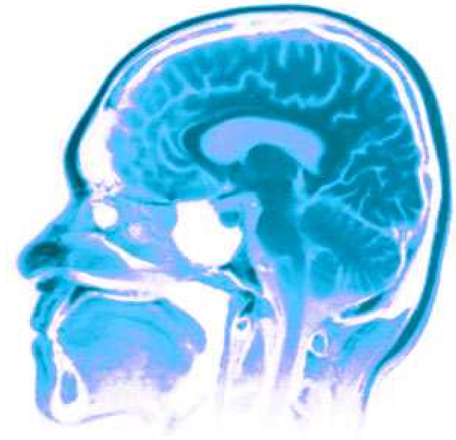
CPSC540



Logistic Regression and Neuron Models

Nando de Freitas

October, 2011

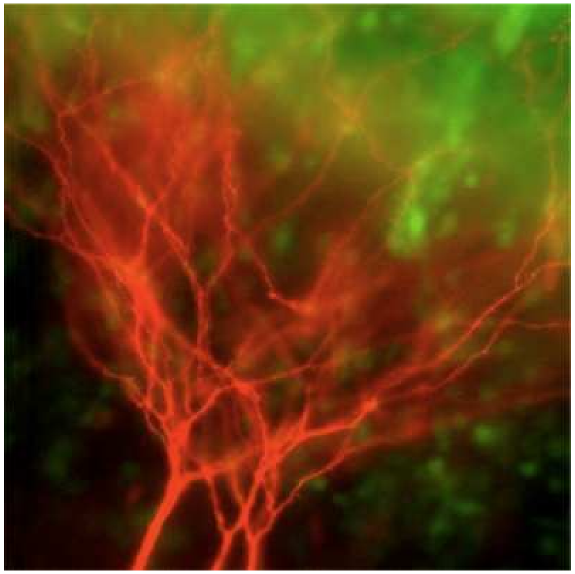
University of British Columbia

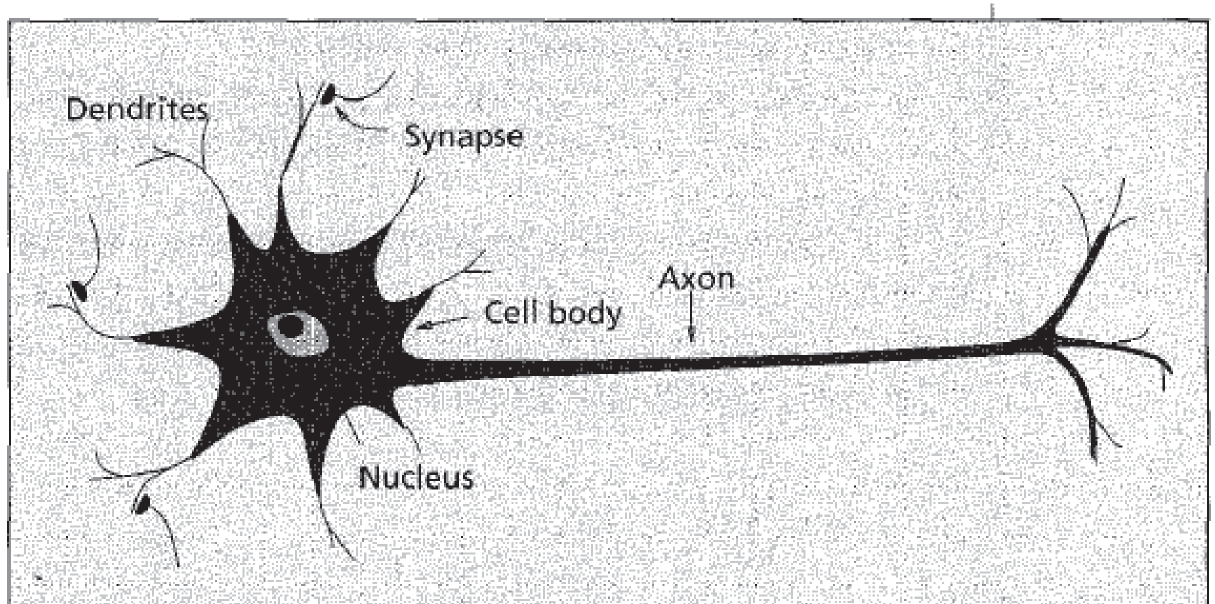
Outline of the lecture

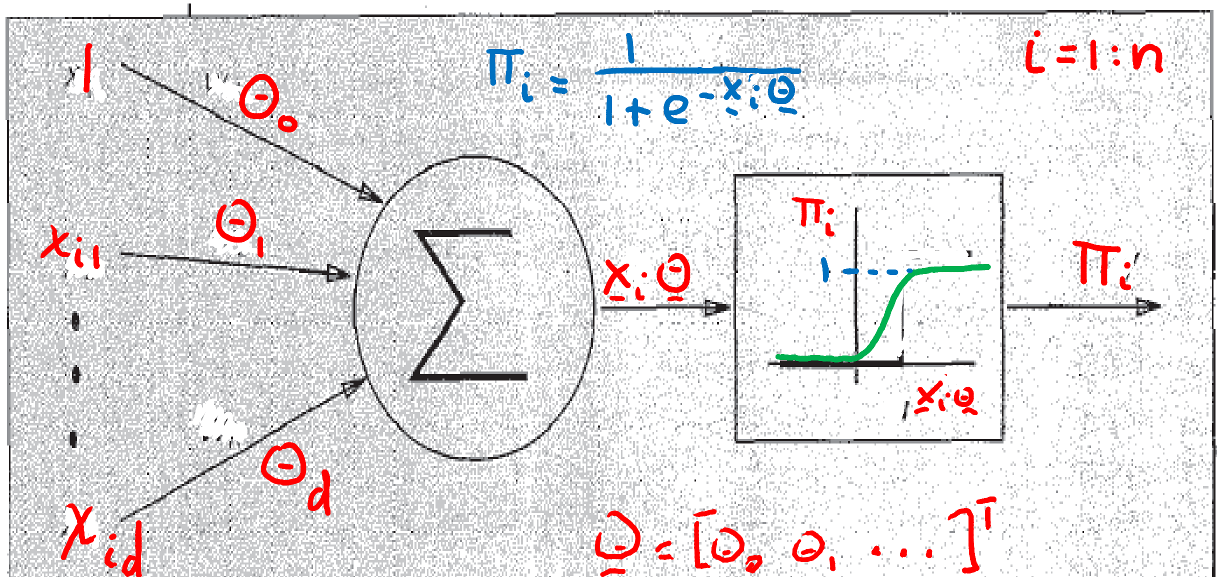
This lecture describes the construction of binary classifiers using a technique called Logistic Regression. The objective is for you to learn:

- ❑ How to apply logistic regression to discriminate between two classes.
- ❑ How to formulate the logistic regression likelihood.
- ❑ How to derive the gradient and Hessian of logistic regression on your own.
- ❑ How to incorporate the gradient vector and Hessian matrix into Newton's optimization algorithm so as to come up with an algorithm for logistic regression, which we'll call IRLS.

