02-620 Week 4

Machine Learning for Scientists

Aidan Jan

February 11, 2026

Logistic Regression

Logistic regression aims to fit a logistic curve to data.

$$P(Y_i = 1|x_i; \theta) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x_i)}}$$

The linear part in the exponent is the equation for the decision boundary.

Logistic Regression: Data

- N data points with D features: $x_i \in \mathbb{R}^D$
- Labels: $y_i \in \{0, 1\}$

Logistic Regression: Model

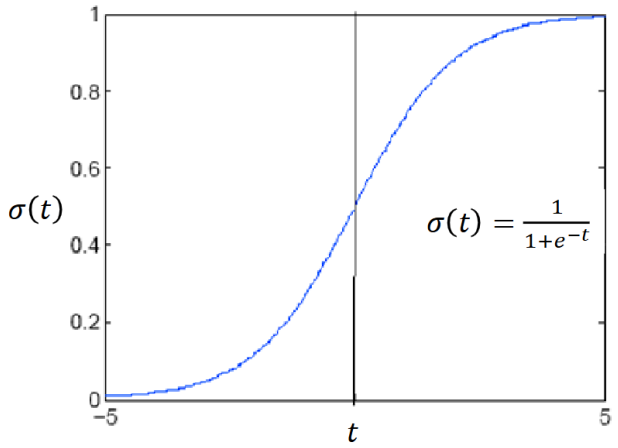
- Parameters: $\theta = (\theta_0, \theta_1, \cdots, \theta_D)$
- Linear predictor: $\mu(x) = \theta_0 + \sum_{j=1}^{D} \theta_j x_j$
- Sigmoid: $\sigma(t) = \frac{1}{1+e^{-t}}$
- Parametric function:

$$P(Y = 1|x; \theta) = \frac{1}{1 + \exp\left(-\left(\theta_0 + \sum_{j=1}^{D} \theta_j x_j\right)\right)} = \sigma(\mu(x)) = \frac{1}{1 + e^{-\mu(x)}}$$

We can find $P(Y = 0|x; \theta)$ by subtracting the $Y = 1$ case from 1 since the probabilities must sum to 1.

$$P(Y = 0|x; \theta) = 1 - \sigma(\mu(x)) = \frac{e^{-\mu(x)}}{1 + e^{-\mu(x)}}$$

Logistic Function (Sigmoid)

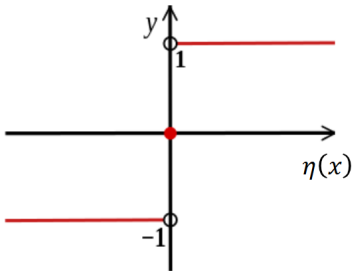


- When $t = 0$, $\sigma(t) = 0.5$
- When $t \to \infty$, $\sigma(t) = 1$
- When $t \to -\infty$, $\sigma(t) = 0$

Simple Classifier vs. Logistic Classifier

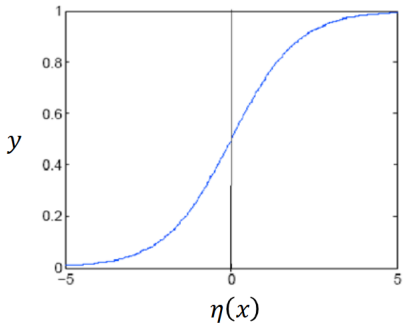
$y = sign(\eta(x))$

- $\eta(x) > 0$ $\quad y = 1$
- $\eta(x) < 0$ $\quad y = -1$ (or $y = 0$)



