1 Decision Trees and Beyond

1. Decision Tree Classification with Continuous Attributes

Given the dataset $D_1 = \{x^{(i)}, y\}_{i=1}^N$ where $x^{(i)} \in \mathbb{R}^2$, $y \in \{\text{Yellow, Purple, Green}\}$ as shown in Fig. 1, we wish to learn a decision tree for classifying such points. Provided with a possible tree structure in Fig. 1, what values of $\alpha$, $\beta$ and leaf node predictions could we use to perfectly classify the points? Now, draw the associated decision boundaries on the scatter plot.

![Figure 1: Classification of 2D points, with Decision Tree to fill in](image-url)
Decision Tree Regression with Continuous Attributes
Now instead if we had dataset $D_2 = \{x^{(i)}, y\}_{i=1}^{N}$, where $x^{(i)} \in \mathbb{R}^2, y \in \mathbb{R}$ as shown in Fig. 2, we wish to learn a decision tree for regression on such points. Using the same tree structure and values of $\alpha, \beta$ as before, what values should each leaf node predict to minimize the training Mean Squared Error (MSE) of our regression? Assume each leaf node just predicts a constant.

Figure 2: Regression on 2D points, with Decision Tree to fill in
2 \( k \)-NN

2.1 A Classification Example

Using the figure below, what would you categorize the green circle as with \( k = 3 \)? \( k = 5 \)?

![Figure 3: From wiki](image)

2.2 \( k \)-NN for Regression

You want to predict a continuous variable \( Y \) with a continuous variable \( X \). Having just learned \( k \)-NN, you are super eager to try it out for regression. Given the data below, draw the regression lines (what \( k \)-NN would predict \( Y \) to be for every \( X \) value if it was trained for the given data) for \( k \)-NN regression with \( k = 1 \), weighted \( k = 2 \), and unweighted \( k = 2 \). For weighted \( k = 2 \), take the weighted average of the two nearest points. For unweighted \( k = 2 \), take the unweighted average of the two nearest points. (Note: the points are equidistant along the x-axis)
(a) $k = 1$

(b) weighted $k = 2$

(c) unweighted $k = 2$
3 Linear Regression

3.1 Defining the Objective Function

1. What does an objective function $J(\theta)$ do?

2. What are some properties of this function?

3. What are some examples?

3.2 Solving Linear Regression using Gradient Descent

\[
\begin{array}{c|ccccc}
  & x^{(1)} & x^{(2)} & x^{(3)} & x^{(4)} & x^{(5)} \\
\hline
  x_1 & 1.0 & 2.0 & 3.0 & 4.0 & 5.0 \\
  x_2 & -2.0 & -5.0 & -6.0 & -8.0 & -11.0 \\
  x_3 & 3.0 & 8.0 & 9.0 & 12.0 & 14.0 \\
  y & 2.0 & 4.0 & 7.0 & 8.0 & 11.0 \\
\end{array}
\]

Now, we want to implement the gradient descent method.

Assuming that $\alpha = 0.1$ and $w$ has been initialized to $[0, 0, 0]^T$, perform one iteration of gradient descent:

1. What is the gradient of the objective function, $J(\theta)$, w.r.t $\theta$: $\nabla_{\theta} J(\theta)$

2. How do we carry out the update rule?
4 Perceptron

4.1 Perceptron Mistake Bound Guarantee

If a dataset has margin $\gamma$ and all points inside a ball of radius $R$, then the perceptron makes less than or equal to $(R/\gamma)^2$ mistakes.

![Perceptron Mistake Bound Setup](image)

4.2 Definitions

Margin:

- The margin of example $x$ wrt a linear separator $w$ is the (absolute) distance from $x$ to the plane $w \cdot x = 0$.
- The margin $\gamma_w$ of a set of examples $S$ wrt a linear separator $w$ is the smallest margin over points $x \in S$.
- The margin $\gamma$ of a set of examples $S$ is the maximum $\gamma_w$ over all linear separators $w$.

Linear Separability: For a binary classification problem, a set of examples $S$ is linearly separable if there exists a linear decision boundary that can separate the points.

We say (batch) perceptron algorithm has converged when it stops making mistakes on the training data.

4.3 Theorem: Block, Novikoff

Given dataset $D = (x^{(i)}, y^{(i)})_{i=1}^N$. Suppose:

1. Finite size inputs: $||x^{(i)}|| \leq R$
2. Linearly separable data: $\exists \theta^*$ and $\gamma > 0$ s.t. $||\theta^*|| = 1$ and $y^{(i)}(\theta^* \cdot x^{(i)}) \geq \gamma, \forall i$

Then, the number of mistakes made by the Perceptron algorithm on this dataset is $k \leq (R/\gamma)^2$
Proof:
Part 1: For some $A$, $A \kappa \leq \|\theta^*\|$

Part 2: For some $B$, $\|\theta^*\| \leq B\sqrt{k}$

Part 3: Combine the bounds

Main Takeaway:
5 Summary

5.1 $k$-NN

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
<th>Inductive bias</th>
<th>When to use</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Simple, minimal assumptions made about data distribution</td>
<td>• Becomes slow as dataset grows</td>
<td>• Similar (i.e. nearby) points should have similar labels</td>
<td>• Small dataset</td>
</tr>
<tr>
<td>• No training of parameters</td>
<td>• Requires homogeneous features</td>
<td>• All label dimensions are created equal</td>
<td>• Small dimensionality</td>
</tr>
<tr>
<td>• Can apply to multi-class problems and use different metrics</td>
<td>• Selection of $k$ is tricky</td>
<td></td>
<td>• Data is clean (no missing data)</td>
</tr>
<tr>
<td></td>
<td>• Imbalanced data can lead to misleading results</td>
<td></td>
<td>• Inductive bias is strong for dataset</td>
</tr>
<tr>
<td></td>
<td>• Sensitive to outliers</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.2 Linear regression

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
<th>Inductive bias</th>
<th>When to use</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Easy to understand and train</td>
<td>• Sensitive to noise (other than zero-mean Gaussian noise)</td>
<td>• The relationship between the inputs $x$ and output $y$ is linear. i.e. hypothesis space is Linear Functions</td>
<td>• Most cases (can be extended by adding non-linear feature transformations)</td>
</tr>
<tr>
<td>• Closed form solution</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.3 Decision Tree

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
<th>Inductive bias</th>
<th>When to use</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Easy to understand and interpret</td>
<td>• Tree may grow very large and tend to overfit.</td>
<td>• Prefer the smallest tree consistent w/ the training data (i.e. 0 error rate)</td>
<td>• Most cases. Random forests are widely used in industry.</td>
</tr>
<tr>
<td>• Very fast for inference</td>
<td>• Greedy behaviour may be sub-optimal</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


5.4 Perceptron

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
<th>Inductive bias</th>
<th>When to use</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Easy to understand and works in an online learning setting.</td>
<td>• No guarantees on finding maximum-margin hyperplane (like in SVM), only that you will find a separating hyperplane.</td>
<td>• The binary classes are separable in the feature space by a line.</td>
<td>• The basic perceptron algorithm is not used much anymore, but other variants mentioned in class such as kernel perceptron or structured perceptron may have more success.</td>
</tr>
<tr>
<td>• Provable guarantees on mistakes made if the data is known to be linearly separable (perceptron mistake-bound).</td>
<td>• Output is sensitive to noise in the training data.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>