For the final project I have chosen to experiment with PAC learning theory. In particular, I have chosen three different classifiers, Perceptron, Neural Network (NN), and Support Vector Machine (SVM). By varying the complexity of these classifiers and the number of training example, I demonstrated that PAC learning theory holds for the experiments I conducted, and the general intuition from PAC learning are practical guides to pattern recognition tasks.

1 Introduction

In this project, I have chosen to experiment with the PAC learning theory. From the PAC learning theory, it provided the framework that learning is actually feasible, and has a general bound. From these bound, there are several useful properties, explained in the following sections. These properties can be a guide when choosing what classifier to use for pattern recognition tasks. I also tried to answer the question whether if classifier needs to be deep like a NN or a shallow architecture, such as SVM, is enough.

2 Learning Theory and Classifiers

2.1 PAC Learning Framework and Intuition

From PAC Learning there is the following learning bound

\[
1 - \delta \text{ for all } h \in H \left| \frac{(x,y) \in S: h(x) = y}{|S|} - \Pr_{(x,y) \in D}(h(x) = y) \right| < c \sqrt{\frac{\text{VCdim}(H) + \ln(\frac{1}{\delta})}{|S|}}
\]

As can be seen from the learning bound, to get the tightest bound, one want to choose a hypothesis space, \( H \), with the smallest VC dimension, and use as much training example as possible. However, choosing the smallest hypothesis space may cause that even the best hypothesis in \( H \) is far from the optimal and give poor performance. Further large hypothesis spaces with high VC-dimension tend to leads to over fitting. Therefore, model selection is a very important factor that determines how well a classifier will perform.

2.2 Shattering & VC Dimension

The learning bound depends on the VC dimension. The larger the VC dim, the more expressive the set of the hypothesis is. The definition of VC dimension is defined in this section [4].
Definition 2.1. A set of hypothesis $H$ is said to shatter a set of tokens if, for all assignment of labels to those tokens, there exists a hypothesis that can give those labels to the set of tokens.

Definition 2.2. VC dimension of $H$ over instances space $X$ is the size of the largest finite subset of $X$ that can be shattered by $H$.

2.3 Perceptron

The perceptron has parameters $w \in \mathbb{R}^{n+1}$ and makes the prediction as $\hat{y} = \text{sign}(w^T x)$. The loss function for perceptron is defined as

$$J(w) = \sum_{i=1}^{n} \max(0, 1 - y^{(i)} w^T x^{(i)})$$

then with stochastic gradient descent for learning the weights the perceptron learning algorithm is defined as follows [2]

Algorithm 1 Perceptron Algorithm

1: Initialize $w$ as 0 vector, and a learning rate $\eta$
2: repeat
3: for tokens $(x^{(i)}, y^{(i)}) \in S$ do
4: $\hat{y} \leftarrow w^T x^{(i)}$
5: if $\hat{y} \neq y^{(i)}$ then $w \leftarrow \eta y^{(i)} x^{(i)}$
6: else $w \leftarrow w$
7: until convergence

The perceptron algorithm has a mistake bound as follows

Theorem 2. Suppose there exists a $||u|| = 1$, and $\gamma > 0$, such that $\forall i \ y^{(i)} (w^T x^{(i)}) \geq \gamma$, and $R = \max(||x^{(i)}||)$. Then, the number of mistakes made by the perceptron is $\leq \left( \frac{R \gamma}{\eta^2} \right)^2$.

From the above mistake bound, it means that if the data is linearly separable, then the perceptron algorithm can learn and make no mistake on it. This is used as a general guideline to choose the number of training example when experimenting with PAC learning.