$y = \sigma(\eta(x)) = \frac{1}{1+e^{-\eta(x)}}$

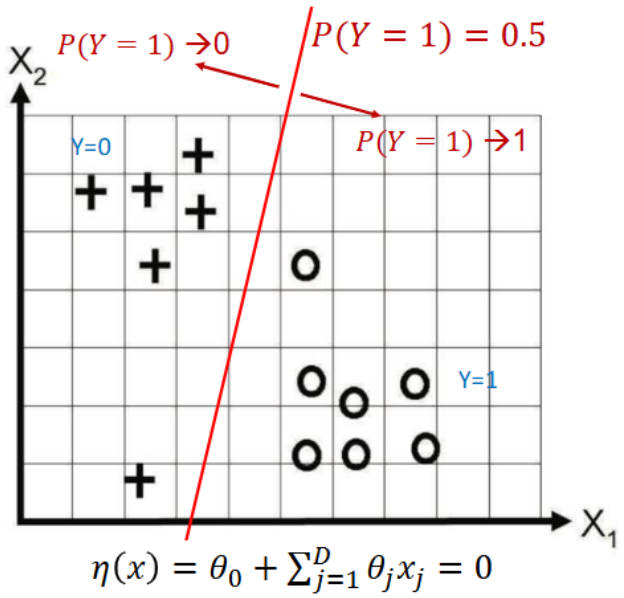
- $\eta(x) \to \infty$ $\quad y = 1$
- $\eta(x) = 0$ $\quad y = 0.5$
- $\eta(x) \to -\infty$ $\quad y = 0$



$\eta(x) = \theta_0 + \sum_{j=1}^{D} \theta_j x_j$

Typically, when the logistic function is used for classification, we round the value to the nearest integer. However, the logistics curve is useful because the curve is differentiable, which makes gradient descent possible.

Decision Boundary in Logistic Regression



We know that the probability of the class label $Y = 1$ and $Y = 0$ are:

$$P(Y = 1|x; \theta) = \frac{1}{1 + e^{-\mu(x)}} P(Y = 0|x; \theta) \qquad = 1 - \sigma(\mu(x)) = \frac{e^{-\mu(x)}}{1 + e^{-\mu(x)}}$$

Therefore,

$$\frac{P(Y = 1|x; \theta)}{(Y = 0|x; \theta)} = e^{\mu(x)}$$

We can log both sides to get

$$\mu(x) = \log \left(\frac{P(Y = 1|x; \theta)}{(Y = 0|x; \theta)} \right)$$

Notice that $\mu(x)$ is a linear model! $\mu(x) = \theta_0 + \sum_{j=1}^{D} \theta_j x_j = 0$

Logistic Regression: Beyond Binary Classification

Data

- N data points with D features: $x_i \in \mathbb{R}^D$
- Labels: $y_i \in \{1, \cdots, K\}$

Model

- Parameters: $\theta = (\theta_{k0}, \theta_{k1}, \cdots, \theta_{kD})$, for $k = 1, \ldots, K-1$
- Linear predictor: $\mu_k(x) = \theta_{k0} + \sum_{j=1}^{D} \theta_{kj} x_j$, for $k = 1, \ldots, K-1$
- Sigmoid: $\sigma(t) = \frac{1}{1+e^{-t}}$
- Parametric function:
  - For $k = 1, \ldots, K-1$, $P(Y = k|x; \theta) = \sigma(\mu_k(x)) = \frac{e^{-\mu_k(x)}}{1+\sum_{k=1}^{K-1} e^{-\mu_k(x)}}$
  - $P(Y = K|x; \theta) = \frac{1}{1+\sum_{k=1}^{K-1} e^{-\mu_k(x)}}$
  - Note that $\sum_{k=1}^{K} P(Y = k|x; \theta) = 1$.

Logistic Regression: Inference

For a new data point $x_{i'}$, predict $\hat{y}_{i'} = f(x_{i'}; \hat{\theta})$
Compute and classify accordingly:

- $P(Y = 1|x; \theta) = \sigma(\mu(x))$
- $P(Y = 0|x; \theta) = 1 - \sigma(\mu(x))$

Logistic Regression: Learning via MLE

- **Data:** $\mathcal{D} : x_1, \cdots, x_N \in \mathbb{R}^D$, $y_1, \cdots, y_N \in \{0, 1\}$
- **Estimate parameters:** $\theta = (\theta_0, \theta_1, \cdots, \theta_D)$
- **MLE estimate:** $\hat{\theta} = \arg\max_\theta \sum_{i=1}^N \log P(Y_i|x_i; \theta) = \arg\max_\theta l(\theta)$

$$
\begin{aligned}
l(\theta) &= \sum_{i=1}^{N} [Y_i \log P(Y_i = 1|x_i; \theta) + (1 - Y_i) \log P(Y_i = 0|x_i; \theta)] \\
&= \sum_{i=1}^{N} \left[Y_i \log \frac{P(Y_i = 1|x_i; \theta)}{P(Y_i = 0|x_i; \theta)} + \log P(Y_i = 0|x_i; \theta) \right] \\
&= \sum_{i=1}^{N} \left[Y_i \left(\theta_0 + \sum_{j=1}^{D} \theta_j x_{ij} \right) - \log \left(1 + \exp \left(\theta_0 + \sum_{j=1}^{D} \theta_j x_{ij} \right) \right) \right]
\end{aligned}
$$

