Lista 3

Aluno: Luis Vitor Pedreira Iten Zerkowski - 9837201
Professora: Nina Hirata
Disciplina: Introdução ao aprendizado de máquina (MAC0460)

Part I

1.

A polynomial regression problem can be solved basically in the same way as a linear regression can be solved. We could, for example, try to minimize the mean squared error between our curve and the true data values, which is a typical technique for linear regression as well. The main difference between these regressions does not reside on the approach to the problem, but in the optimization problem itself. Using the same techniques as before - SGD and the direct matricial solution to the problem -, but with a bigger amount of weights, we can find the best curve without much difficulty.

2.

The logistic regression and the SVM algorithms both try to find a good linear separator to subdivide data points by their labels. Although they relate to each other in their purpose, they do this process with completely different approaches. The logistic regression model tries to maximize log likelihood, therefore it tries to maximize the probability of the data points being drawn according to our separator. The SVM model, on the other hand, tries to maximize the margin between points that reside on the very "limit" of separation between classes. And not only do they differ from each other on the linear separation approach, but also on the data preprocessing. When using the SVM model, one typically makes use of the kernel trick, non-linear transformations of the data to a space where the points are linearly separable.

Now that we have already discussed these methods main differences, let us visualize some cases when it gets a little more observable that SVM and logistic regression can generate different linear separators:

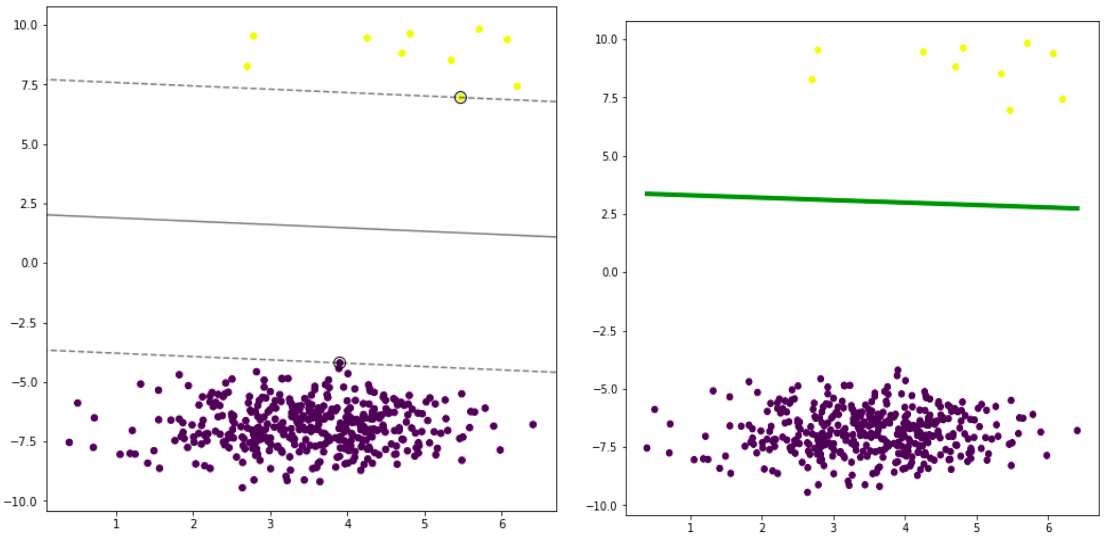

Figure 1: Illustrative image of different separators generated by SVM (left) and logistic regression (right) models on the same dataset. It is possible to observe the logistic regression linear separator went a little further into the yellow points trying to maximize the probability of the purple points - the majority of the points - being indeed classified as purple. The SVM separator, on the other hand, stood right into the middle of the support vectors.

3.

They are both complex and robust methods that can lead us to great results in classification tasks, but they work in completely different ways. The neural networks train several layers of non-linear combinations of features to reach a final non-linear separator. The SVM model, on the other hand, uses non-linear transformation to the data so it can try to linearly separate the data in this new space.

Since they use completely different approaches, they both have their pros and cons. For the neural network, we can get a little more creative when modeling, making use of completely different architectures that lead to various results and even making it possible for us to solve problems other than classification ones. Although this is a very powerful technique, it is pretty hard to fully understand what is happening in each layer and even harder to understand what is being learned in each neuron, which can make the learning process kind of a black box.

Now for the SVM, we can not get much creative when building the model, so we are limited to creative modeling the problem to solve. Because of the natural difficulty of robust modeling a complex task, it is typically the case that neural networks and their variety of architectures perform better than SVMs. Even though this could be a big problem, when we are able to overcome this issue it is much more intuitive to understand what SVMs are doing and how they are fitting the data. It is also easier to theoretically study them, which makes SVMs a little closer to the PAC-Learning field and makes it possible to try to formalize the approach - studying sample complexity and accuracy guarantees, for example.

4.