McCulloch-Pitts model of a neuron





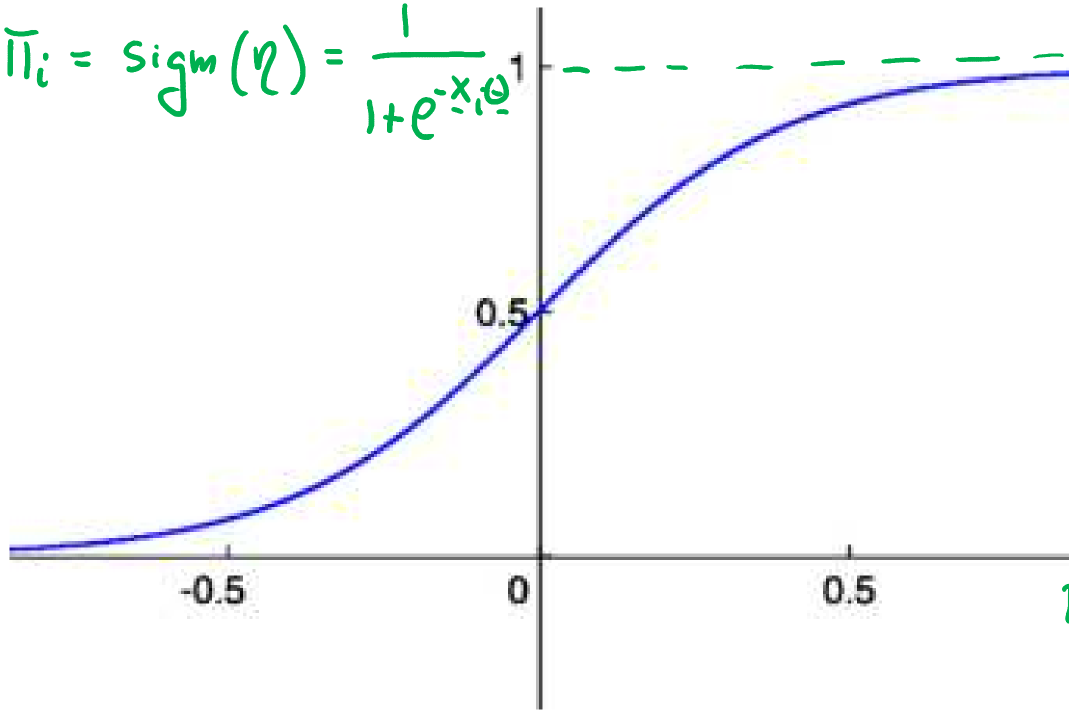


Sigmoid function

$\text{sigm}(\eta)$ refers to the **sigmoid** function, also known as the **logistic** or **logit** function:

$$\text{sigm}(\eta) = \frac{1}{1+e^{-\eta}} = \frac{e^{\eta}}{e^{\eta}+1}$$

$\pi_i = \text{sigm}(\eta) = \frac{1}{1+e^{-\underline{x}_i\underline{\theta}}}$



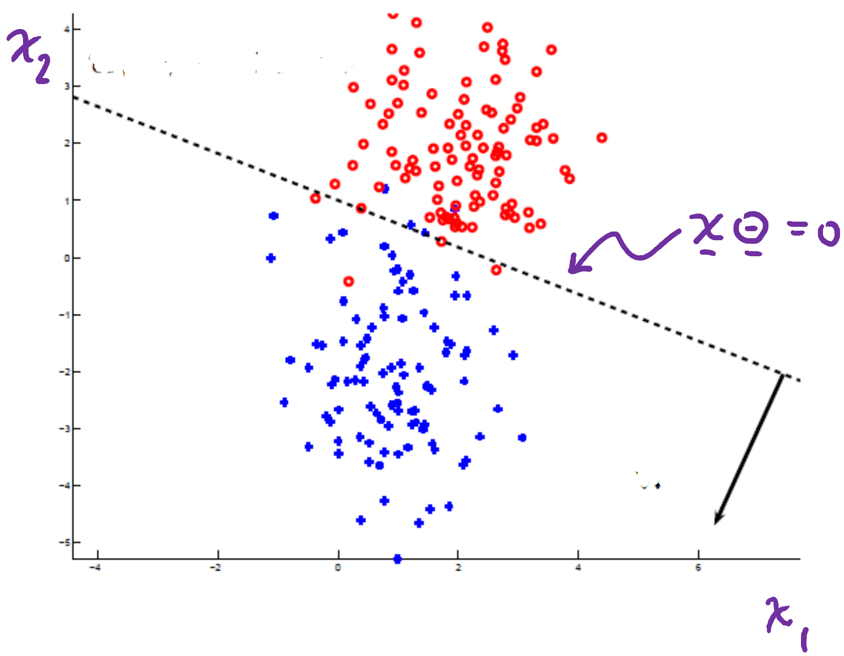
$\eta = \underline{x}_i\underline{\theta}$