Take derivative of $l(\theta)$ with respect to θ

$$
\begin{aligned}
\frac{\partial l(\theta)}{\partial \theta_0} &= \sum_{i=1}^{N} \left[Y_i - \frac{e^{\mu_i}}{1 + e^{\mu_i}} \right] \\
\frac{\partial l(\theta)}{\partial \theta_j} &= \sum_{i=1}^{N} x_{ij} \left[Y_i - \frac{e^{\mu_i}}{1 + e^{\mu_i}} \right] \quad \text{for } j = 1, \ldots, D
\end{aligned}
$$

Note that

- $l(\theta)$ is concave
- Gradient ascent/descent gives the optimal solution.

To gradient ascent,

- For $t = 1, 2, \ldots$:
  - For all j, update $\theta_j \leftarrow \theta_j + \Delta \frac{\partial l(\theta)}{\partial \theta_j}$ (Δ is the step size or learning rate.)
- Stop when $\left| l(\theta^{(t)}) - l(\theta^{(t-1)}) \right| < \epsilon$
- ϵ is a small value (error).

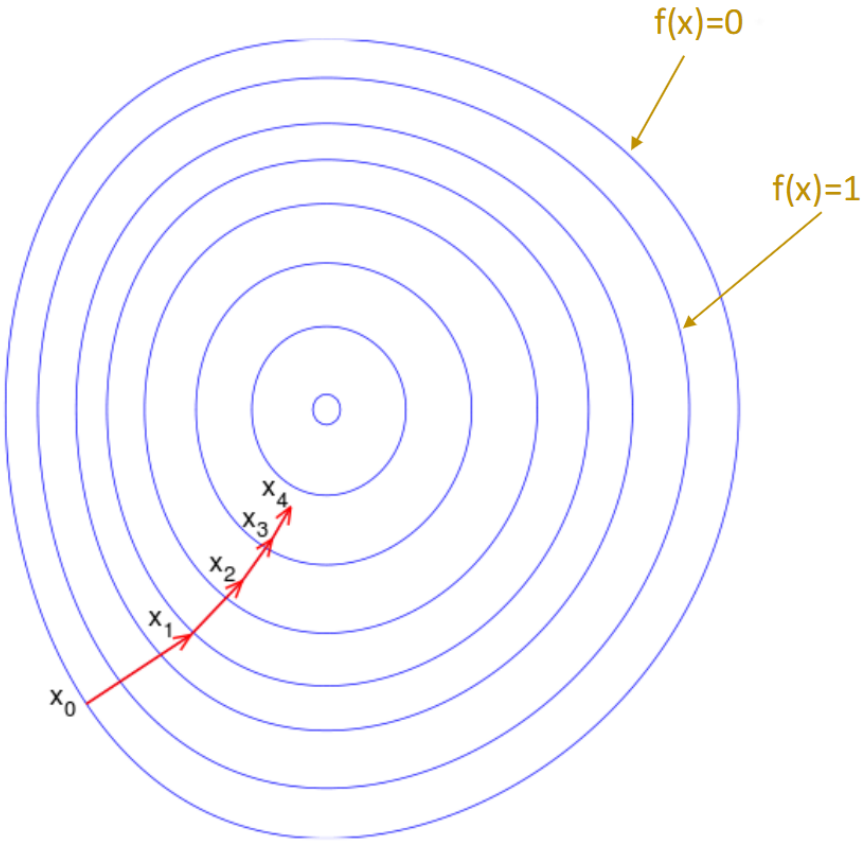
Aside: Gradient Ascent

To find x that maximizes a function $f(x)$:

- Start with random initialization x_0
- For $t = 1, \ldots$

- $x_{t+1} = x_t + \lambda f'(x_t)$ with step size λ

- λ
  - $\lambda \uparrow$: overshoot
  - $\lambda \downarrow$: too many iterations.



