1a. There is no linear boundaries for a voted perceptron. If you are given a sample of integers ranging from -1,0, 1 and trying to find which is positive or negative you can find the weight of both vectors by determining that if it is below 0 it is negative and above is positive.

1b. There are linear boundaries for an average perceptron. You have a linear boundary since you can find the average of two different samples.

2. When you change the algorithm to work that way you must adjust your weights to a higher number when your examples fail. When you use the prediction algorithm you need to include the h(t) to the equation.

3. One way to adjust to find a better accuracy is to check for overfitting. You need to check the training and testing accuracy. If your training is higher than the testing then your accuracy is overfitted and you need to adjust your algorithm by adjusting the learning rate to better match your data.

4.

To begin we will have a weight set to 1. For each applicant we will compare against another and do something along the line of:

For each feature vector:

If k > l:

Increment weight

If k == l:

No change

If k < l:

Decrement weight

We do this because if K is a better candidate than L then we need to update the weight to show importance. And if K is worse than L we decrement the vector meaning K is less important.

5.

You would then take the summation of each one then find the dot product.

6.

When I begin reading the paper, I really like the way it started off by explaining what learning really means in Machine Learning. It stated learning = representation + evaluation + optimization. A key part I took away from representation was that a classifier must be represented in some sort of formal language. A key takeaway from evaluation I learned was that it is needed to have functions that can distinguish between good and bad classifiers. The last part to the equation, optimization, taught me that we need to choose a correct learning rate to make sure our program stays efficient and produces accurate results.

Another major topic talked about is making sure that your classifier can learn beyond training examples. Some good advice given said that it is important if you hire someone to make a perceptron, to make sure to test it with data that was been used before to make sure the results are randomized but real learned data and accuracy.

An issue that is ran into is called overfitting. It happens when there is not enough data and is insufficient to make a proper classifier. What happens with overfitting is it will run the risk of making up parts of the classifier i.e., hallucinating. It happens when the training is 100% accurate but the testing is 50% accurate. The way to fix this overfitting is to watch over your weights and bias to make sure they are being properly maintained as the algorithm is run.

A big problem that seems to happen after overfitting is dimensionality. It means as the generalizing becomes exponentially harder as the number of features grow. It becomes even more of an issue when dealing with the nearest neighbor problem. As things get more spread out it becomes harder and less efficient to account for everything. There was some effect called “blessing of non-uniformity” where in most examples there is not a huge spread but seems to be a more uniform cluster of data.

A key point that was made that I will be remember is that some machine learning projects succeed while others fail. The main difference that determines success and failure is the factor of features used. A machine can learn easily if the features are independent. It becomes difficult when the function of features becomes complex, it makes it harder to learn from. It was said this is where most of the effort in machine learning goes into. It makes sure the feature used will be the best for the algorithm.

Another good topic discussed was more data seems to beat a clever algorithm. There was an example given where you have what seems to be a perfect algorithm but when given new data the classifiers aren’t accurate enough. The two choices you can make are design a better algorithm or gather more data. It was said people tend to want to gather more data for their algorithms. So, there is a rule of thumb saying a dumber algorithm with more data beats a cleverer algorithm with less data. It was also mentioned that it is good to learn from many models not just one. It used to be people had their favorite go to machine learning algorithms. But nowadays it is good to learn from many different models and take the best from each.

Overall, from this reading it really summarized how to make efficient classifiers, choose the correct data, and what to do with the data.