Lecture 6 – Logistic Regression

Dan Sheldon

September 26, 2012

Plan for Today

- MATLAB & HW 2
- Review
  - Normal equations by matrix calculus
  - Gradient descent
  - Feature normalization
- Logistic regression

Aside: Matrix Calculus

Succinct (and cool!) way to solve for normal equations:

$$
\begin{aligned}
0 &= \frac{d}{d\mathbf{w}}(X\mathbf{w}-\mathbf{y})^T(X\mathbf{w}-\mathbf{y}) \\
0 &= 2(X\mathbf{w}-\mathbf{y})^T X \\
0 &= X^T(X\mathbf{w}-\mathbf{y}) \\
X^T X\mathbf{w} &= X^T\mathbf{y} \\
\mathbf{w} &= (X^T X)^{-1} X^T\mathbf{y}
\end{aligned}
$$

Review of Gradient Descent

Algorithm:

1. Initialize $w_0, w_1, \ldots, w_d$ arbitrarily
2. Repeat until convergence

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\mathbf{w}), \qquad j = 0, \ldots, d.$$

In matrix-vector notation:

Feature Normalization

- Features may have very different numeric ranges

Width	Thickness	Height	# Pages	Hardcover	Weight
8	1.8	10	1152	1	4.4
8	0.9	9	584	1	2.7
7	1.8	9.2	738	1	3.9
6.4	1.5	9.5	512	1	1.8

- Advice: normalize your features!
  - Subtract mean (center)
  - Divide by standard deviation (scale)

Feature Normalization

For each feature j, compute

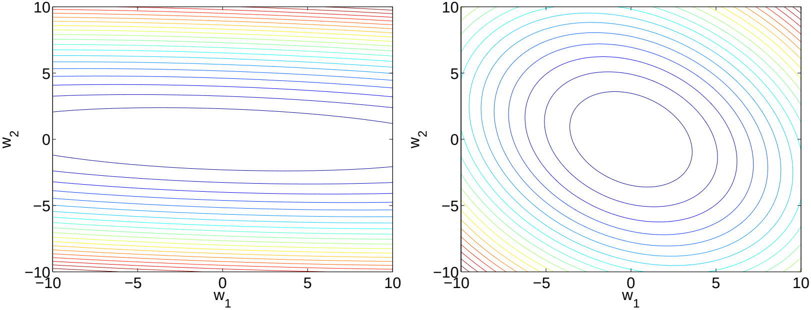
$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}, \qquad \sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N}(x_{i,j} - \mu_j)^2$$

Then, subtract μ_j and divide by σ_j:

$$x_{i,j} \leftarrow (x_{i,j} - \mu_j)/\sigma_j$$

Feature Normalization

Example: cost function contours before and after normalization



Main Topic: Logistic Regression

- Classification
- Model
- Cost function
- Gradient descent
- Linear classifiers and decision boundaries

Classification

- Input: $\mathbf{x} \in \mathbb{R}^d$
- Output: $y \in \{0, 1\}$
- Model (hypothesis class): ?
- Cost function: ?

Classification as regression?

The Model

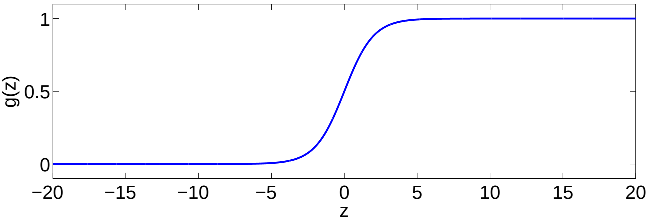
Exercise: fix the linear regression model

$$h_{\mathbf{w}}(\mathbf{x}) = g(\mathbf{w}^T \mathbf{x}), \qquad g : \mathbb{R} \to [0, 1].$$

What should g look like?

