Learning

Learning Definitions

- *Learning* is improvement of performance (time, accuracy).
- In *supervised learning*, from training examples of input-output pairs, predict the output of a new input.
- In *unsupervised learning*, examples do not have outputs. The most common task is *clustering*.
- In *semi-supervised learning*, some examples have outputs. For example, in *reinforcement learning*, an input is a series of actions, and the output is intermittent feedback.

Supervised Learning

- Assume the learner is given the following:
  - a set of input features, $X_1, \ldots, X_n$;
  - a target feature, Y;
  - a set of *training examples*, each with values for the X's and Y
  - a set of *test examples*, each with values only for the X's.
- The learner finds a *hypothesis* h to predict the target from the inputs.
- Usually, h is restricted to be an element from a *hypothesis space*.
- *Regression* is when the target is continuous.
- *Classification* is when the target is discrete.

Example of Examples

No.	Input Features				Target
	Outlook	Temp	Humidity	Windy	
1	sunny	hot	high	false	neg
2	sunny	hot	high	true	neg
3	overcast	hot	high	false	pos
4	rain	mild	high	false	pos
5	rain	cool	normal	false	pos
6	rain	cool	normal	true	neg
7	overcast	cool	normal	true	pos
8	sunny	mild	high	false	neg
9	sunny	cool	normal	false	pos
10	rain	mild	normal	false	pos
11	sunny	mild	normal	true	pos
12	overcast	mild	high	true	pos
13	overcast	hot	normal	false	pos
14	rain	mild	high	true	neg

Evaluating Predictions

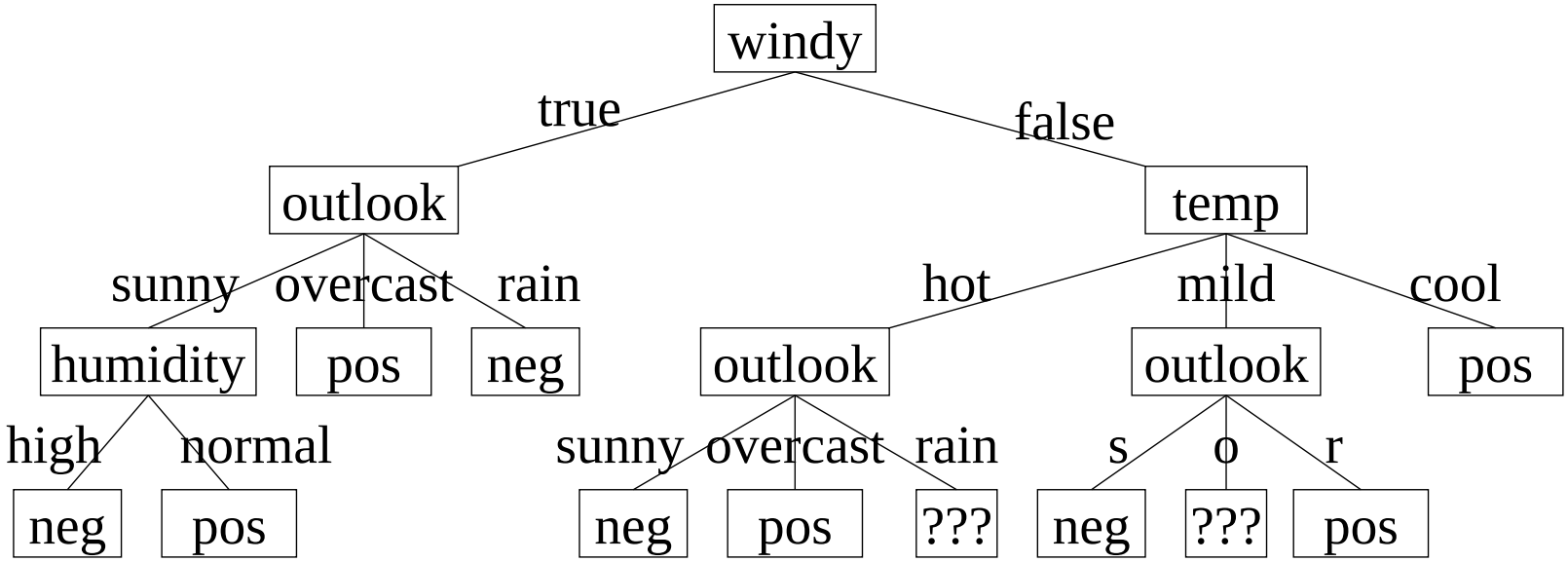
- Let y_e be the target value for example e.
- Let $\widehat{y}_e$ be the predicted value.
- *Error* (or *loss*) measures how close $\widehat{y}_e$ is to y_e.
- Zero-One Error: if $y_e \neq \widehat{y}_e$, then 1, else 0
- Absolute Error: $|y_e - \widehat{y}_e|$
- Squared Error: $(y_e - \widehat{y}_e)^2$
- Entropy: $-(y_e \log \widehat{y}_e + (1 - y_e) \log(1 - \widehat{y}_e))$
  (assumes y_e and $\widehat{y}_e$ are probabilities.)
- and many variations.