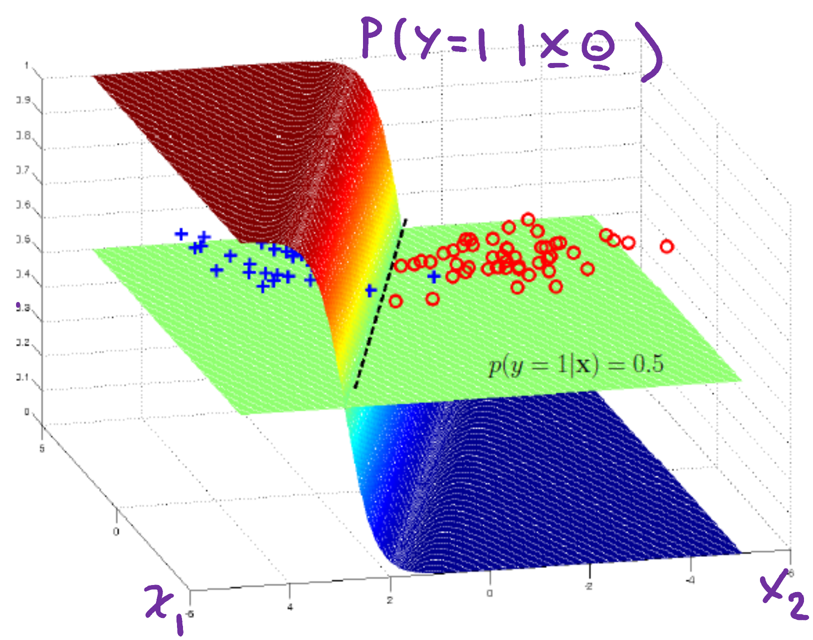
Linear separating hyper-plane

$$P(y_i=1 \mid \underline{x}_i, \underline{\theta}) = \text{sigm}(\underline{x}_i\underline{\theta}) = \frac{1}{2} \quad \overset{\text{when}}{\Longleftrightarrow} \quad \underline{x}_i\underline{\theta} = 0$$

EQUATION OF A PLANE.

i.e. $\frac{1}{1+e^{-0}} = \frac{1}{2}$





[Greg Shakhnarovich]

Logistic regression

The logistic regression model specifies the probability of a binary output $y_i \in \{0, 1\}$ given the input $\mathbf{x}_i$ as follows:

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \prod_{i=1}^{n} \mathrm{Ber}(y_i|\mathrm{sigm}(\mathbf{x}_i\boldsymbol{\theta})) = \prod_{i=1}^{n} \pi_i^{y_i} (1-\pi_i)^{1-y_i}$$

$$= \prod_{i=1}^{n} \left[\frac{1}{1+e^{-\mathbf{x}_i\boldsymbol{\theta}}}\right]^{y_i} \left[1 - \frac{1}{1+e^{-\mathbf{x}_i\boldsymbol{\theta}}}\right]^{1-y_i}$$

where $\mathbf{x}_i\boldsymbol{\theta} = \theta_0 + \sum_{j=1}^{d} \theta_j x_{ij}$

$$\pi_i = \frac{1}{1+e^{-x_i\theta}}$$

Gradient and Hessian of binary logistic regression

The gradient and Hessian of the negative loglikelihood, $J(\boldsymbol{\theta}) = -\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})$, are given by:

$$\begin{aligned}
\mathbf{g}(\mathbf{w}) &= \frac{d}{d\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{i=1}^{n} \mathbf{x}_i^T (\pi_i - y_i) = \mathbf{X}^T (\boldsymbol{\pi} - \mathbf{y}) \\
\mathbf{H} &= \frac{d}{d\boldsymbol{\theta}} \mathbf{g}(\boldsymbol{\theta})^T = \sum_i \pi_i (1 - \pi_i) \mathbf{x}_i \mathbf{x}_i^T = \mathbf{X}^T \mathrm{diag}(\pi_i(1 - \pi_i)) \mathbf{X}
\end{aligned}$$

where $\pi_i = \mathrm{sigm}(\mathbf{x}_i \boldsymbol{\theta})$

One can show that $\mathbf{H}$ is positive definite; hence the NLL is **convex** and has a unique global minimum.