The validation set is a subset of the original training set used to tune hyperparameters, for regularization purposes, or to compare models. As we do not fit our models to this set initially, It works as a test set for a brief period of time, during the training of multiple instances of the model process. After choosing our model and its parameters, which already makes the validation set a biased score estimator, we can train the model one more time with a bigger amount of data, the training set union the validation set. The test set, on the other hand, is meant to be an unbiased estimate of our performance metrics, therefore it is only accessed by the model on the very end of the typical machine learning pipeline. Since we haven't done any kind of model selection, regularization or hyperparameters tuning with it, the test set can be considered free of data leakages and, if randomly drawn from the same distribution we are drawing our training data, it can one for all give us a real and more reliable estimate of our model performance.

5.

Overfitting is the phenomenon of fitting our model too much on the training set, probably making our model poor in terms of generalization. We always want to avoid overfitting when trying to build a good predictor, therefore it is important to make use of several techniques to detect that it happened and to prevent it.

The most classic way to observe overfitting is by making use of a validation split - use a small percentage of the training set as a validation set - and measuring the cost function you are trying to minimize on this little validation set. When the training set loss continues to decrease but the validation set loss starts to increase, you are probably in the beginning of an overfitting process. The image below shows exactly this phenomenon:

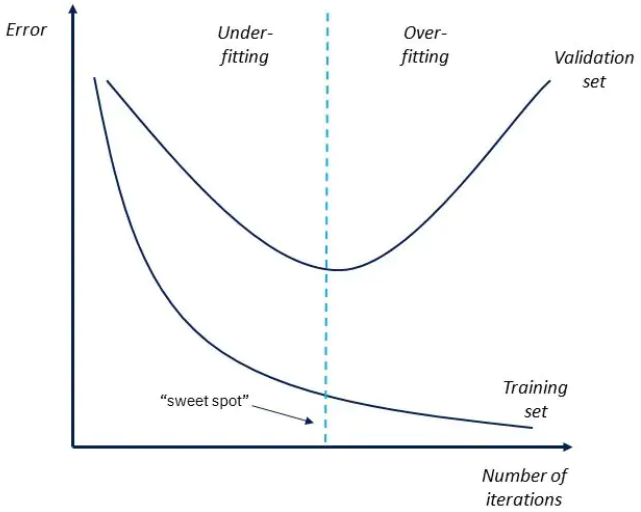

Figure 2: Illustrative image of a model overfitting during the training process. The sweet spot indicated in the image will be discussed in the next paragraph when talking about early stopping regularization method.

The methods used to prevent overfitting are called regularization methods and there are many of them now, since overfitting turns out to be a major issue when training a model. The first, and probably most intuitive one, we discuss here is the early stopping method. As its name already indicates, it consists of stopping the training process right when detecting the start of overfitting. Another famous and easy to explain one is the dropout. It is typically used on neural networks and it consists of dropping some features - deactivating some neurons by making their weights zero - to prevent the model from getting too attached to certain features, forcing it to generalize a little more. There are many other techniques in addition to these two, but there is no need to list them all here.

Part II

6.

1. 17/20 (85%)
2. I think there are several topics that I feel like I've learnt a lot within the course. I could revisit and learn a lot from the mathematical and theoretical part of machine learning section and I also could dive a little deeper into neural networks and convolutional neural networks. I feel like the joint work of the professor with professor Mostafa made topics like Optimization of a cost function, Linear regression, its analytic solution, solution based on gradient descent, Logistic regression, maximization of the likelihood function, Hoeffding inequality, dichotomies, VC bound, Bias-variance theory and SVM, linearly separable case and non-linearly separable case much easier to understand. I also feel the professor's mastery of deep learning helped a lot in the understanding process of topics like Perceptron networks and multilayer neural networks and Convolutional neural networks.
   If I had to highlight one of these as the one I learnt the most, I would say it is SVM. I now understand what is the purpose of the model and how it works together with the kernel trick to generate non-linear separators. I'm also able to understand the margin behavior and the error associated with it now.
3. I think the two topics I learnt the least were Discriminative approaches × Generative approaches and Clustering, dimensionality reduction because we mentioned them only in the last part of the course and for a very small amount of time during classes, so I didn't have time to get used to them, study them, or work with them.
4. I would attribute exactly 8.5 to my learning. I think I was able to understand the most important and fundamental concepts about machine learning and even work a bit with them during the EPs, but I also think I was not able to perfectly understand a few last topics of the course, as mentioned before.

7.

1. 17/20 (85%). I attended almost all the online classes and watched every video of the Caltech playlist on youtube. I also studied Mostafa's book and consulted a lot of extra

material. I did not study much of Generative approaches, Quadratic optimization problems and Viola-Jones, though.

2. I’ve done every QT, list and EP completely and submitted them without delay.
3. Considering attendance frequency, I would say 100%.

8.

I think I performed quite well on the course. I studied a lot and understood almost all the topics listed, I attended almost every class, I participated quite a lot in class - this was a little easier for me, since I’m a talkative and curious person -, and I also think I’ve worked hard and performed well on the tasks. I would say my overall performance was 100%, but it really depends on what is expected from the teacher. If it is sufficient to work hard and understand almost every listed topic, but not literally all of them, then I think my overall score is indeed a 100%.