**CSC411 – A3**

YUHENG HUANG

Dec 2017

1. ***20 Newsgroups predictions report:***
2. **Bernoulli Naive Bayes (Baseline):**

*Accuracy on training set:* ***0.5987272405868835***

*Accuracy on test set:* ***0.4579129049389272***

1. **Multinomial Naive Bayes:**

*Accuracy on training set:* ***0.9589004772847799***

*Accuracy on test set:* ***0.7002124269782263 (best performance)***

*Alpha:* ***0.01***

**Q:** how to pick the hyperparameter Alpha:

**A:** I use cross validation ***(kfold = 10)*** to pick the best hyperparameter **alpha** for Multinomial Naive Bayes. First, I select alpha’s range from **0.01 to 1**, and then I uniformly random generate 100 samples of alpha. Second, I put all those 100-different alphas into cross validation and then calculate its mean score (cross validation will return me 10 scores for each alpha). After I get 100 mean scores, I will pick the highest score as the best hyperparameter alpha.

**Q:** Why I pick this method?

**A:** Because the classifier that we are going to create is 20 classes classifier, I think it is not belong to Bernoulli, but Multinomial. Therefore, I decide to try Multinomial Naïve Bayes, and they work just as I thought, much better than Bernoulli Naive Bayes.

*\*For detail, please visit q1.py*

1. **Linear SVM Classifier:**

*Accuracy on training set:* ***0.9671203818278239***

*Accuracy on test set:* ***0.6972915560276155***

*C:* ***0.53***

how to pick the hyperparameter C:

I use cross validation ***(kfold = 5)*** to pick the best hyperparameter **C** for Linear SVM Classifier. First, I select alpha’s range from **0.01 to 3**, and then I uniformly random generate 10 samples of alpha. Second, I put all those 10-different C into cross validation and then calculate its mean score (cross validation will return me 5 scores for each C). After I get 10 mean scores, I will pick the highest score as the best hyperparameter C.

**Q:** Why I pick this method?

**A:** SVM is very Effective in high dimensional spaces, so I decide to give it a try. And the result is not bad too, just like what I expected. However, the training process is a little bit long.

*\*For detail, please visit q1.py*

1. **Logistic Regression:**

*Accuracy on training set:* ***0.9399858582287431***

*Accuracy on test set:* ***0.6836165693043016***

*C:* ***2.19***

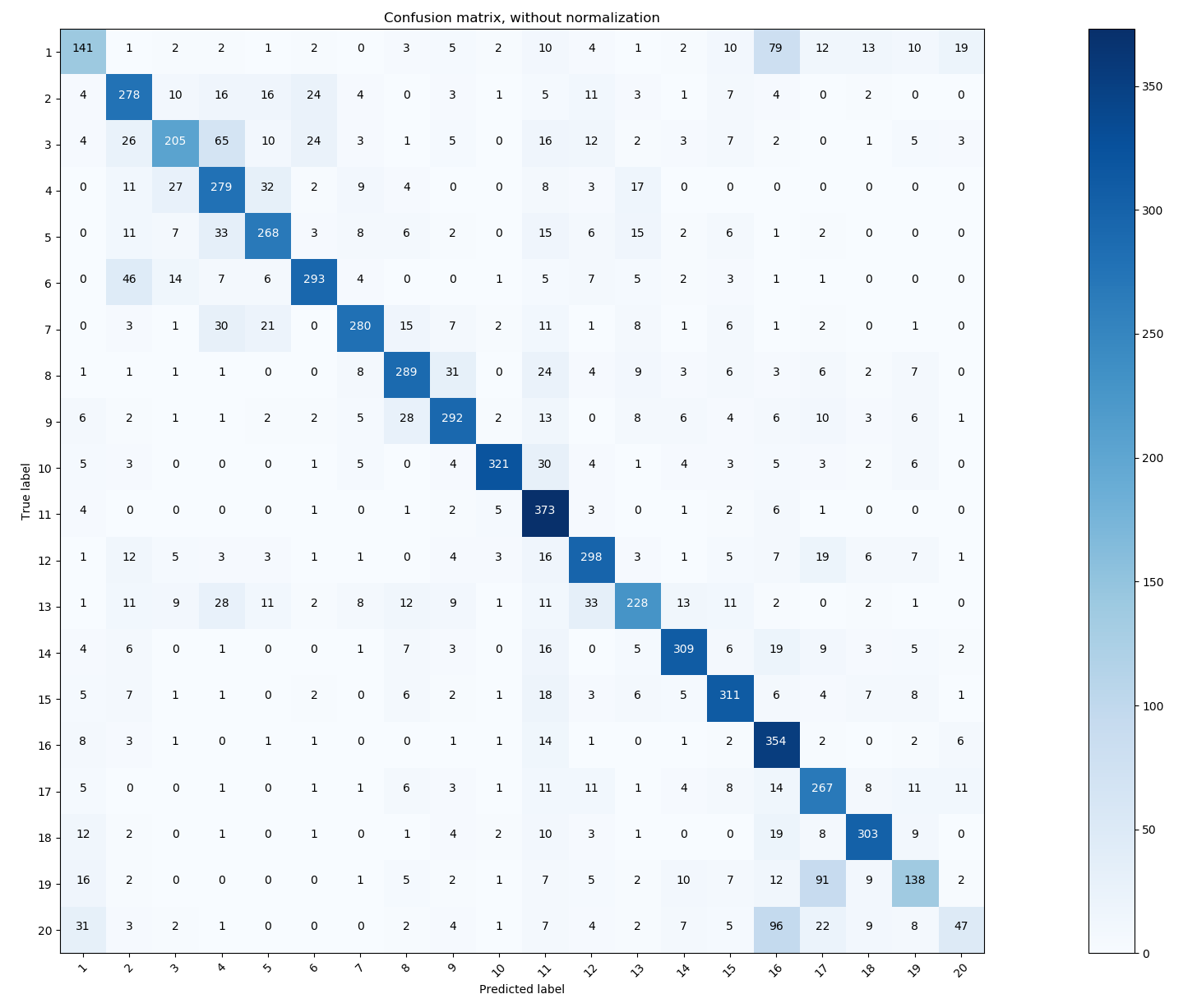
how to pick the hyperparameter C:

