Concept Inventory

Here is a list of almost all the topics people wrote on their concept inventory. I have grouped them based on how I would categorize them, but this does not mean it is wrong to put them in different categories. Most topics appear in multiple places so this list would be long if we included every instance, but this is just how I think about them in my “ML Landscape”. This is not meant to be an exhaustive list of topics, just big concepts that were listed on students’ reflections.

<table>
<thead>
<tr>
<th>Regression</th>
<th>Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Linear Regression</td>
<td>• Binary/multi-class classification</td>
</tr>
<tr>
<td>• Polynomial Regression (linear regression with polynomial features)</td>
<td>– Sentiment analysis</td>
</tr>
<tr>
<td>• Quality Metric: RSS/MSE/RMSE</td>
<td>– Spam detection</td>
</tr>
<tr>
<td>• Coefficients/weights/parameters</td>
<td>– Medical Diagnoses</td>
</tr>
<tr>
<td>• Regularization</td>
<td>• Majority class classifier</td>
</tr>
<tr>
<td>– Ridge (L2 Penalty)</td>
<td>• Random Classifier</td>
</tr>
<tr>
<td>– LASSO (L1 Penalty)</td>
<td>• Score Classifier</td>
</tr>
<tr>
<td>– ElasticNet (L1 + L2)</td>
<td>• Sigmoid Functions (Logistic function)</td>
</tr>
<tr>
<td>• Local Methods</td>
<td>• Logistic regression</td>
</tr>
<tr>
<td>– k-NN regression</td>
<td>• Decision Trees</td>
</tr>
<tr>
<td>– Weighted k-NN regression</td>
<td>• Assessing Performance</td>
</tr>
<tr>
<td>– Kernel Regression</td>
<td>– Classification Error/Accuracy</td>
</tr>
<tr>
<td>* kernels (boxcar, Gaussian)</td>
<td>– True Positive, True Negative, False Positive, False Negative</td>
</tr>
<tr>
<td>* bandwidth</td>
<td>– Confusion matrix</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Document Retrieval
- **k-NN Retrieval**
- **Clustering**
- **k-means/k-means+++**
- **Hierarchical Clustering**
  - Agglomerative
    * Single linkage
    * Complete linkage
  - Divisive
    * Recursive k-means
  - Dendrograms
- **Similarity/Distance metrics**
  - Euclidean Distance
  - Cosine Distance
  - Manhattan Distance
- **Locality sensitive hashing**
- **Embeddings**
  - Bag of words
  - TF-IDF

### Misc
- **ML Pipeline**
- **Supervised vs. unsupervised learning**
- **Feature vs input vs output**
  - Polynomial features
- **Model Complexity**
- **Overfitting/Underfitting**
- **Bias/Variance trade-off**
- **Hyperparameters**
- **Hyperparameter tuning**
  - Grid search
  - Random search
- **Train/Validation/Test**
- **Cross-Validation**
  - k-fold validation
- **Loss Function/Quality Metric**
- **Gradient Descent/Ascent**
- **Coordinate Descent/Ascent**
- **Maximum Likelihood Estimation (MLE)**
- **Feature engineering**
- **Scaling Features/Normalization**
- **One-hot encoding**
- **Missing data**
- **Curse of Dimensionality**
- **Greedy algorithms**
- **Efficiency \(O(n)\)**
- **Norms**
  - L1 Norm
  - L2 Norm
- **Dot Product/Inner Product**
Convolution

1. Suppose we are given the following image:

\[
\begin{array}{cccc}
1 & 2 & 1 & 3 \\
4 & 3 & 3 & 3 \\
5 & 5 & 4 & 4 \\
1 & 1 & 2 & 4 \\
3 & 2 & 1 & 5 \\
\end{array}
\]

Below, write out the result of doing a convolution with the following kernel. Assume we use no padding and a 1x1 stride.

\[
\begin{array}{ccc}
0 & 1 & 2 \\
0 & 0 & 2 \\
0 & 0 & 1 \\
\end{array}
\]
Neural Networks

1. Consider a neural network where each unit has a weight $w$ and uses the hard threshold activation function defined as

$$ h(x) = \begin{cases} 
1, & w^T x \geq 0 \\
0, & \text{otherwise}
\end{cases} $$

Construct the AND, OR gates with two inputs $X_1$ and $X_2$. Construct the NOT gate with $X_1$.

2. (a) What is an epoch?

(b) What is a batch?

(c) Given the algorithm defined in class

```python
for i in range(num_epochs):
    for batch in batches(training_data):
        preds = model.predict(batch.data)
        diffs = compare(preds, batch.labels)
        model.backprop(diffs)
```

How many iterations of back-prop will we run? (answer as of a function of $\text{num\_epochs}$ and $\text{num\_batches}$)