- For classification, use $y_e \in \{0, 1\}$ or $\{-1, 1\}$.
- Secret of machine learning:
  update hypothesis to reduce error.

Definition

- Decision trees are a representation for classification.
  - Each nonleaf is labeled by a feature.
  - Edges from nonleaf to children are labeled by feature values.
  - Each leaf is labeled by a prediction.
- Typical Algorithm: Construct the tree top-down.
  - Find the "best" feature.
  - Split examples based on feature's values.

Example of a Decision Tree



Algorithm for Learning Decision Trees

Procedure *DTLearner*(X, Y, E)

Inputs X: set of input features, $X = \{X_1, \ldots, X_n\}$

Y: target feature

E: set of training examples

if stopping criterion is true then

return a leaf labeled with prediction of Y

Select feature $X_i \in X$, with domain V

let $T =$ nonleaf node labeled X_i

for each $v \in V$

let $E' = \{e \in E : X_i = v\}$

let $T' =$ *DTLearner*(X, Y, E')

add edge from T to T' labeled v

return T

Selecting a Feature: Information Gain

- p positive examples and n negative examples
- The information contained is:

$$I(p,n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

- Feature X_i has v values, p_j positive examples and n_j negative examples when $X_i = v_j$
- The $Remainder$ of X_i is:

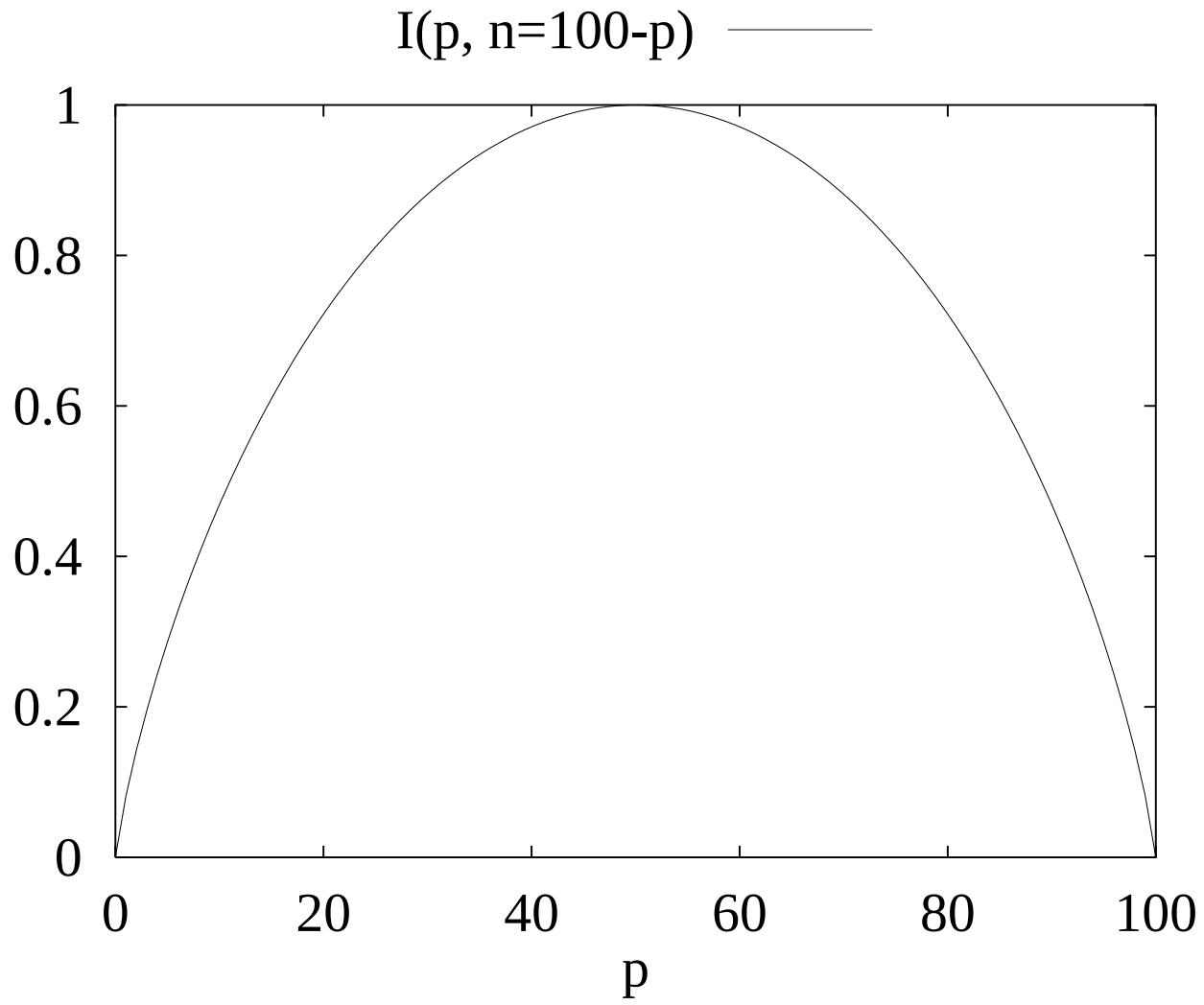
$$Remainder(X_i) = \sum_{j=1}^{v}\frac{p_j+n_j}{p+n}\ I(p_j,n_j)$$

- The information gain of X_i is:

