ML in Practice

Tirgul 7 Part II
ML in Practice

• Preprocessing data
• Machine Learning Evaluation Metrics
Preprocessing data

• Machine Learning algorithms tend to learn more effectively if the inputs $x$ and targets $y$ are prepared for analysis before the network is trained.
A Basic Example: the target values

• HW:
  • MNIST – Handwritten digit database
  • Use Perceptron to predict the number written in the image

\[ y = 8 \quad y_{\text{new}} = 1 \]

\[ y = 1 \quad y_{\text{new}} = -1 \]
Scaling Input Data

- Housing E.g.: The features are all in different scales

![Graph showing the relationship between size in square feet and number of bedrooms.](image)

E.g.:
- $x_1$ – size (0-2000 feet$^2$)
- $x_2$ - number of bedrooms (1-12)
Scaling Input Data

• Standardizing the range of the features is called Feature Scaling.

• **Motivation**: Since the range of values of data varies widely, some Machine Learning algorithms will not work properly without feature scaling.
  
  • E.g.: classifiers that calculate distance between two points
    
    • Use **Euclidean** distance
    
    • If one feature has a broad range of values, the distance will be governed by this specific feature.
    
    • With feature scaling, every feature contributes approx. proportionately to the final distance.

  • Gradient Descent converges faster with feature scaling.

---

Euclidean Distance:

\[ d(a, b) = \sqrt{(a_1 - b_1)^2 + (a_2 + b_2)^2 + \ldots + (a_n + b_n)^2} \]
Feature Scaling

• Housing E.g.: Idea: make sure features are on similar scale

E.g.:
• $x_1$ – size (0-2000 feet²)
• $x_2$ - number of bedrooms (1-12)

\[
\begin{align*}
x_1 &= \frac{\text{size}}{2000} \\
x_2 &= \frac{\#\text{bedrooms}}{12}
\end{align*}
\]

$0 \leq x_1 \leq 1 \quad 0 \leq x_2 \leq 1
Feature Scaling Methods (1)

• Rescaling:

\[ x' = \frac{x - \text{min}(x)}{\text{max}(x) - \text{min}(x)} \in [0,1] \]

- \(\text{min}(x)\) – the minimal value of the feature \(x\) in the dataset
- \(\text{max}(x)\) – the maximal value of the feature \(x\) in the dataset
Feature Scaling Methods (2)

• Standardization

\[ x' = \frac{x - \bar{x}}{\sigma} \]

• Every feature in \( x' \) has zero-mean and unit variance
• This method is widely used for normalization in many machine learning algorithms.
Feature Scaling Methods (2)

• Standardization: implementation
  • Very easy to perform using NumPy:
    • np.mean()
    • np.var()
  • Example:

\[
\mathbf{x}' = \frac{\mathbf{x} - \bar{x}}{\sigma}
\]

Notice: axis=0 sums down the columns and axis=1 sums across the rows

```python
>>> data = (data - data.mean(axis=0))/data.std(axis=0)
```
Feature Scaling Methods (3)

• Scaling to Unit length

\[ x' = \frac{x}{\|x\|} \]

• Dividing each component by the Euclidean length of the vector.
• The complete vector now has length 1.
Feature Scaling

• If we normalize the training and test sets separately:
  • A datapoint that is in both sets will end up being different in the two
  • Since the mean/variance or min/max are probably different in the two sets.

• Therefore:
  • Normalize the dataset before splitting to train and test
Machine Learning Evaluation Metrics

• Confusion Matrix
• Accuracy Metrics
Confusion Matrix

• For classification problems
• Suppose we have 3 classes: $C_1$, $C_2$, and $C_3$
• Make a square matrix that contains all classes in both horizontal and vertical directions.
• Element $(i, j)$ shows how many inputs had:
  • $y = C_i$
  • $\hat{y} = C_j$
Confusion Matrix

• Example:

<table>
<thead>
<tr>
<th></th>
<th>C₁</th>
<th>C₂</th>
<th>C₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>C₁</td>
<td>5</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>C₂</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>C₃</td>
<td>2</td>
<td>0</td>
<td>4</td>
</tr>
</tbody>
</table>

• Anything on the *leading diagonal* is a correct answer.
Accuracy Metrics

• Example:

<table>
<thead>
<tr>
<th></th>
<th>C₁</th>
<th>C₂</th>
<th>C₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>C₁</td>
<td>5</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>C₂</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>C₃</td>
<td>2</td>
<td>0</td>
<td>4</td>
</tr>
</tbody>
</table>