2.4 Support Vector Machine (SVM)

SVM are basically like a perceptron, with the same parameter of $w \in \mathbb{R}^{n+1}$ and makes the prediction as $\hat{y} = \text{sign}(w^T x)$. But different from a perceptron, SVM has the loss function of

$$J(w) = \frac{1}{2} ||w|| \text{ such that } y^{(i)} (w^T x^{(i)}) \geq 1, \forall (x_i, y_i) \in S.$$ 

The reason for this loss function is that SVM tries to maximize the margin, $\frac{2}{||w||}$ of the linear separator. The above loss function can be relaxed into a Soft SVM (allowing examples in the margin) by introducing a slack variable. Then the loss function becomes $J(w) = \frac{1}{2} ||w|| + C \sum_i \max(0, 1 - y^{(i)} w^T x^{(i)})$ The variable $C$ controls the trade off between the cost and regularizer. The larger the C, soft-SVM behaves more like a hard-SVM. [3,5]

2.5 Neural Network (NN)

A Neural Network is a multilayer perceptron. Basically Neural Network stacks these multilayer perceptrons. Each activation $a^{(n)} = b + w^T x^{(n)}$ is a linear classifier, the activation is passed through a nonlinearity of $g(a) = \tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$. Stacking several of these perceptrons together, will form a neural network. Denote the output from the last layer as $\hat{y}^{(n)}$, then the loss function is defined as $\sum_n (\hat{y}^{(n)} - y^{(n)})^2$, the mean square error. [2] NN can be train using the standard back propagation gradient descent algorithm.
3 VC Dimension of Each Classifier

3.1 VC Dimension For Perceptron

Theorem 3. (Wenocur and Dudley): Let $H_n$ be linear threshold hypothesis from $\mathbb{R}^n$ to 0, 1, then $H_n$ has VC-dimension $n + 1$.

Because perceptron has a linear threshold hypothesis, then the VC-dimension of perceptron is $n + 1$. [1]

3.2 VC Dimension for SVM

One would expect that SVM and Perceptron to have the same VC dimension as they are both linear threshold hypothesis. However that is not the case, according to the following theorem SVM has the VC dimension of $\min(\lceil \frac{r^2}{\gamma^2} \rceil, n) + 1$. This can be understood by visualizing the concept of margin as a “thick” linear separator, and thus it can shatter less points. [1]

Theorem 4. (Vapnik): $H_n$ is the space of all linear classifier in $\mathbb{R}^n$ that separates the training data with margin at least $\gamma$, then $VC(H_n) \leq \min(\lceil \frac{r^2}{\gamma^2} \rceil, n) + 1$

3.3 VC Dimension for NN

According to the following theorem, the VC dimension of a NN is $wlog(w)$, given the NN has $w$ number of weights. [1]

Theorem 5. (Cover): Neural Network with $w$ weights that consists of linear thresholds, as the VC dimension of $O(wlog(w))$

3.4 Analysis of VC Dimension

As can be seen from the previous section. SVM can have a hypothesis size that is at most as descriptive as the Perceptron. By changing the $C$ in a soft-SVM, on can control the VC dimension of the SVM. Finally NN can have much larger VC dimension depending on the number of weights compare to SVM and Perceptron.

4 Experiment Setup and Results

4.1 Training Data Generation & Experiment Setup

For the experiments I have generated datasets that are linearly separable and non-linearly separable, with and without noise; dataset figures shown below. These datasets are used to train the three classifiers mentioned in previous sections, and tested on how well the classifier performs.

Figure 1: Linearly separable dataset
4.2 Linearly Separable Data Set (Small Training Size) Results

As can be seen from the learning bound, both the VC dimension and training set’s size affects the bound. Therefore, to observe the affect of the VC dimension, thus I tried to choose a small training set. By a small training set, I choose the training set size that is equal to the mistake bound of the Perceptron. As the Perceptron can learn the hypothesis in $H$ in worst case of the mistake bound number of examples. In this case the training size is 20, and testing size is 1000.
Table 1: Perceptron Results

<table>
<thead>
<tr>
<th>Perceptron Training Error (%)</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perceptron Testing Error (%)</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table 2: Best SVM Results

| SVM Training Error (%) | 0 |
| SVM Testing Error (%)  | 0.1|

Table 3: Best NN Results

| NN Training Error (%) | 0 |
| NN Testing Error (%)  | 1.3|

Figure 5: Non-linearly separable dataset + noise

Figure 6: Train Error Rate vs. Number of Hidden Nodes
4.3 Linearly Separable Data Set (Large Training Size) Results

For the Large Training Size Case, 1000 examples, for SVM and NN, all configurations basically achieve close to 0% error, so plots are not meaningful, and thus not included in the report.