$$Gain(X_i) = I(p,n) - Remainder(X_i)$$

Plot of Information Function

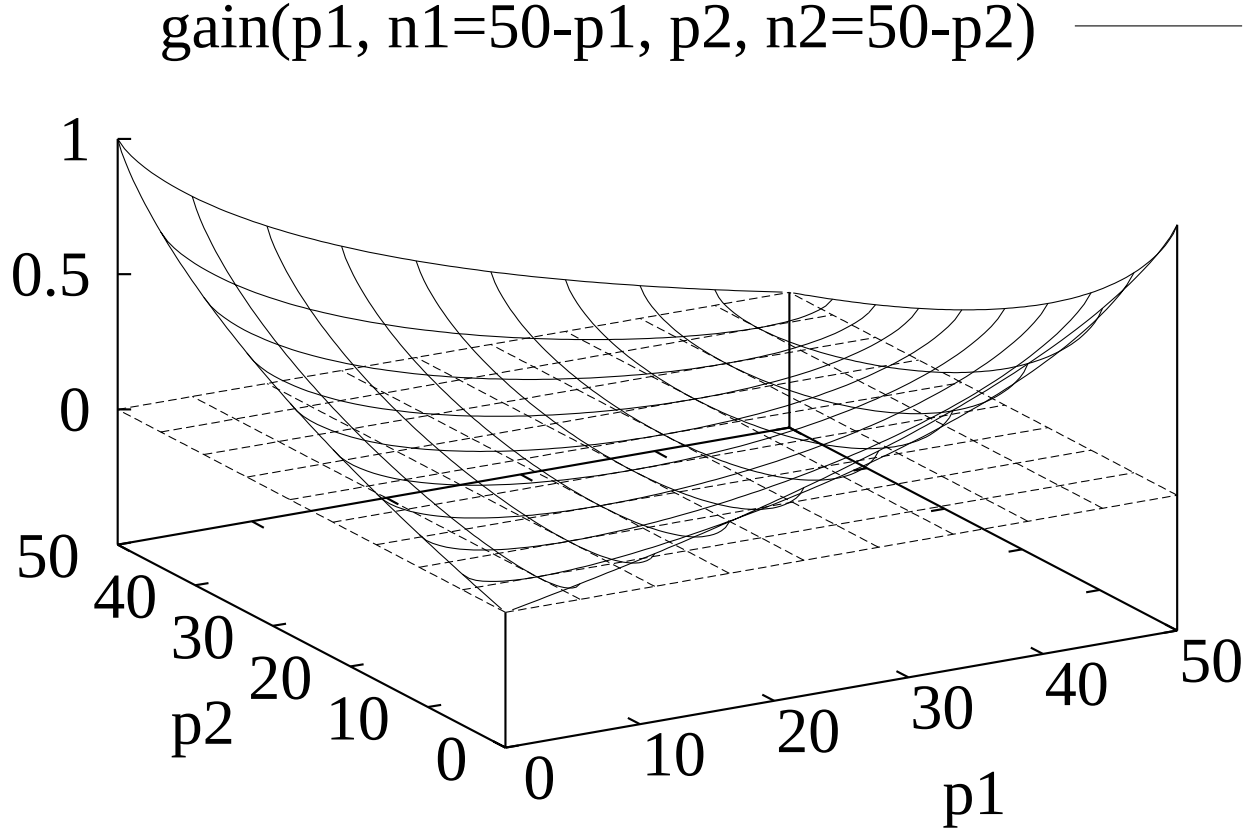
p positive examples and n negative examples



Plot of Information Gain

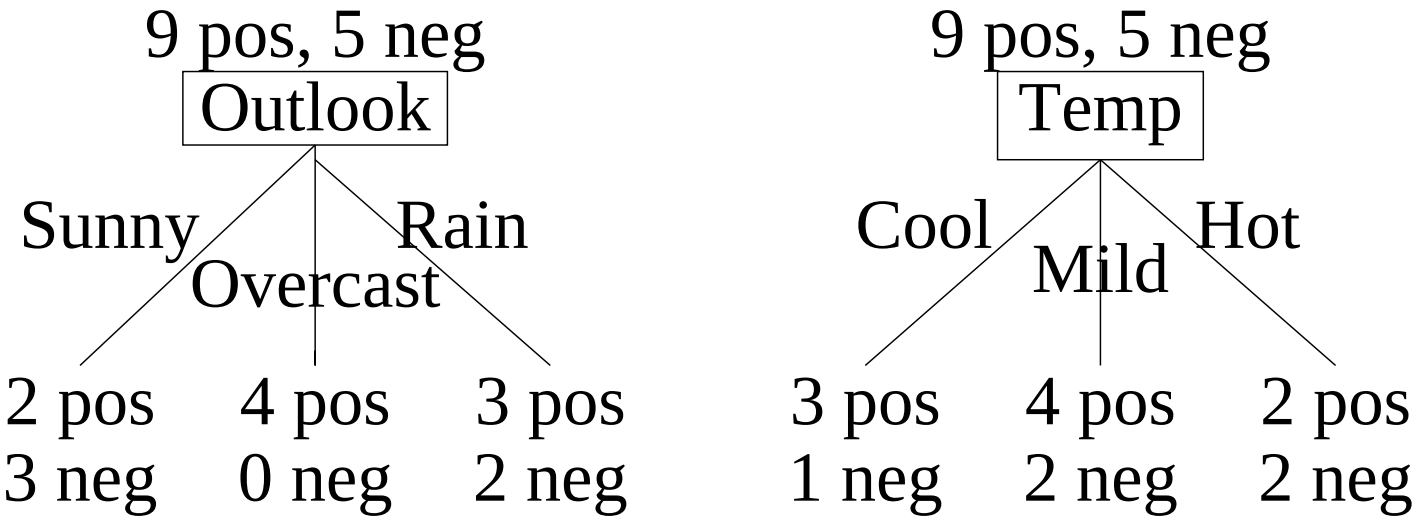
p_1 positive and n_1 negative exs. when $X_i = v_1$

