### Simple Training Data Set

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>PlayTennis?</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>

### Decision Trees Review

A Decision tree for f: *(Outlook, Temperature, Humidity, Wind) → PlayTennis? (X₁, X₂, X₃, X₄) → Y*

- Each internal node: test one discrete-valued attribute $X_i$
- Each branch from a node: selects one value for $X_i$
- Each leaf node: predict $Y$ (or $P(Y|X \in \text{leaf})$)
Entropy

**Entropy** $H(X)$ of a random variable $X$

$$H(X) = - \sum_{i=1}^{n} P(X = i) \log_2 P(X = i)$$

Specific conditional entropy $H(X|Y=v)$ of $X$ given $Y=v$:

$$H(X|Y=v) = - \sum_{i=1}^{n} P(X = i|Y=v) \log_2 P(X = i|Y=v)$$

Conditional entropy $H(X|Y)$ of $X$ given $Y$:

$$H(X|Y) = \sum_{v \in \text{values}(Y)} P(Y = v) H(X|Y = v)$$

Mutual information (aka Information Gain) of $X$ and $Y$:

$$I(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

Example in class

- Try to find $IG(Y,X1)$, $IG(Y,X2)$ and $IG(Y,X3)$

```
<table>
<thead>
<tr>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>+</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>+</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>
```

Information Gain is the mutual information between input attribute $A$ and target variable $Y$

Information Gain is the expected reduction in entropy of target variable $Y$ for data sample $S$, due to sorting on variable $A$

$$Gain(S, A) = I_S(A,Y) = H_S(Y) - H_S(Y|A)$$

```
[29+, 35−]  A1=?
     t       f
[22+, 5−]   [8+, 30−]
```

```
[29+, 35−]  A2=?
     t       f
[13+, 33−]   [11+, 2−]
```

Example in class

- $H(Y) = 1$
- $H(Y|X1=1) = -1/3 \log_2(1/3) - 2/3 \log_2(2/3) = 0.92$
- $H(Y|X1=0) = -1\log_2(1) = 0$
- $H(Y|X1) = 3/4 \cdot H(Y|X1=1) + 1/4 \cdot H(Y|X1=0) \sim 0.92$
- $IG(Y,X1) \sim 0.31$
Example in class

- $H(Y) = 1$
- $H(Y|X_2=1) = -1\log_2(1) = 0$
- $H(Y|X_2=0) = -1\log_2(1) = 0$
- $H(Y|X_2) = \frac{1}{2} \cdot H(Y|X_2=1) + \frac{1}{2} \cdot H(Y|X_2=0) = 0$
- $IG(Y,X_2) = 1$

- Pick $X_2$!

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>+</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>+</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>

Example in class

- $H(Y) = 1$
- $H(Y|X_3=1) = -\frac{1}{2}\log_2(\frac{1}{2}) - \frac{1}{2}\log_2(\frac{1}{2}) = 1$
- $H(Y|X_3=0) = -\frac{1}{2}\log_2(\frac{1}{2}) - \frac{1}{2}\log_2(\frac{1}{2}) = 1$
- $H(Y|X_3) = \frac{1}{2} \cdot H(Y|X_3=1) + \frac{1}{2} \cdot H(Y|X_3=0) = 1$
- $IG(Y,X_3) = 0$

- $X_3$ doesn’t help at all at this step

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>+</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>+</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>

Which Tree Should We Output?

- ID3 performs heuristic search through space of decision trees
- It stops at smallest acceptable tree. Why?

Why Prefer Short Hypotheses? (Occam’s Razor)

Argument in favor:
- Fewer short hypotheses than long ones
  - a short hypothesis that fits the data is less likely to be a statistical coincidence
  - highly probable that a sufficiently complex hypothesis will fit the data

Argument opposed:
- Also fewer hypotheses with prime number of nodes and attributes beginning with “Z”
- What’s so special about “short” hypotheses?
Overfitting

Consider a hypothesis $h$ and its
- Error rate over training data: $\text{error}_{\text{train}}(h)$
- True error rate over all data: $\text{error}_{\text{true}}(h)$

We say $h$ overfits the training data if

\[ \text{error}_{\text{true}}(h) > \text{error}_{\text{train}}(h) \]

Amount of overfitting =

\[ \text{error}_{\text{true}}(h) - \text{error}_{\text{train}}(h) \]

Avoiding Overfitting

How can we avoid overfitting?
- stop growing when data split not statistically significant
- grow full tree, then post-prune
Split data into training and validation set

Create tree that classifies training set correctly

Do until further pruning is harmful:

1. Evaluate impact on validation set of pruning each possible node (plus those below it)
2. Greedily remove the one that most improves validation set accuracy

Empirically successful. Widely used in industry.