I use cross validation ***(kfold = 5)*** to pick the best hyperparameter **C** for Logistic Regression Classifier. First, I select alpha’s range from **0.01 to 3**, and then I uniformly random generate 10 samples of alpha. Second, I put all those 10-different C into cross validation and then calculate its mean score (cross validation will return me 5 scores for each C). After I get 10 mean scores, I will pick the highest score as the best hyperparameter C.

**Q:** Why I pick this method?

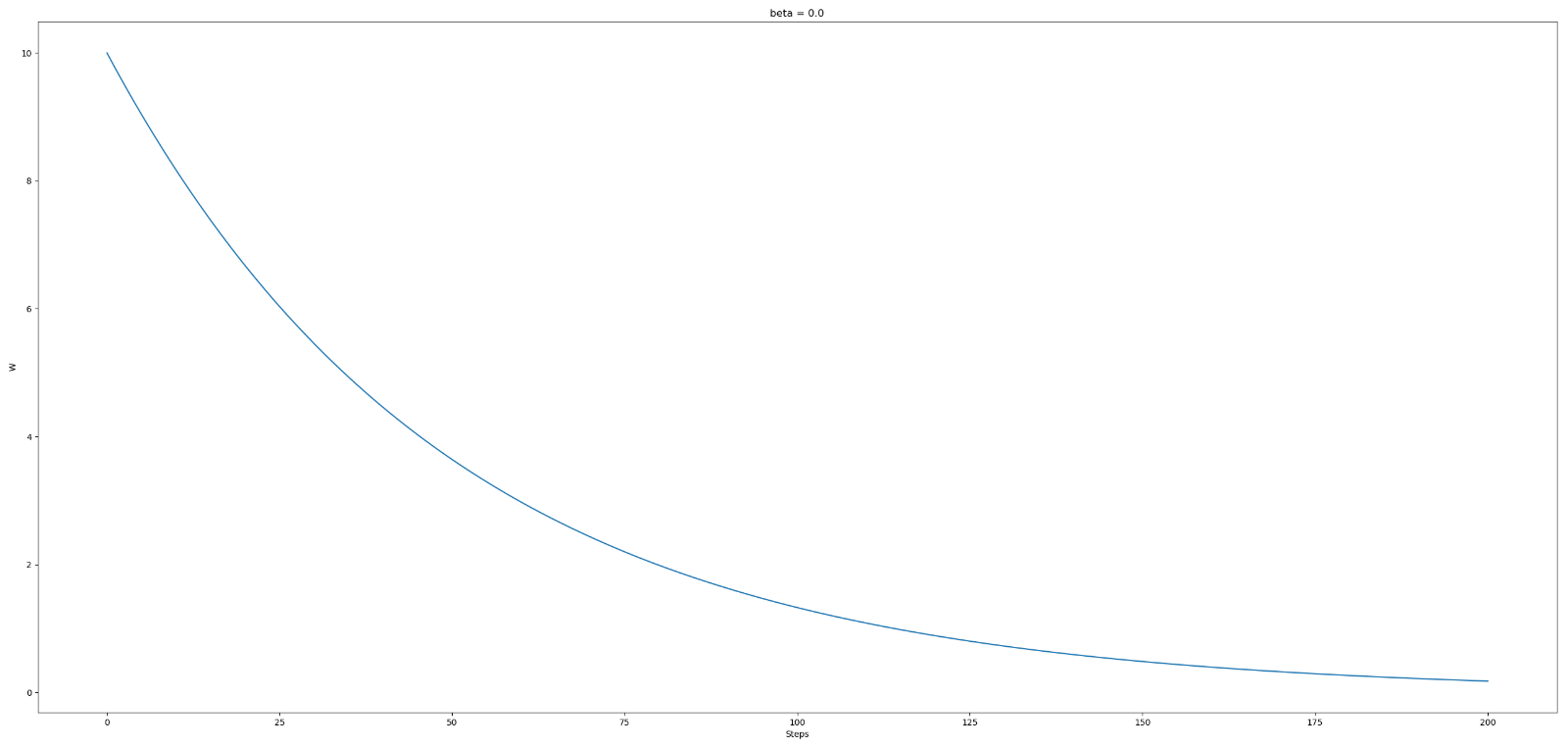
**A:** Because Logistic Regression may handle non linear effects, and its result is as what I thought, the accuracy is higher than the baseline, but lower than SVM and Multinomial Naive Bayes.