Logistic Function

$$g(z) = \frac{1}{1 + e^{-z}}$$



- This is called the *logistic* or *sigmoid* function

$$g(z) = \text{logistic}(z) = \text{sigmoid}(z)$$

The Model

Put it together

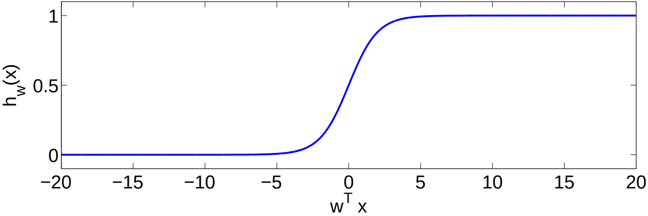
$$h_{\mathbf{w}}(\mathbf{x}) = \text{logistic}(\mathbf{w}^T\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}}$$

Nuance:

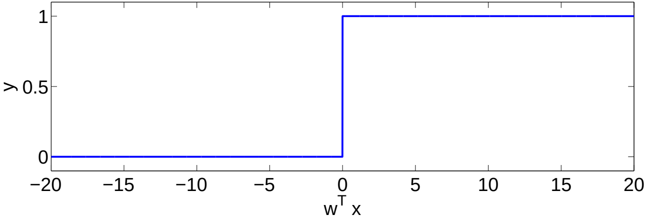
- Output is in $[0, 1]$, not $\{0, 1\}$.
- Interpret as probability

Hypothesis vs. Prediction Rule

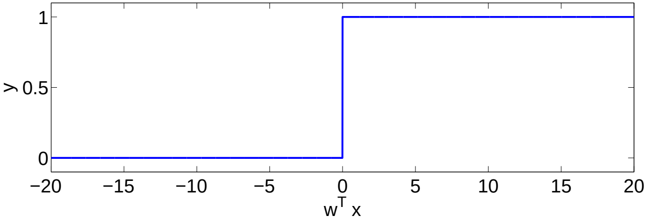
Hypothesis (use during learning)



Prediction rule (for predictions!)



Prediction Rule



$$y = \begin{cases} 0 & h_{\mathbf{w}}(\mathbf{x}) < 1/2 \qquad (\mathbf{w}^T\mathbf{x} < 0) \\ 1 & h_{\mathbf{w}}(\mathbf{x}) \geq 1/2 \qquad (\mathbf{w}^T\mathbf{x} \geq 0). \end{cases}$$

Cost Function

Can we used squared error?

$$J(\mathbf{w}) = \sum_i (h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2$$

R&N does this. But we want to do better.

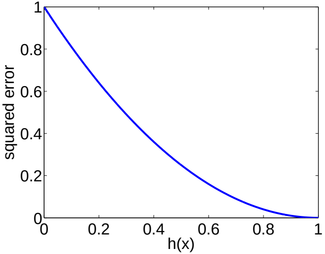
Let's define cost for a single example. E.g., for squared error:

$$J(\mathbf{w}) = \sum_i \text{cost}(h_{\mathbf{w}}(\mathbf{x}_i), y_i)$$

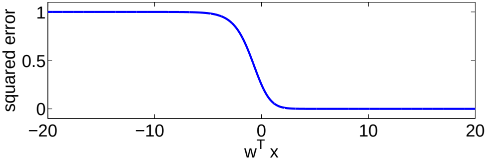
$$\text{cost}(h_{\mathbf{w}}(\mathbf{x}), y) = (h_{\mathbf{w}}(\mathbf{x}) - y)^2$$

Cost Function

Suppose $y = 1$. Squared error looks like this

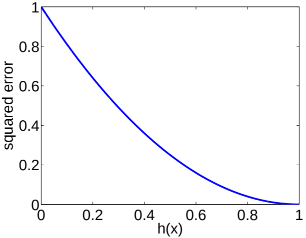


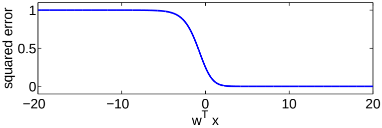
If we undo the logistic transform, it looks like this



Cost Function

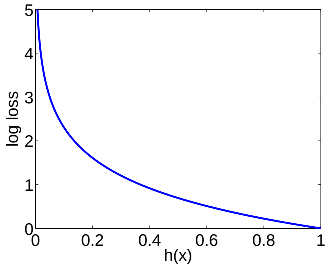
Exercise: fix these

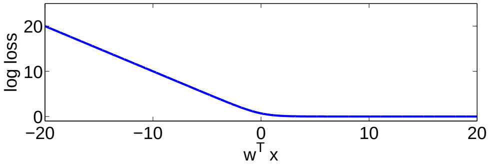




Log Loss $(y = 1)$

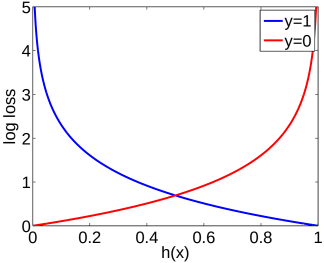
$$\text{cost}(h(\mathbf{x}), 1) = -\log h(\mathbf{x})$$