• **Accuracy:** divide sum of elements on leading diagonal by the sum of all elements of the matrix.

\[
\text{accuracy} = \frac{5 + 4 + 4}{5 + 1 + 1 + 4 + 1 + 2 + 4} = \frac{13}{18} = 0.72
\]
Accuracy Metrics

As in the confusion matrix, here too the entries on the leading diagonal are correct, and those off the diagonal are wrong.
Accuracy Metrics

• Blue background: examples that are positive \((y = 1)\).
• Pink background: examples that are negative \((y = -1)\).
Accuracy Metrics

- **Green background:** examples that were predicted positive ($\hat{y} = 1$).
- **Yellow background:** examples that were predicted negative ($\hat{y} = -1$).

<table>
<thead>
<tr>
<th>True Positives</th>
<th>False Positives</th>
</tr>
</thead>
<tbody>
<tr>
<td>False Negatives</td>
<td>True Negatives</td>
</tr>
</tbody>
</table>
Accuracy Metrics

<table>
<thead>
<tr>
<th>True Positives (TP)</th>
<th>False Positives (FP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(predicted positive and it was true)</td>
<td>(predicted positive and it was false)</td>
</tr>
<tr>
<td>False Negatives (FN)</td>
<td>True Negatives (TN)</td>
</tr>
<tr>
<td>(predicted negative and it was false)</td>
<td>(predicted negative and it was true)</td>
</tr>
</tbody>
</table>

\[
\text{Accuracy} = \frac{\#\text{True Positives} + \#\text{True Negatives}}{\#TP + \#FP + \#FN + \#TN}
\]
Accuracy Metrics

• We always assumed that are dataset is balanced
  • Approx. same amount of examples from every class

• What if it isn’t?

• E.g. : Spam Classifier.
  • 90% of emails are non-spam.
  • 10% of emails are spam.
  • $h(x) = 0 \rightarrow$ has 90% accuracy!
Accuracy Metrics

- **Precision** = \( \frac{\#TP}{\#TP + \#FP} = \frac{\#TP}{\text{predicted pos}} \)

- **Recall** = \( \frac{\#TP}{\#TP + \#FN} = \frac{\#TP}{\text{actual pos}} \)

- A classifier that always outputs \( \hat{y} = 0 \):
  - \( \#TP = 0 \)
  - \( \text{Precision} = \text{Recall} = 0 \)
Accuracy Metrics

• Which algorithm is better?

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm 1</td>
<td>0.5</td>
<td>0.4</td>
</tr>
<tr>
<td>Algorithm 2</td>
<td>0.7</td>
<td>0.1</td>
</tr>
<tr>
<td>Algorithm 3</td>
<td>0.02</td>
<td>1</td>
</tr>
</tbody>
</table>

Precision = \( \frac{\#TP}{\#predicted \ pos} \)

Recall = \( \frac{\#TP}{\#actual \ pos} \)

Could be that I predicted \( \hat{y} = 1 \) most of the time...
Accuracy Metrics

• Precision and Recall can be combined to a single measure:

\[ F_1 = 2 \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \]

• In order for the \( F_1 \) score to be large, both precision and recall have to be large.
### Accuracy Metrics

\[ F_1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \]

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Precision</th>
<th>Recall</th>
<th>(F_1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm 1</td>
<td>0.5</td>
<td>0.4</td>
<td>0.444</td>
</tr>
<tr>
<td>Algorithm 2</td>
<td>0.7</td>
<td>0.1</td>
<td>0.175</td>
</tr>
<tr>
<td>Algorithm 3</td>
<td>0.02</td>
<td>1</td>
<td>0.0392</td>
</tr>
</tbody>
</table>

- If \(\text{Precision} = 0\) and \(\text{Recall} = 0\) \(\rightarrow F_1 = 0\)
- If \(\text{Precision} = 1\) and \(\text{Recall} = 1\) \(\rightarrow F_1 = 1\)
ROC Curve

• ROC curve: a graphical plot that illustrates the performance of a binary classifier system as its discrimination threshold is varied.

• Every point on ROC curve corresponds to specific threshold value.
ROC Curve

- The curve is created by plotting the **true positive rate** (TPR) against the **false positive rate** (FPR) at various threshold settings.

\[
TPR = \frac{TP}{P} = \frac{TP}{TP + FN}
\]

\[
FPR = \frac{FP}{N} = \frac{FP}{FP + TN}
\]
ROC Curve

• **AUC:**
  - Area Under the ROC Curve

• **Goal:**
  - Maximize AUC on unseen data