To find this minimum, we turn to batch optimization.

Iteratively reweighted least squares (IRLS)

For binary logistic regression, recall that the gradient and Hessian of the negative log-likelihood are given by

$$\begin{aligned}
\mathbf{g}_k &= \mathbf{X}^T(\boldsymbol{\pi}_k - \mathbf{y}) \\
\mathbf{H}_k &= \mathbf{X}^T\mathbf{S}_k\mathbf{X} \\
\mathbf{S}_k &:= \mathrm{diag}(\pi_{1k}(1-\pi_{1k}), \ldots, \pi_{nk}(1-\pi_{nk})) \\
\pi_{ik} &= \mathrm{sigm}(\mathbf{x}_i\boldsymbol{\theta}_k)
\end{aligned}$$

The Newton update at iteration $k+1$ for this model is as follows (using $\eta_k = 1$, since the Hessian is exact):

$$\begin{aligned}
\boldsymbol{\theta}_{k+1} &= \boldsymbol{\theta}_k - \mathbf{H}^{-1}\mathbf{g}_k \\
&= \boldsymbol{\theta}_k + (\mathbf{X}^T\mathbf{S}_k\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{y} - \boldsymbol{\pi}_k) \\
&= (\mathbf{X}^T\mathbf{S}_k\mathbf{X})^{-1}\left[(\mathbf{X}^T\mathbf{S}_k\mathbf{X})\boldsymbol{\theta}_k + \mathbf{X}^T(\mathbf{y} - \boldsymbol{\pi}_k)\right] \\
&= (\mathbf{X}^T\mathbf{S}_k\mathbf{X})^{-1}\mathbf{X}^T\left[\mathbf{S}_k\mathbf{X}\boldsymbol{\theta}_k + \mathbf{y} - \boldsymbol{\pi}_k\right]
\end{aligned}$$

Iteratively reweighted least squares (IRLS)

```
from __future__ import division
import numpy as np

def logistic(a):
        return 1.0 / (1 + np.exp(-a))

def irls(X, y):
        theta = np.zeros(X.shape[1])
        theta_ = np.inf
        while max(abs(theta-theta_)) > 1e-6:
                a = np.dot(X, theta)
                pi = logistic(a)
                SX = X * (pi - pi*pi).reshape(-1,1)
                XSX = np.dot(X.T, SX)
                SXtheta = np.dot(SX, theta)
                theta_ = theta
                theta = np.linalg.solve(XSX, np.dot(X.T, SXtheta + y - pi))
        return theta
```

Iteratively reweighted least squares (IRLS)

```
if __name__ == "__main__":
        # load the data.
        X = np.loadtxt('spambase.data', delimiter=',', skiprows=1)

        # split X/y and add a constant column to X.
        y = X[:,-1]
        X = X[:,:-1]
        X = np.c_[np.ones(X.shape[0]), X]
        Xtrain, Xtest = X[0:4000], X[4000:]
        ytrain, ytest = y[0:4000], y[4000:]

        theta = irls(Xtrain, ytrain)

        train_rate = sum((logistic(np.dot(Xtrain, theta)) > .5) != ytrain) / ytrain.size
        test_rate  = sum((logistic(np.dot(Xtest,  theta)) > .5) != ytest)  / ytest.size

        print "Training data misclassification rate: %.4f" % train_rate
        print "Test data misclassification rate:     %.4f" % test_rate
```

Next lecture

In the next lecture, we consider a generalization of logistic regression, with many logistic units, called multi-layer perceptron (MLP). MLPs are the most commonly used type of artificial neural networks.