<table>
<thead>
<tr>
<th>Perceptron Training Error (%)</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perceptron Testing Error (%)</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Table 4: Perceptron Results

<table>
<thead>
<tr>
<th>SVM Training Error (%)</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM Testing Error (%)</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Table 5: Best SVM Results

<table>
<thead>
<tr>
<th>NN Training Error (%)</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN Testing Error (%)</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Table 6: Best NN Results

4.4 Linearly Separable Data Set (Large Training Size) + Noise Results

The training and testing set was added with Gaussian noise with mean of 0.05. Note that the data was generate with Gaussian distribution with mean of 0.5.

<table>
<thead>
<tr>
<th>Perceptron Training Error (%)</th>
<th>9.6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perceptron Testing Error (%)</td>
<td>8.9</td>
</tr>
</tbody>
</table>

Table 7: Perceptron Results
4.5 Not Linearly Separable Data (Large Data Set) Set Results

Table 10: Perceptron Results

<table>
<thead>
<tr>
<th>Perceptron Training Error (%)</th>
<th>49</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perceptron Testing Error (%)</td>
<td>49</td>
</tr>
</tbody>
</table>

Table 11: Best SVM Results

<table>
<thead>
<tr>
<th>SVM Training Error (%)</th>
<th>50.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM Testing Error (%)</td>
<td>49.1</td>
</tr>
</tbody>
</table>

Table 12: Best NN Results

<table>
<thead>
<tr>
<th>NN Training Error (%)</th>
<th>0.25</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN Testing Error (%)</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table 13: Best Kernel SVM Results

<table>
<thead>
<tr>
<th>Kernel SVM Training Error (%)</th>
<th>0.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel SVM Testing Error (%)</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Figure 8: Train Error Rate vs. Number of Hidden Nodes
5 Discussion

5.1 Discussion Linearly Separable Dataset (Small Training Size)

From the experiment of linearly separable dataset with small training size, the importance of VC dimension can be observed. From Table 1,2,3, observe that SVM have the lowest test error, and NN has the highest test error. This is consistent with PAC learning bound. Because we know the target hypothesis is in $H$, and when the number of training example is small, the dominant term should be the VC dimension. From the previous analysis, we can see that the SVM has the smallest VC dimension, then the Perceptron, and lastly NN. Furthermore from Figure 6 and 7, one can see that the two hidden layer NN performs slightly worse, and there is roughly a “trend” of increasing error as the number of hidden node increases; again consistent with the PAC learning bound.

5.2 Discussion Linearly Separable Dataset (Large Training Size)

For both with and without noise, when the training set is large, all of the classifiers perform about the same at nearly 0% error, from Table 4,5,6,7,8,9. This is consistent with the PAC learning bound as the number of example increases, the tighter the bound.

5.3 Discussion Non-linearly Separable Dataset (Large Training Size)

The purpose of the non-linearly separable dataset is to show that if the hypothesis space does not contain the target hypothesis, then no matter how much training data is given, the target function cannot be learned. This is exactly the result observe from the experiment. Both Perceptron and SVM only learn linearly separable hypothesis, therefore they performed poorly on this dataset. On the other hand, NN's hypothesis space contains the target function, except for the ones with 1 hidden node, therefore consistent with the result in Figure 8 and 9; the 1 hidden node NN performs poorly.

Next, I also ran a degree two polynomial kernel SVM on this dataset, and it performs slightly better than the NN. This is again consistent with the PAC learning, as by using a polynomial kernel, the hypothesis space now contains the target hypothesis.

6 Conclusion

For this project I have experiment with the PAC learning bound and have shown empirical results that are consistent with the learning bound. Therefore, for pattern recognition tasks, one should choose a hypothesis space $H$ that contains the target hypothesis with a VC dimension as small as possible and use as much training data as possible. However, choosing such hypothesis space $H$ is a hard task, as the target hypothesis is unknown. Therefore, one possible solution is to choose a
$H$ with high VC dimension then apply regularization to reduce the VC dimension; illustrated in the experiments using SVM and Perceptron. Therefore, from PAC learning and the experiments seems like NN does not have to be deep, shallow models like SVM should be able to perform just as well with the correct kernel, as it is all about how to choose a $H$ that contains the target hypothesis and still having a small VC dimension so learning is possible.

References


[4] Andrew Moore’s VC dimension tutorial