CS 188: Artificial Intelligence
Learning III: Linear regression & Perceptron

Agent Testing Today!

Instructors: Stuart Russell and Dawn Song
University of California, Berkeley
Recap: Decision Tree

- Iterative process, select the most distinguishing/informative attribute to split on next
- Entropy: measure uncertainty in a probability distribution $\langle p_1, \ldots, p_n \rangle$
  \[
  H(\langle p_1, \ldots, p_n \rangle) = \text{______________}
  \]
  - Quiz: Higher entropy means _____ uncertainty.
    A. more
    B. less
Recap: Decision Tree

- Iterative process, select the most distinguishing/informative attribute to split on next
- Entropy: measure uncertainty in a probability distribution \( \langle p_1, \ldots, p_n \rangle \)
  \[
  H(\langle p_1, \ldots, p_n \rangle) = \sum_i -p_i \log p_i
  \]
- Information gain:
  - reduction on entropy with additional information
- Learning decision tree:
  - Iterative process, selecting the attribute with the highest information gain to split on
Recap: Model Selection & Hyperparameter Tuning

- Data: labeled instances, e.g. emails marked spam/ham
  - Training set
  - Held out set
  - Test set

- Features: attribute-value pairs which characterize each x

- Experimentation cycle
  - Learn parameters (e.g. model probabilities) on training set
  - (Tune hyperparameters on held-out set)
  - Compute accuracy of test set
  - Very important: never “peek” at the test set!

- Evaluation
  - Accuracy: fraction of instances predicted correctly

- Overfitting and generalization
  - Want a classifier which does well on test data
  - Overfitting: fitting the training data very closely, but not generalizing well
  - Underfitting: fits the training set poorly
Supervised Learning

- **Classification** = learning $f$ with discrete output value
- **Regression** = learning $f$ with real-valued output value
Linear Regression

Hypothesis family: Linear functions
Linear Regression

(x, y=f(x)), x: house size, y: house price

Berkeley house prices, 2009
Linear regression = fitting a straight line/hyperplane

Prediction: $h_w(x) = w_0 + w_1 x$

Berkeley house prices, 2009
Prediction error

Error on one instance: $y - h_w(x)$
Find $w$

- Define loss function

- Find $w^*$ to minimize loss function
Least squares: Minimizing squared error

- L2 loss function: sum of squared errors over all examples
  - Loss = ____________________________
- We want the weights $w^*$ that minimize loss
- At $w^*$ the derivatives of loss w.r.t. each weight are zero:
  - $\frac{\partial \text{Loss}}{\partial w_0} = ____________________________$
  - $\frac{\partial \text{Loss}}{\partial w_1} = ____________________________$
- Exact solutions for $N$ examples:
  - $w_1 = [N\Sigma_j x_j y_j - (\Sigma_j x_j)(\Sigma_j y_j)]/[N\Sigma_j x_j^2 - (\Sigma_j x_j)^2]$ and $w_0 = 1/N [\Sigma_j y_j - w_1 \Sigma_j x_j]$
- For the general case where $x$ is an $n$-dimensional vector
  - $X$ is the data matrix (all the data, one example per row); $y$ is the column of labels
  - $w^* = (X^T X)^{-1} X^T y$
Least squares: Minimizing squared error

- **L2 loss function**: sum of squared errors over all examples
  - Loss = \( \sum_j (y_j - h_w(x_j))^2 = \sum_j (y_j - (w_0 + w_1 x_j))^2 \)

- We want the weights \( w^* \) that minimize loss.

- At \( w^* \) the derivatives of loss w.r.t. each weight are zero:
  - \( \frac{\partial \text{Loss}}{\partial w_0} = -2 \sum_j (y_j - (w_0 + w_1 x_j)) = 0 \)
  - \( \frac{\partial \text{Loss}}{\partial w_1} = -2 \sum_j (y_j - (w_0 + w_1 x_j)) x_j = 0 \)

- Exact solutions for \( N \) examples:
  - \( w_1 = \frac{N \sum_j x_j y_j - (\sum_j x_j)(\sum_j y_j)}{N \sum_j x_j^2 - (\sum_j x_j)^2} \) and \( w_0 = \frac{1}{N} [\sum_j y_j - w_1 \sum_j x_j] \)

- For the general case where \( x \) is an n-dimensional vector
  - \( X \) is the data matrix (all the data, one example per row); \( y \) is the column of labels
  - \( w^* = (X^T X)^{-1} X^T y \)
Regression vs Classification