Log-likelihood of the logistic regression is concave. Gradient descent/ascent will:

- reach the maximum of the log likelihood
- find the optimal estimate of the parameters

Logistic Regression: Adding Regularization

- **MLE estimate**

$$\hat{\theta} = \arg\max_{\theta} \sum_{i=1}^{N} \log P(Y_i|x_i;\theta)$$

- **Ridge regression**

$$\hat{\theta} = \arg\max_{\theta} \sum_{i=1}^{N} \log P(Y_i|x_i;\theta) + \lambda\|\theta\|_2^2$$

- **LASSO regression**

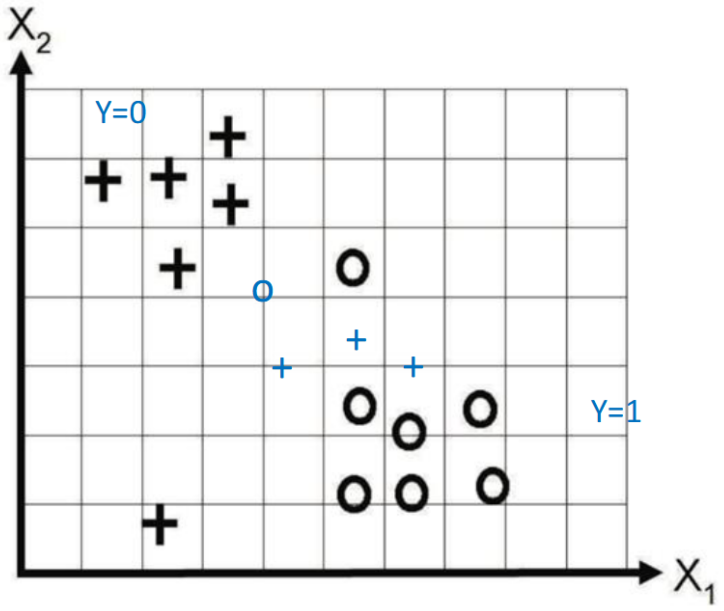
$$\hat{\theta} = \arg\max_{\theta} \sum_{i=1}^{N} \log P(Y_i|x_i;\theta) + \lambda\|\theta\|_1$$

Inseparable Distributions

Recall that

$$P(Y = 1|x; \theta) = \frac{1}{1 + \exp\left(-\left(\theta_0 + \sum_{j=1}^{D} \theta_j x_j\right)\right)}$$

What if positive and negative samples are not completely separable?



- How would logistic regression handle this?
- Limitations of logistic regression?

Generative vs. Discriminative Classifiers

Training classifiers involves estimating $f : X \to Y$ or $P(Y|X)$.

- **Generative Classifiers** (e.g., Naive Bayes)
  - Model and learn $P(Y, X) = P(X|Y)P(Y)$
  - Derive $P(Y|X)$ from $P(Y, X)$ using Bayes rule
  - Find $\theta = \arg\max_\theta \prod_i P(y_i, x_i|\theta)$
  - Different assumptions about *generative process* for the data: $P(X, Y)$, priors on $\theta, \ldots$.
- **Discriminative classifiers** (e.g., Logistic regression)
  - Model $P(Y|X)$ directly
  - Find $\theta = \arg\max_\theta \prod_i P(y_i|x_i, \theta)$
  - Different assumptions about conditional probability: $P(Y|X)$, priors on $\theta, \ldots$

Measuring Accuracy of Classifier

- Precision = (number classified as positive AND positive in data) / (number classified as positive)
  - e.g., how many of the patients classified as "disease" are in fact truly "disease"?
- Recall = (number classified as positive AND positive in data) / (number ground truth positive in data)

  - e.g., how many of the "disease" patients were classified as "disease"?
- True positive (classified as positive, positive in data)
- False positive (classified as positive, negative in data)
- True negative (classified as negative, negative in data)
- False negative (classified as negative, positive in data)

Summary

- Naive Bayes Classifier
  - Model description via $P(X|Y)$ and $P(Y)$
  - MLE: closed-form solution, no iterative learning
- Logistic regression
  - Model description for $P(Y|X)$
  - MLE: no closed-form solution, iterative learning via gradient descent
- Generative vs. Discriminative classifier