p_2 positive and n_2 negative exs. when $X_i = v_2$



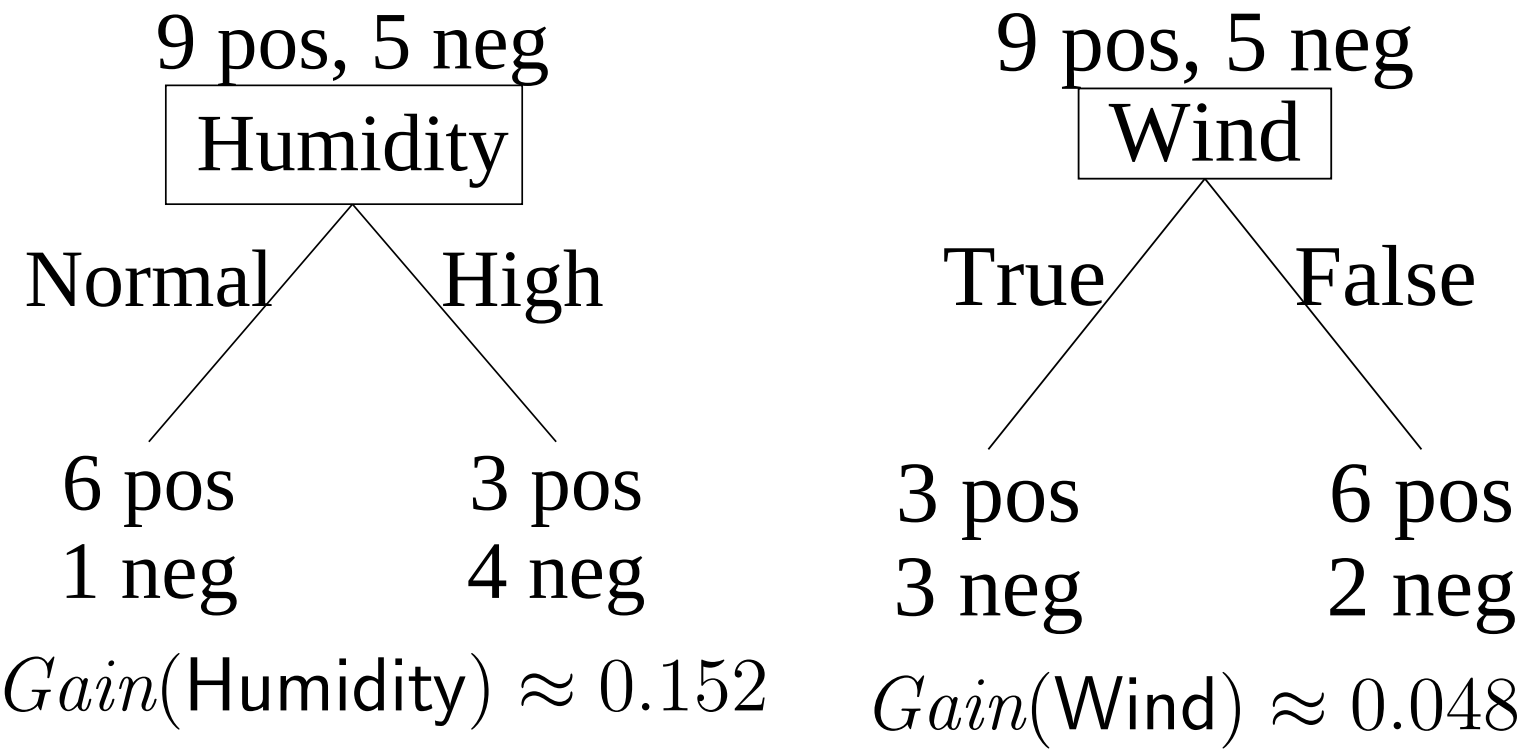
Example of Feature Selection

Refer to Example of Examples earlier.