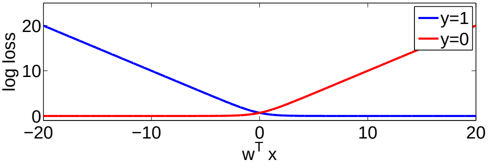




Log Loss

$$\text{cost}(h(\mathbf{x}), y) = \begin{cases} -\log h(\mathbf{x}) & y = 1 \\ -\log(1 - h(\mathbf{x})) & y = 0 \end{cases}$$





Equivalent Expression for Log-Loss

$$\mathsf{cost}(h(\mathbf{x}), y) = \begin{cases} -\log h(\mathbf{x}) & y = 1 \\ -\log(1 - h(\mathbf{x})) & y = 0 \end{cases}$$

$$\mathsf{cost}(h(\mathbf{x}), y) = -y \log h(\mathbf{x}) - (1 - y) \log(1 - h(\mathbf{x}))$$

Review so far

- Input: $\mathbf{x} \in \mathbb{R}^d$
- Output: $y \in \{0, 1\}$
- Model (hypothesis class)

$$h_{\mathbf{w}}(\mathbf{x}) = \text{logistic}(\mathbf{w}^T\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}}$$

- Cost function (log loss):

$$J(\mathbf{w}) = \sum_{i=1}^{N} \Big(-y_i \log h_{\mathbf{w}}(\mathbf{x}_i) - (1 - y_i) \log(1 - h_{\mathbf{w}}(\mathbf{x}_i)) \Big)$$

Gradient Descent for Logistic Regression

1. Initialize $w_0, w_1, \ldots, w_d$ arbitrarily
2. Repeat until convergence

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\mathbf{w}), \qquad j = 0, \ldots, d.$$

Partial derivatives for logistic regression:

$$\frac{\partial}{\partial w_j} J(\mathbf{w}) = 2 \sum_{i=1}^{N} (h_{\mathbf{w}}(\mathbf{x}_i) - y_i) x_{i,j}$$

(Same as linear regression! But $h_{\mathbf{w}}(\mathbf{x})$ is different)

Decision Boundaries

Example from R&N (Fig. 18.15).

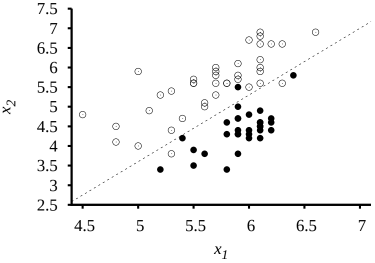
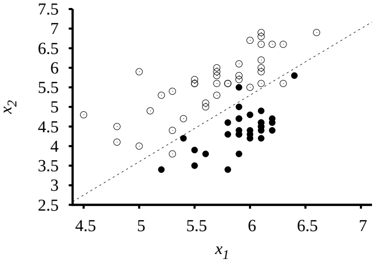


Figure: Earthquakes (white circles) vs. nuclear explosions (black circles) by body wave magnitude ($x1$) and surface wave magnitude ($x2$)

Decision Boundaries



E.g., suppose hypothesis is

$$h(x_1, x_2) = \text{logistic}(1.7x_1 - x_2 - 4.9)$$

Predict nuclear explosion if:

$$1.7x_1 - x_2 - 4.9 \geq 0$$

$$x_2 \leq 1.7x_1 - 4.9$$

Linear Classifiers

Predict

$$y = \begin{cases} 0 & \text{if } \mathbf{w}^T\mathbf{x} < 0, \\ 1 & \text{if } \mathbf{w}^T\mathbf{x} \geq 0. \end{cases}$$

Many other learning algorithms use linear classification rules

- Perceptron
- Support vector machines (SVMs)
- Linear discriminants

Nonlinear Decision Boundaries by Feature Expansion

Example (Ng)

$$(x_1, x_2) \mapsto (1, x_1, x_2, x_1^2, x_2^2, x_1 x_2),$$
$$\mathbf{w} = \begin{bmatrix} -1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}^T$$

Exercise: what does decision boundary look like in (x_1, x_2) plane?