Decision Trees

Data

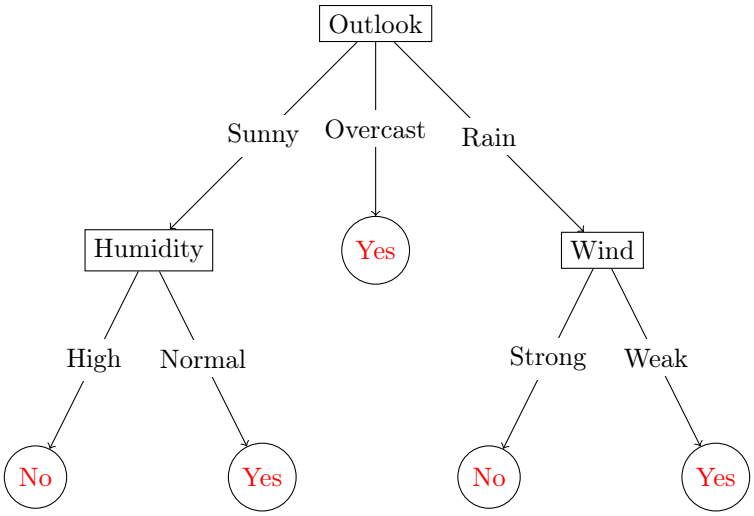
- N data points with D attributes: x_i (discrete / continuous / mixed)
- Labels: $y_i \in \{0, 1\}$

For example:

Day	Outlook	Temperature	Humidity	Wind	PlayTennis (Label)
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	No
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Model

- Each **internal node** selects an attribute x_j and applies a decision rule (e.g., $x_j = v$ for discrete or $x_j < \tau$ for continuous features)
- Each **branch** corresponds to an outcome of the rule
- Each **leaf node** outputs a prediction: a class label or $P(Y|x \in leaf)$



Inference

- For a new data point, traverse the tree from the root to a leaf to obtain the prediction
- Day 1 → No (Outlook: Sunny → Humidity: High → No).

Decision Tree Motivations

- Can produce small, accurate, and interpretable classifiers
- Often provide insight into the problem and help debug features
- Naturally captures feature interactions (e.g., "sunny AND humid") - interactions that linear models miss.
- Fast and cheap at test time - not all features need to be computed
- Can account for feature acquisition costs - some features are much cheaper than others (e.g., "blood pressure" vs. "MRI")

Decision Tree Learning

- Estimate internal nodes and decision rules for the decision tree
- **Initialize:** assign all data points to the root node.
- **Recursively grow the tree.** For data points at the current node, select the best **attribute** x_j and split:
  - Categorical: $x_j = c_1, c_2, \ldots$
  - ContinuousL $x_j < \tau$ or $x_j \geq \tau$
- Ask: Does x_j help predict y? (e.g., lower the **entropy** of y in the resulting partitions (details later).)
- Assign data points to corresponding child nodes, repeat until a node is sufficiently homogeneous, and make the node a leaf and assign majority label.

Common stopping criteria include:

- Entropy (or Gini) below a threshold
- Maximum tree depth reached

- Minimum node size reached

Procedure: For each attribute $x_1, \ldots, x_D$, evaluate the **entropy** of the resulting splits. Select the one with the most entropy reduction.

Entropy

Entropy measures impurity in a set of points. The lower the entropy, the more pure the data is.

- For example, consider the decision for *PlayTennis* above. If we originally tested for Humidity instead of Outlook, then we have

3+, 4- (E = 0.985) if humidity is high

6+, 1- (E = 0.592) if humidity is normal

- + represents the count of "Yes" for *PlayTennis*, and − represents the count for "No" for *PlayTennis*
- The more the counts lean to one direction, the lower the entropy. Therefore, if every label is "Yes" or if every label is "No", then the entropy is 0.
- On the contrary, equal counts in "Yes" and "No" lead to an entropy of 1, the maximum value of entropy possible.

Entropy is represented by a distribution.