*\*For detail, please visit q1.py*

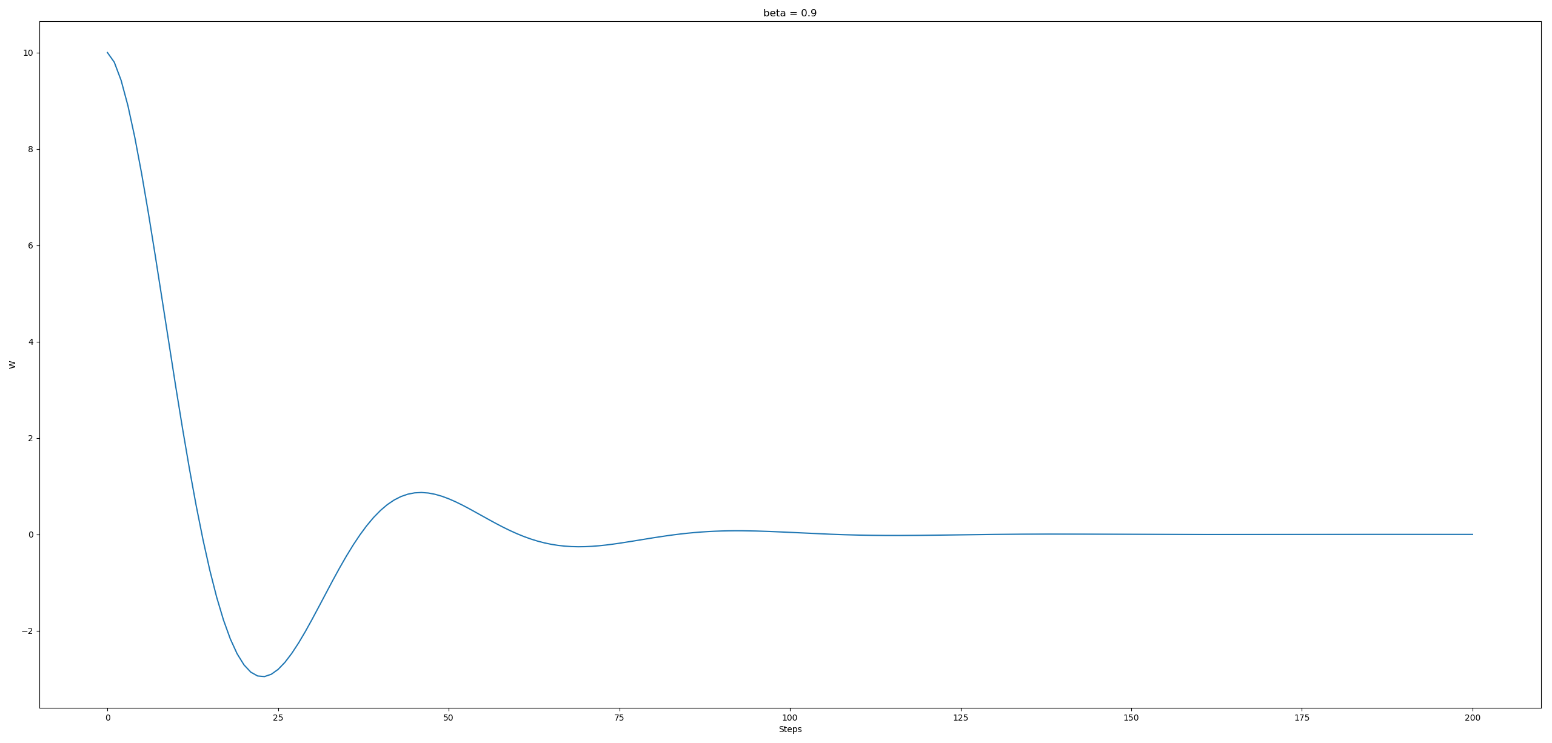
1. **Confusion matrix for Multinomial Naive Bayes**

As the above confusion matrix shows, **16th class and 20th class** are most confused.