- Linear regression when output is binary, $y \in \{0, 1\}$
  - $h_w(x) = w_0 + w_1 x$

- Linear classification
  - Used with discrete output values
  - Threshold a linear function
  - $h_w(x) = 1$, if $w_0 + w_1 x \geq 0$
  - $h_w(x) = 0$, if $w_0 + w_1 x < 0$
  - Activation function $g$
Threshold perceptron as linear classifier
A threshold perceptron is a single unit that outputs
- \[ y = h_w(x) = 1 \text{ when } w \cdot x \geq 0 \]
  \[ = 0 \text{ when } w \cdot x < 0 \]

In the input vector space
- Examples are points \( x \)
- The equation \( w \cdot x = 0 \) defines a hyperplane
- One side corresponds to \( y = 1 \)
- Other corresponds to \( y = 0 \)
Dear Stuart, I'm leaving Macrosoft to return to academia. The money is great here but I prefer to be free to do my own research; and I really love teaching undergrads!

Do I need to finish my BA first before applying?

Best wishes

Bill

\[
\begin{align*}
    w_0 & : -3 \\
    w_{\text{free}} & : 4 \\
    w_{\text{money}} & : 2 \\
    x_0 & : 1 \\
    x_{\text{free}} & : 1 \\
    x_{\text{money}} & : 1 \\
    w \cdot x &= -3x_1 + 4x_1 + 2x_1 = 3
\end{align*}
\]
Need a different solution than before given the characteristic of perceptron
Perceptron learning rule

- If true $y \neq h_w(x)$ (an error), adjust the weights
- If $w.x < 0$ but the output should be $y=1$
  - This is called a false negative
  - Should increase weights on positive inputs
  - Should decrease weights on negative inputs
- If $w.x > 0$ but the output should be $y=0$
  - This is called a false positive
  - Should decrease weights on positive inputs
  - Should increase weights on negative inputs
- The perceptron learning rule does this:
  - $w \leftarrow w + \alpha (y - h_w(x)) x$
Perceptron Learning Rule (Different setup)

- Start with weights = 0
- For each training instance:
  - Classify with current weights
    
    \[ y = \begin{cases} 
    +1 & \text{if } w \cdot x \geq 0 \\
    -1 & \text{if } w \cdot x < 0 
    \end{cases} \]

- If wrong: adjust the weight vector by adding or subtracting the feature vector. Subtract if y is -1.

\[ w = w + y \cdot x \]

\[ y = h_w(x) = 1 \text{ when } w \cdot x \geq 0 \]
\[ = -1 \text{ when } w \cdot x < 0 \]
Dear Stuart, I wanted to let you know that I have decided to leave Macrosoft and return to academia. The money is great here but I prefer to be free to pursue more interesting research and I really love teaching undergraduates! Do I need to finish my BA first before applying?

Best wishes
Bill

\[ w \leftarrow w + \alpha (y - h_w(x)) x \]
\[ \alpha = 0.5 \]

\[ w \leftarrow (-3,4,2) + 0.5 (0 - 1) (1,1,1) \]
\[ = (-3.5,3.5,1.5) \]
A learning problem is **linearly separable** iff there is some hyperplane exactly separating positive from negative examples.

Convergence: if the training data are **separable**, perceptron learning applied repeatedly to the training set will eventually converge to a perfect separator.
Example: Earthquakes vs nuclear explosions

63 examples, 657 updates required
A learning problem is **linearly separable** iff there is some hyperplane exactly separating +ve from –ve examples.

Convergence: if the training data are separable, perceptron learning applied repeatedly to the training set will eventually converge to a perfect separator.

Convergence: if the training data are **non-separable**, perceptron learning will converge to a minimum-error solution provided the learning rate $\alpha$ is decayed appropriately (e.g., $\alpha=1/t$).
Perceptron learning with fixed $\alpha$

71 examples, 100,000 updates
fixed $\alpha = 0.2$, no convergence
Perceptron learning with decaying $\alpha$

71 examples, 100,000 updates
decaying $\alpha = 1000/(1000 + t)$, near-convergence
Non-Separable Case

Even the best linear boundary makes at least one mistake.
Other Linear Classifiers

- Perceptron is perfectly happy as long as it separates the training data.

- Logistic Regression:
  \[ g_{\text{sigmoid}}(x) = \frac{1}{1 + e^{-x}} \]

- Support Vector Machines (SVM):
  - Maximize margin between boundary and nearest points.
Logistic Regression

- Sigmoid function

\[ \phi(z) = \frac{1}{1 + e^{-z}} \]