- **Binary distribution:** If $Y \sim Bern(p)$, then $H(Y) = -p \log_2 p - (1-p) \log_2 (1-p)$.
- **Discrete distribution:** If Y takes values $\{c_1, \ldots, c_K\}$ with probabilities $p_1, \ldots, p_k$, then $H(y) = -\sum_{k_1}^{K} p_k \log_2 p_k$
- **Sample entropy (binary labels):** For a set of data points S, let p_0 and p_1 be the proportion of 0's and 1's. Then $H(S) = -p_0 \log_2 p_0 - p_1 \log_2 p_1$

Conditional Entropy and Information Gain

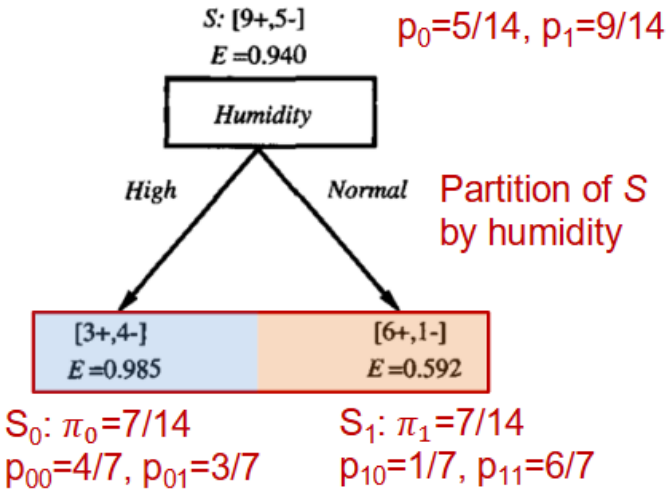
For discrete random variables X and Y, the entropy $H(Y) = \sum_y P(Y = y) \log_2 P(Y = y)$ quantifies the uncertainty of a random variable Y.

- **Conditional entropy:** $H(Y|X) = \sum_x P(X = x) H(Y|X = x)$ quantifies the *remaining uncertainty* of Y after knowing X.
- **Information gain:** $IG(Y; X) = H(Y) - H(Y|X)$ quantifies the reduction in uncertainty due to knowing X.

Example

For a dataset S, let p_0 and p_1 be the proportion of 0's and 1's. Partition of S into S_0 and S_1 with proportions π_0 and π_1.

- Let p_{00} and p_{01} be the proportion of 1's and 0's in S_0
- Let p_{10} and p_{11} be the proportion of 1's and 0's in S_1



Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Entropy:

$$H(S) = -p_0 \log_2 p_0 - p_1 \log_2 p_1$$

$$H(S_0) = -\frac{4}{7}\log_2\frac{4}{7} - \frac{3}{7}\log_2\frac{3}{7} = 0.985$$

$$H(S_1) = -\frac{1}{7}\log_2\frac{1}{7} - \frac{6}{7}\log_2\frac{6}{7} = 0.592$$

Conditional Entropy:

$$H(S|partition) = \pi_0 H(S_0) + \pi_1 H(S_1) = -\pi_0(p_0 \log_2 p_{00} - p_1 \log_2 p_{01}) - \pi_1(p_0 \log_2 p_{10} - p_1 \log_2 p_{11})$$
$$= -\frac{7}{14}\cdot 0.985 - \frac{7}{14}0.592 = 0.788$$

quantifies the remaining uncertainty of S after knowing the partition.

Information gain:

$$IG(S|partition) = H(S) - H(S|partition)$$
$$= 0.94 - 0.788 = 0.152$$

quantifies the reduction in uncertainty due to knowing the partition.

We want the attribute with the largest information gain.

Decision Tree Can Completely Overfit the Data

- Decision trees can be grown arbitrarily deep to achieve zero training error.
- However, overly complex trees may not generalize well.

Recall that overfitting occurs if:

- Error(training data) is low, and
- Error(All data) is high.

How to avoid overfitting:

- Stop growing the tree when the data split is not statistically significant

- Growing tree only to a given depth
- Grow full tree then prune
- **Split the data into training and test sets.** Evaluate the tree on the test set, but train on the training set.

How to select the best tree

- Should we use the training data or test data to measure the classification accuracy of the decision tree?

