerator chips increases the speed of computation. Calculating a complex model will take shorter time. But this does not help to solve the problem of overfitting.

4. K-Nearest Neighbors

(a) KNN is subject to scale effects. Different features may have different measurement scales. If we do not normalize the features, features with larger scale will have a greater influence on the distance between samples and may bias the performance of the classifier. Therefore, by normalizing the features, they are set to similar range and scale. Each feature should weigh similarly in the distance calculation.

(b) To avoid tie in the case that there are same number of neighbors from different classes.

(c) Calculate normalized = x/max(x)

x	y	L
0.167	0.461	0
0.083	0.987	0
0.250	0.921	0
0.083	0.658	0
0.167	0.658	0
1.000	0.526	1
0.750	0.539	1
0.917	0.513	1
0.083	1.000	1

(d) For (0.1, 760), the closest three points are (0.1, 760) in 1, (0.1, 750) in 0, and (0.3, 700) in 0. It belongs to 0.
For (0.2, 700), the closest three points are (0.1, 760) in 1, (0.1, 750) in 0, and (0.3, 700) in 0. It belongs to 0.
For (0.6, 200), the closest three points are (0.2, 350) in 0, (0.9, 410) in 1, and (1.1, 390) in 1. It belongs to 1.

(e) For (0.1, 760), the closest points are (0.1, 760). It belongs to 1.
For (0.2, 700), the closest points are (0.3, 700). It belongs to 0.
For (0.6, 200), the closest points are (0.9, 410). It belongs to 1.

(f) For (0.1, 760), the closest five points are (0.1, 760), (0.1, 750), (0.3, 700), (0.1, 500), (0.2, 500). It belongs to 0.
For (0.2, 700), the closest five points are (0.1, 760), (0.1, 750), (0.3, 700), (0.1, 500), (0.2, 500). It belongs to 0.
For (0.6, 200), the closest five points are (0.2, 350), (0.9, 410), (1.1, 390), (1.2, 400), (0.2, 500). It belongs to 1.

(g) The model can suffer from overfitting with small value k. Noise will have higher influence on the result.

(h) High value k smooths decision boundary. It could cause over generalization and underfitting. The computation is also expensive and ineffective.

(i) Use N-fold cross validation. Suppose the labeled instances are divide into n subsets of equals size. Run classifier n times, with each of the fold as test sets, the rest n-1 folds for training. Each run gives an accuracy result.

(j) Run KNN with different k values and observe the classifiers' accuracy results. Once the accuracy stops improving at a value of k, that k is optimal.

5. Principal Component Analysis

(a) Yes. PCA is mathematically defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by some scalar projection of the data comes to lie on the first coordinate, so PCA works best on complex linear data.

(b) No. If your dataset is highly nonlinear using PCA will distort the distances between points and most likely your downstream algorithm will yield misleading results.

(c) No. We should first normalize features and then do PCA. PCA is performed by finding eigenvectors of a covariance matrix. If the features are not normalized, the covariance will be larger on features with large scale and thus affect PCA.

(d) No. PCA converts a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. If the features are already independent of each other, no feature extraction can be done without losing important information. PCA cannot effectively reduce dimensionality.

6. Artificial Neural Networks

(a) Define:

$z_5 = w_2 * x_1 + w_3 * x_2 + b_1$
$z_4 = w_0 * x_1 + w_1 * x_2 + b_0$
$z_3 = g(z_5)$
$z_2 = g(z_4)$
$z_1 = w_4 * z_2 + w_5 * z_3 + b_2$
$y_{pred} = g(z_1)$
$Loss = \frac{1}{2}(y - y_{pred})^2$

$$y_{pred} = g(w_4 * g(w_0 * x_1 + w_1 * x_2 + b_0) + w_5 * g(w_2 * x_1 + w_3 * x_2 + b_1) + b_2) = 0.075$$

$$\frac{\partial Loss}{\partial b_2} = (y_{pred} - y) * g(z_1) * (1 - g(z_1)) * 1$$

$$\frac{\partial Loss}{\partial w_4} = (y_{pred} - y) * g(z_1) * (1 - g(z_1)) * z_2$$

$$\frac{\partial Loss}{\partial w_5} = (y_{pred} - y) * g(z_1) * (1 - g(z_1)) * z_3$$

$$\frac{\partial Loss}{\partial b_0} = (y_{pred} - y) * g(z_1) * (1 - g(z_1)) * w_4 * g(z_4) * (1 - g(z_4)) * 1$$