$Gain(\text{Outlook}) \approx 0.246$ $Gain(\text{Temp}) \approx 0.029$

Feature Selection, Continued



$Gain(\text{Humidity}) \approx 0.152$

$Gain(\text{Wind}) \approx 0.048$

Outlook has the highest gain.

Overcast branch is pure.

Need to construct DTs for two branches.

Other Choices

- When to stop:
  - all examples are classified the same
  - all examples have the same feature values
  - too few examples
- *Overfitting* occurs when the algorithm tries to fit noise in the training data (outliers, random fluctuations, approx. decision boundary).
- Handling overfitting: use part of training set as a *validation set*.
  - create decision tree with *training set*
  - prune decision tree with *validation set*

Special Cases in Decision Trees

- Feature X_i is numeric.
  - Find best $X_i \leq v$ test. Requires sorting.
  - Or: Discretization. Partition X_i into ranges.
- Feature X_i has missing values.
  - Pretend missing is just another value.
  - Or: Ignore missing values. Split examples with missing values across branches.
- Feature X_i has many discrete values.
  - Find best $X_i = v$ test. Forms binary tree.
  - Or: Partition values into subsets.

Iris Dataset

No.	Input Features				Target
	Sepal length	Sepal width	Petal length	Petal width	
1	5.8	4.0	1.2	0.2	I. setosa
2	5.7	4.4	1.5	0.4	I. setosa
3	4.8	3.4	1.9	0.2	I. setosa
4	5.0	3.5	1.6	0.6	I. setosa
5	6.7	3.0	5.0	1.7	I. versicolor
6	6.0	2.7	5.1	1.6	I. versicolor
7	5.9	3.2	4.8	1.8	I. versicolor
8	6.0	3.4	4.5	1.6	I. versicolor
9	7.6	3.0	6.6	2.1	I. virginica
10	4.9	2.5	4.5	1.7	I. virginica
11	7.3	2.9	6.3	1.8	I. virginica
12	6.7	2.5	5.8	1.8	I. virginica

Numerical Learning

- Numerical learning methods learn the parameters or weights of a model, often by optimizing an error function. Examples include:
- Calculate the parameters of a probability distribution.
- Separate positive from negative examples by a decision boundary.
- Find points close to positive but far from negative examples.
- Update parameters to decrease error.

Naive Bayes

- For target class Y and features X_i, assume:

$$P(Y, X_1, \ldots, X_n) = P(Y)P(X_1|Y)...P(X_n|Y)$$

- This corresponds to a Bayesian network where Y is the sole parent of each X_i.
- To calculate the belief in Y:

$$P(Y \mid X_1, \ldots, X_n) = \frac{P(Y, X_1, \ldots, X_n)}{P(X_1, \ldots, X_n)}$$

- The denominator is the same for all values of Y, so to compare only the numerator needs to be calculated.

Naive Bayes Learning

- Estimate prior and conditional probabilities by counting, e.g.,
  - $Y =$ pos in 9 of the 14 examples
  - $X_1 =$ sunny in 2 examples where $Y =$ pos.
- If an outcome occurs m times out of n examples, Laplace's law of succession recommends the estimate $(m+1)/(n+k)$ where k is the number of outcomes.
  - Estimate $P(Y = \text{pos}) = (9+1)/(14+2) = 10/16$
  - Estimate $P(X_1 = \text{sunny} \mid Y = \text{pos}) = (2+1)/(9+3) = 3/12$