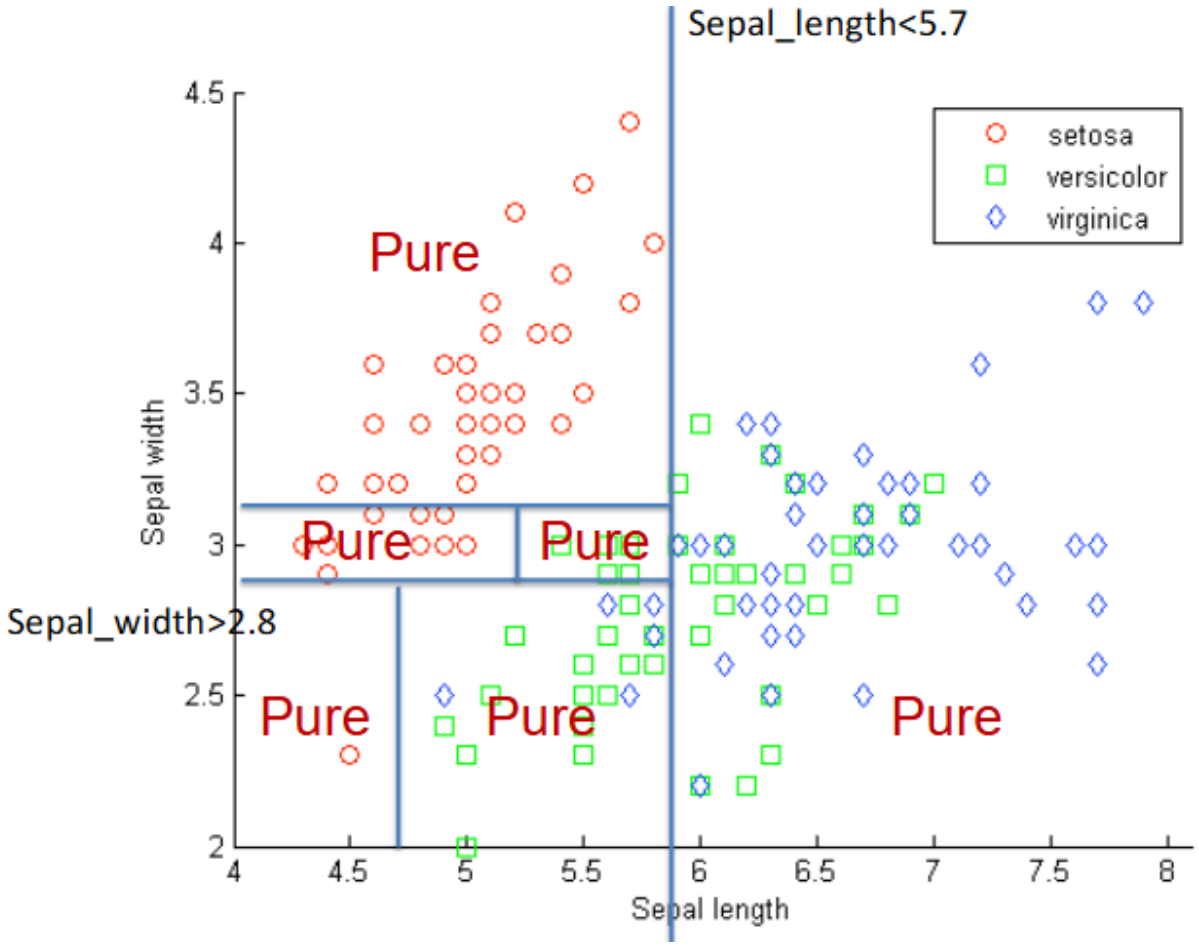
Decision Trees with Continuous Attributes

Suppose for our *PlayTennis* example, we now want to consider temperature. To do this, we **learn a decision threshold τ.**

- Sort the sample values
- Evavluate all candidate thresholds that produce distinct splits
- Select the threshold with the highest information gain
- Complexity $O(N_{node})$

The decision boundary may be decided using other models, for example logistic regression. Alternatively, we can just section out the points like follows:

What's the decision boundary for decision trees (red dots vs. rest)?



In this case, even just for two labels, the decision boundary is very complicated.

Decision Tree Pros and Cons

- Pros:
  - Tree structure: easy to interpret
  - Handle mixed discrete and continuous inputs
  - Automatic variable selection
  - Scale well to large data
- Cons:
  - Prediction accuracy is often lower than other classifiers
  - Small changes in data have a large change on the tree structure (called **high variance**)
  - An error at the top of the tree affect the rest of the tree growing process

Decision Tree Classifier Summary

- Model:
  - Tree-structured, non-linear model
- Learning:
  - Recursively grow the tree
  - Select attributes and splits based on information gain
  - Prune the tree to avoid overfitting
- Inference:
  - Traverse the tree to make predictions
  - Computationally efficient at test time