1. **Training SVM with SGD**

**2.1 SGD With Momentum:**

Beta = 0.0



Beta = 0.9

**2.2 Training SVM:**

***For detail, please visit q2.py***

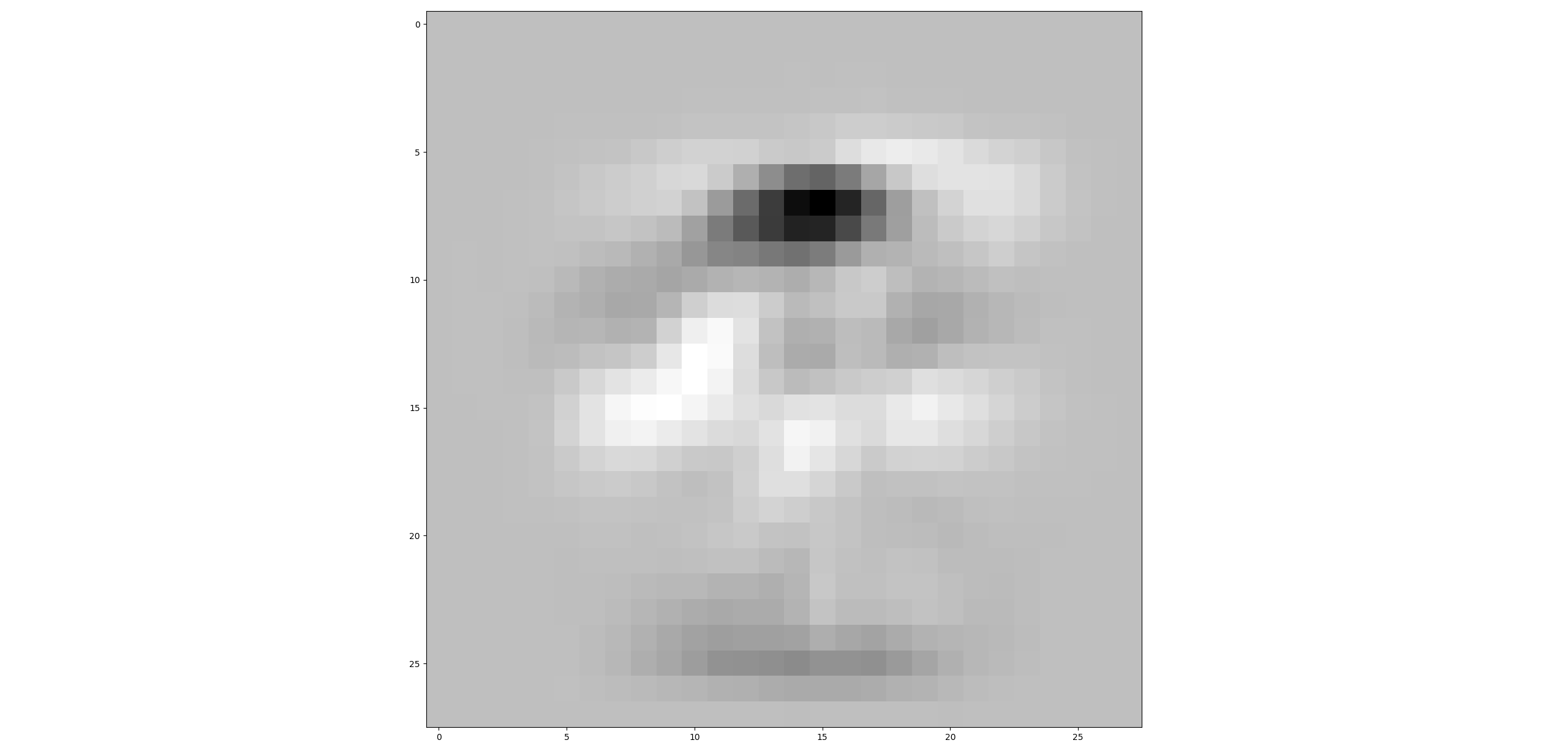
**2.3 Apply on 4-vs-9 digits on MNIST:**

1. model use β = 0:

The training loss: 0.342191688002

The test loss: 0.337671534146

Training accuracy = 0.9329705215419501

Test accuracy = 0.9328980776206021

1. model use β = 0.1:

The training loss: 0.342207800632

The test loss: 0.337795184641

Training accuracy = 0.9340589569160997

Test accuracy = 0.9339862169024302

