1 Decision Trees and Beyond

1. Decision Tree Classification with Continuous Attributes
   Given the dataset $D_1 = \{x^{(i)}, y^{(i)}\}_{i=1}^N$ where $x^{(i)} \in \mathbb{R}^2$, $y^{(i)} \in \{\text{Yellow, Purple, Green}\}$ 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)
2. Decision Tree Regression with Continuous Attributes

Now instead if we had dataset $D_2 = \{x^{(i)}, y^{(i)}\}_{i=1}^N$ where $x^{(i)} \in \mathbb{R}^2$, $y^{(i)} \in \mathbb{R}$ 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](image)

3. Choosing a Tree: What might happen if we increased the max-depth of the tree? When predicting on unseen data, would we prefer the depth-2 tree above or a very deep tree?
2 \hspace{1mm} k-NN

2.1 A Classification Example

Using the figure below, what would you categorize the green circle as with $k = 3$? $k = 5$? $k = 4$?

![Figure 3: An example of k-NN on a small dataset; image source from Wikipedia](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 \( k = 2 \)
2.3 \( k \)-NN Decision Boundary

Draw the decision boundaries for the above training dataset given using \( k \)NN algorithm considering \( k = 1 \).
3 Linear Regression

3.1 Defining the Objective Function

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

2. What are some examples?

3. What are some properties of this function?
3.2 Solving Linear Regression using Gradient Descent

\[
\begin{array}{cccccc}
\mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \mathbf{x}^{(3)} & \mathbf{x}^{(4)} & \mathbf{x}^{(5)} \\
x_1 & 1.0 & 2.0 & 3.0 & 4.0 & 5.0 \\
x_2 & -2.0 & -5.0 & -6.0 & -8.0 & -11.0 \\
y & 2.0 & 4.0 & 7.0 & 8.0 & 11.0 \\
\end{array}
\]

Now, we want to implement the gradient descent method.

Assuming that \( \gamma = 0.1 \) and \( \theta \) has been initialized to \([0, 0, 0]^{T}\), perform one iteration of gradient descent:

1. What is the gradient of the objective function \( J(\theta) \) with respect to \( \theta \): \( \nabla_{\theta} J(\theta) \)?

2. How do we carry out the update rule?

3. How could we pick which value of \( \gamma \) to use if we weren’t given the step size?
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.

![Figure 5: 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.

Update Rule: When the $k$-th mistake is made on data point $x^{(i)}$, the parameter update is

$$\theta^{(k+1)} = \theta^{(k)} + y^{(i)} x^{(i)}$$

We say the (batch) perceptron algorithm has converged when it stops making mistakes on the training data.
4.3 Perceptron Mistake Bound: Example

Given dataset $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$, suppose:

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

Then, the number of mistakes $k$ made by the perceptron algorithm on $\mathcal{D}$ is bounded by $(R/\gamma)^2$.

The following table shows a dataset of linearly separable datapoints.

<table>
<thead>
<tr>
<th>x1</th>
<th>x2</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>-1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Assuming that the linear separator with the largest margin is given by:

$$\theta^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0, \text{ where } \theta = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Calculate the theoretical mistake bound for the perceptron.
4.4 Theorem: Block, Novikoff

Given dataset \( \mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^N \), 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 \( k \) made by the perceptron algorithm on \( \mathcal{D} \) is bounded by \( (R/\gamma)^2 \).

**Proof:**

Part 1: For some \( A \), \( Ak \leq \|\theta^{(k+1)}\| \)

Part 2: For some \( B \), \( \|\theta^{(k+1)}\| \leq B\sqrt{k} \)

Part 3: Combine the bounds

Main Takeaway:
## 5 Summary

### 5.1 Decision Tree

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

### 5.2 $k$-NN

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
<th>Inductive bias</th>
<th>When to use</th>
</tr>
</thead>
</table>
| • No training of parameters  
• Can apply to multi-class problems and use different metrics | • Slow for large datasets  
• Must select good $k$  
• Imbalanced data and outliers can lead to misleading results | • Similar (i.e. nearby) points should have similar labels  
• All label dimensions are created equal | • Small dataset  
• Small dimensionality  
• Data is clean (no missing data)  
• Inductive bias is strong for dataset |

### 5.3 Linear regression

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
<th>Inductive bias</th>
<th>When to use</th>
</tr>
</thead>
</table>
| • Easy to understand and train  
• Closed form solution | • Sensitive to noise (other than zero-mean Gaussian noise) | • The true relationship between the inputs and output is linear. | • Most cases (can be extended by adding non-linear feature transformations) |

### 5.4 Perceptron

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
<th>Inductive bias</th>
<th>When to use</th>
</tr>
</thead>
</table>
| • Easy to understand and works for online learning.  
• Provable guarantees on mistakes made for linearly separable data. | • No guarantees on finding best (maximum-margin) hyperplane.  
• Output is sensitive to noise in the training data. | • The binary classes are separable in the feature space by a line. | • Not used much anymore, but variants (kernel perceptron, structured perceptron) may have more success. |