Naive Bayes Example

Using Laplace's law of succession on the 14 examples:

$$P(Y = \text{pos}) = (9+1)/(14+2) = 10/16$$

$$P(Y = \text{neg}) = (5+1)/(14+2) = 6/16$$

$$P(X_1 = \text{sunny} \mid Y = \text{pos}) = (2+1)/(9+3) = 3/12$$

$$P(X_1 = \text{overcast} \mid Y = \text{pos}) = (4+1)/(9+3) = 5/12$$

$$P(X_1 = \text{rain} \mid Y = \text{pos}) = (3+1)/(9+3) = 4/12$$

Naive Bayes Example Continued

For the first example:

$$\begin{aligned} & P(Y = \text{pos} \mid \text{sunny}, \text{hot}, \text{high}, \text{false}) \\ = \;& \alpha\,(10/16)\,(3/12)\,(3/12)\,(4/11)\,(7/11) \\ \approx \;& \alpha\,0.00904 \end{aligned}$$

$$\begin{aligned} & P(Y = \text{neg} \mid \text{sunny}, \text{hot}, \text{high}, \text{false}) \\ = \;& \alpha\,(6/16)\,(4/8)\,(3/8)\,(5/7)\,(3/7) \\ \approx \;& \alpha\,0.02152 \\ \approx \;& \frac{0.02152}{0.00904 + 0.02152} \approx 0.704 \end{aligned}$$

Linear Functions

- A *linear function* of the input features is a dot product of the *weights* and the inputs.

  Inputs: $\mathbf{x} = (1.0, x_1, \ldots, x_n)$
  Weights: $\mathbf{w} = (w_0, w_1, \ldots, w_n)$
  Dot product: $\mathbf{w} \cdot \mathbf{x} = w_o + w_1 x_1 + \ldots + w_n x_n$

- If y is the target and $\widehat{y} = \mathbf{w} \cdot \mathbf{x}$:

  Regression:
  - Squared error: $(y - \widehat{y})^2$
  - Absolute error: $|y - \widehat{y}|$

  Classification (assume $y \in \{-1, 1\}$):
  - Hinge loss: $\max(0, 1 - y\widehat{y})$
  - Logistic loss: $\log(1 + e^{-y\widehat{y}})$

Linear Classification

Why $y * \widehat{y}$ all over the place?

- The goal of linear regression is $y = \widehat{y}$.
- The goal of linear classification is not $y = \widehat{y}$, but sign$(y) =$ sign$(\widehat{y})$.
- If y and $\widehat{y}$ have the same sign, then $y\widehat{y} > 0$.
- The hinge loss includes a *margin*. Its goal is $y\widehat{y} \geq 1$.
- The logistic loss is for interpreting $\widehat{y}$ as a probability $P(y = 1) = 1/(1 + e^{-\widehat{y}})$. This loss function is also known as cross-entropy.
- The global minimum can be found for the hinge and logistic loss.

Example of Numeric Examples

No.	Input Features						Target Feature
	Sunny	Rainy	Hot	Cool	Humid	Windy	
1	1	0	1	0	1	0	−1
2	1	0	1	0	1	1	−1
3	0	0	1	0	1	0	1
4	0	1	0	0	1	0	1
5	0	1	0	1	0	0	1
6	0	1	0	1	0	1	−1
7	0	0	0	1	0	1	1
8	1	0	0	0	1	0	−1
9	1	0	0	1	0	0	1
10	0	1	0	0	0	0	1
11	1	0	0	0	0	1	1
12	0	0	0	0	1	1	1
13	0	0	1	0	0	0	1
14	0	1	0	0	1	1	−1

Generic Linear Learning Algorithm

Procedure *LinearLearner*(X, Y, E, η)

Inputs X: set of input features, $X = \{X_1, \ldots, X_n\}$

Y: target feature

E: set of training examples

η: learning rate

initialize all weights $w_0, w_1, \ldots, w_n$ to zero

repeat until termination

for each example $e = (\mathbf{x}, y) \in E$

$\widehat{y} \leftarrow \mathbf{w} \cdot \mathbf{x}$

$\delta \leftarrow$ update based on y and $\widehat{y}$

$\mathbf{w} \leftarrow \mathbf{w} + \eta\delta\mathbf{x}$

return $\mathbf{w}$

Updates

Name	$\delta \leftarrow$
Regression:	
Squared error	$y - \widehat{y}$
Absolute error	sign$(y - \widehat{y})$
Classification:	
Perceptron	if $y\widehat{y} \leq 0$ then y else 0
Hinge loss	if $y\widehat{y} < 1$ then y else 0
Logistic loss	$y/(1 + e^{y\widehat{y}})$

Except for perceptron, the update is based on the derivative of the error wrt the weights.