- human pose recognition in Microsoft kinect
- medical imaging – cortical parcellation
- classify disease from gene expression data

How to train different trees

1. Train on different random subsets of data
2. Randomize the choice of decision nodes
Random Forests

Key idea:
1. learn a collection of many trees
2. classify by taking a weighted vote of the trees

Empirically successful. Widely used in industry.
- human pose recognition in Microsoft Kinect
- medical imaging – cortical parcellation
- classify disease from gene expression data

How to train different trees
- Train on different random subsets of data
- Randomize the choice of decision nodes

more to come

later lecture on boosting

Questions to think about (1)
- Consider target function \( f: (x_1,x_2) \to y \), where \( x_1 \) and \( x_2 \) are real-valued, \( y \) is boolean. What is the set of decision surfaces describable with decision trees that use each attribute at most once?

Questions to think about (2)
- ID3 and C4.5 are heuristic algorithms that search through the space of decision trees. Why not just do an exhaustive search?

Questions to think about (3)
- Why use Information Gain to select attributes in decision trees? What other criteria seem reasonable, and what are the tradeoffs in making this choice?
Inspired by biological neurons

The Perceptron

Dendrites (inputs from other neurons, can be excitatory or inhibitory)

Axon (output to other neurons)
Perceptron

\[ a = b + \sum_{d=1}^{D} w_d x_d \]

Error driven learning

\[ a = b + \sum_{d=1}^{D} w_d x_d \]

- At each step, return SIGN(a)
- if SIGN(a) ≠ y update parameter
- otherwise don’t change

Example: \( y = 1 \) and prediction is -1

- update \( w' = w + yx = w + x \)
- \( b' = b + y = b+1 \)
Does this move \( a \) in the right direction?

- update \( \mathbf{w}' = \mathbf{w} + y\mathbf{x} = \mathbf{w} + \mathbf{x} \)
- \( b' = b + y = b+1 \)

\[
a' = \sum_{d=1}^{D} w'_d x_d + b' \\
= \sum_{d=1}^{D} (w_d + x_d) x_d + (b + 1) \\
= \sum_{d=1}^{D} w_d x_d + b + \sum_{d=1}^{D} x_d x_d + 1 \\
= a + \sum_{d=1}^{D} x_d^2 + 1 > a
\]

Does this move \( a \) in the right direction?

- update \( \mathbf{w}' = \mathbf{w} + y\mathbf{x} = \mathbf{w} + \mathbf{x} \)
- \( b' = b + y = b+1 \)

\[
a' = \sum_{d=1}^{D} w'_d x_d + b' \\
= \sum_{d=1}^{D} (w_d + x_d) x_d + (b + 1) \\
= \sum_{d=1}^{D} w_d x_d + b + \sum_{d=1}^{D} x_d x_d + 1 \\
= a + \sum_{d=1}^{D} x_d^2 + 1 > a
\]

Does this move \( a \) in the right direction?

- update \( \mathbf{w}' = \mathbf{w} + y\mathbf{x} = \mathbf{w} + \mathbf{x} \)
- \( b' = b + y = b+1 \)

\[
a' = \sum_{d=1}^{D} w'_d x_d + b' \\
= \sum_{d=1}^{D} (w_d + x_d) x_d + (b + 1) \\
= \sum_{d=1}^{D} w_d x_d + b + \sum_{d=1}^{D} x_d x_d + 1 \\
= a + \sum_{d=1}^{D} x_d^2 + 1 > a
\]

When do we stop?

- Hyperparameter MaxIter
- training too long could lead to overfitting
- training for too few steps could lead to underfitting
Randomizing samples helps

- permute the samples before starting
- even better: permute the samples for each iteration

What is the decision boundary?

- $w_a = b + D \sum_{d=1}^{D} w_d x_d = 0$

How good is this algorithm?

- Convergence: an entire pass without changing the weights.

- If the data is linearly separable, the algorithm will converge. But not necessarily to the “best” boundary
The notion of margin is defined as:

\[
\text{margin}(D, w, b) = \begin{cases} 
\min_{(x,y) \in D} y(x \cdot w + b) & \text{if } w \text{ separates } D \\
-\infty & \text{otherwise}
\end{cases}
\]

Therefore, the margin of the data set is:

\[
\text{margin}(D) = \sup_{w,b} \text{margin}(D, w, b)
\]

If the data is linearly separable with margin \( \gamma \) and \( ||x|| \leq 1 \), then the algorithm will converge in \( \frac{1}{\gamma^2} \) updates.

The relationship to stochastic gradient descent:

- We can write the loss function of the perceptron as:
  \[
  L(y, \hat{y}) = \max(0, -y(b + \sum_d w_d x_d))
  \]
- This is not differentiable, we need to learn more about sub-gradient methods.
- At each step, we update using only one datapoint.
Neural Networks

Every node is analogous to a neuron

sigmoid unit

\[
\begin{align*}
\sum_{i} w_{ij} x_i + b &= a_j \\
\sigma(a_j) &= y_j
\end{align*}
\]
How to train

- Calculate each output
- Calculate output error $E$
- Back-propagate $E$ (weighting it by the gradient of previous layer and activation function)
- Calculate the gradients $dE/dw$ and $dE/db$
- Update the parameters

Every node is analogous to a neuron

Backprop with one node per layer

sigmoid unit