$$\frac{\partial Loss}{\partial w_0} = (y_{pred} - y) * g(z_1) * (1 - g(z_1)) * w_4 * g(z_4) * (1 - g(z_4)) * x_1$$

$$\frac{\partial Loss}{\partial w_1} = (y_{pred} - y) * g(z_1) * (1 - g(z_1)) * w_4 * g(z_4) * (1 - g(z_4)) * x_2$$

$$\frac{\partial Loss}{\partial b_1} = (y_{pred} - y) * g(z_1) * (1 - g(z_1)) * w_5 * g(z_5) * (1 - g(z_5)) * 1$$

$$\frac{\partial Loss}{\partial w_2} = (y_{pred} - y) * g(z_1) * (1 - g(z_1)) * w_5 * g(z_5) * (1 - g(z_5)) * x_1$$

$$\frac{\partial Loss}{\partial w_3} = (y_{pred} - y) * g(z_1) * (1 - g(z_1)) * w_5 * g(z_5) * (1 - g(z_5)) * x_2$$

$$w_i \leftarrow w_i - 0.1 * \frac{\partial Loss}{\partial w_i}$$

Parameter	Initial	1st iter.	2nd iter.	3rd iter.
b0	1.00000	0.99977	0.99978	0.99997
b1	-6.00000	-6.00001	-5.99997	-5.98843
b2	-3.93000	-3.93057	-3.92940	-3.91473
w0	3.00000	3.00000	3.00000	3.00000
w1	4.00000	4.00000	4.00000	4.00020
w2	6.00000	6.00000	6.00003	6.00003
w3	5.00000	5.00000	5.00003	5.01157
w4	2.00000	1.99958	2.00075	2.01532
w5	4.00000	4.00000	4.00116	4.00511

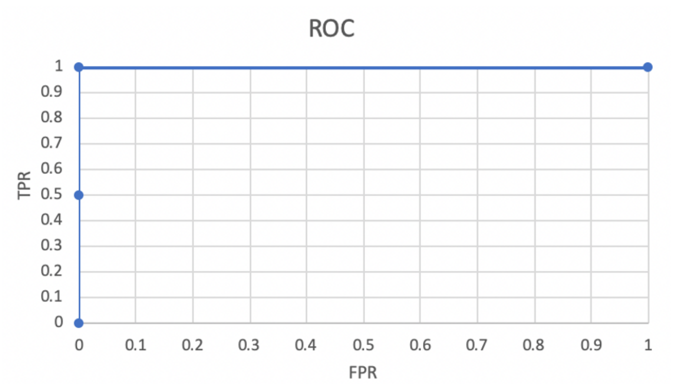
x1	x2	$y_{pred,raw}$	y_{pred}	y
0	0	0.078715724	0	0
1	1	0.884991643	1	1
0	1	0.295335168	0	1

(b) Accuracy = 2/3 = 67%

(c) Precision = 1/1 = 100%

(d) Recall = 1/2 = 50%

(e) F1 score = 2*1*0.5/1.5 = 67%

(f) 

(g) AUC = 1*1 = 1

8. Moneyball

(a) Oakland A lost three key players because the team was under budget. The general manager Billy wants to find other cheaper players who can replace these three players. Players with good reputation often cost a lot of money but the owner refuses to allocate more money. Bill then met Peter Brand, who is a Yale graduate with an economics degree. He told Bill that the problem is not to buy players but to buy runs. Bill was inspired and hired Peter to be his new assistant GM. In order to solve the problem of finding players who can replace the lost players, we need to use Data Science. First, we need to collect data about many baseball players. The features include batting statistics, baserunning statistics, pitching statistics, and fielding statistics. Depending on which position the player is about to play, we weight each set of statistics differently. In the data cleaning step, we can remove the players who do not meet some enforced requirements. In the movie's case, we can remove the super expensive players, because the team could't afford them and their original team wouldn't agree to make the deal. Then, we can train a model with all existing players' stats to predict a player's performance based on his statistics. The player with the best performance and an affordable price should be the player Bill wanted.

(b) There is randomness in every game. Although the data might tell the performance of a player, we cannot guarantee that he will do well in every game and has no fault. Therefore, hard work and good training are also necessary. OBP cannot entirely determine the value of a player. There are other metrics such as slugging percentage and on-base weighted average. Also, the reaction of fans depends on the selection of players, too. Choosing players that do not seem promising in the eyes of the public might increase fans' discontent and media's criticism. These factors can decrease the commercial value of a baseball team. It is not wise to choose players purely based on his OBP. While there are players who are chosen because of their OBP, we also need to retain all-star players who have a lot fans and high commercial value to bring more popularity to the team.