Note: For squared error, the optimal solution can be directly computed.

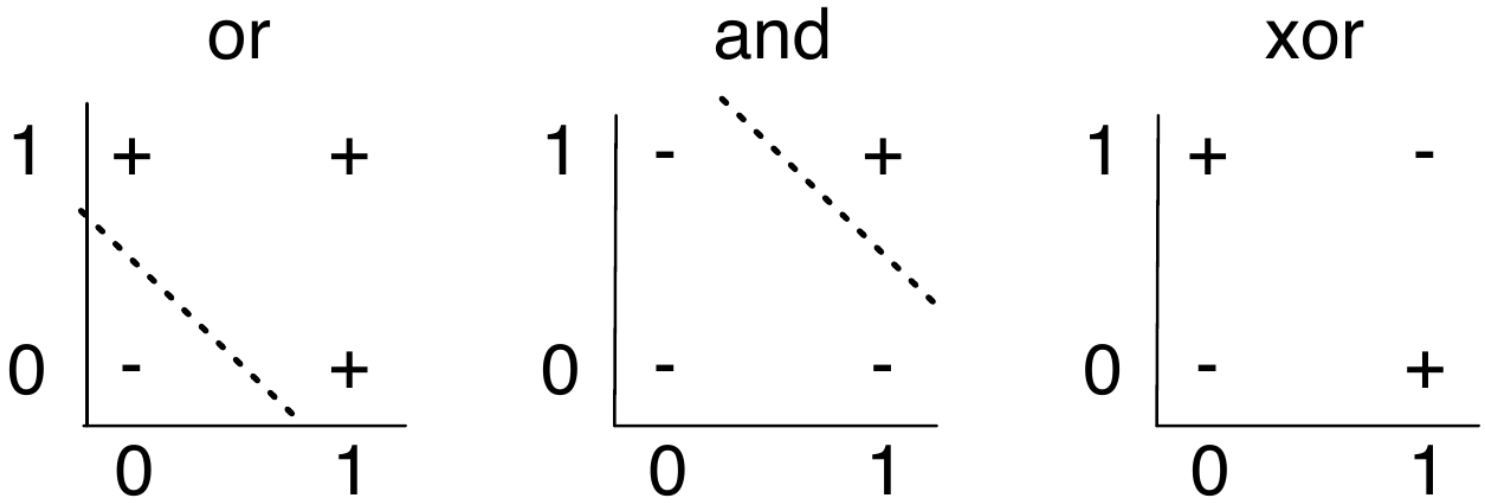
Perceptron Example

Using the learning rate $\eta = 1$:

Features								Weights				
X_0	X_1	X_2	X_3	X_4	Y	$\widehat{y}$	$y\widehat{y}$	w_0	w_1	w_2	w_3	w_4
								0	0	0	0	0
1	0	0	0	1	-1	0	0	-1	0	0	0	-1
1	1	1	1	0	1	-1	-1	0	1	1	1	-1
1	1	1	1	1	1	2	2	0	1	1	1	-1
1	0	0	1	1	-1	0	0	-1	1	1	0	-2
1	0	0	0	0	1	-1	-1	0	1	1	0	-2
1	0	1	0	1	-1	-1	1	0	1	1	0	-2
1	1	0	0	0	1	1	1	0	1	1	0	-2
1	1	0	1	1	1	-1	-1	1	2	1	1	-1
1	0	1	0	0	-1	2	-2	0	2	0	1	-1

Perceptron Properties

- The perceptron can learn *linearly separable* examples with zero error. Linearly separable = exists $\mathbf{w}$ with zero error on all examples.



- Usually, many *epochs* (passes over the training examples) are needed until convergence.
- If zero error is not possible, use hinge/logistic loss and $\eta \approx 0.1/n$, where n is max $\mathbf{x} \cdot \mathbf{x}$.