# Logistic Regression

- Learn the conditional distribution P($y$ | **x**)
- Let $p_y$(**x**; **w**) be our estimate of P($y$ | **x**), where **w** is a vector of adjustable parameters. Assume only two classes $y$ = 0 and $y$ = 1, and

$$p_1(\mathbf{x};\mathbf{w}) = \frac{\exp \mathbf{w}\cdot\mathbf{x}}{1+\exp \mathbf{w}\cdot\mathbf{x}}.$$

$$p_0(\mathbf{x};\mathbf{w}) = 1 - p_1(\mathbf{x};\mathbf{w}).$$

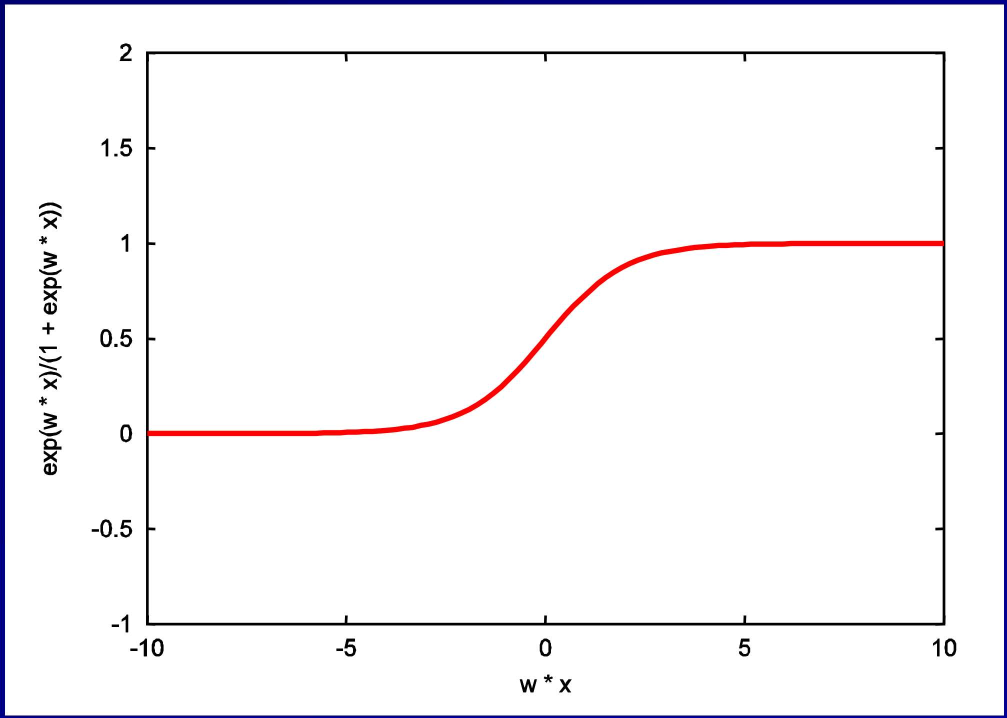
- On the homework, you will show that this is equivalent to

$$\log \frac{p_1(\mathbf{x};\mathbf{w})}{p_0(\mathbf{x};\mathbf{w})} = \mathbf{w}\cdot\mathbf{x}.$$

- In other words, the log odds of class 1 is a linear function of **x**.

# Why the exp function?

- One reason: A linear function has a range from [–∞, ∞] and we need to force it to be positive and sum to 1 in order to be a probability:



# Deriving a Learning Algorithm

- Since we are fitting a conditional probability distribution, we no longer seek to minimize the loss on the training data. Instead, we seek to find the probability distribution *h* that is most likely given the training data
- Let S be the training sample. Our goal is to find *h* to maximize P(*h* | S):

$$\begin{aligned}
\operatorname*{argmax}_h P(h|S) &= \operatorname*{argmax}_h \frac{P(S|h)P(h)}{P(S)} && \text{by Bayes' Rule} \\
&= \operatorname*{argmax}_h P(S|h)P(h) && \text{because } P(S) \text{ doesn't depend on } h \\
&= \operatorname*{argmax}_h P(S|h) && \text{if we assume } P(h) = \text{uniform} \\
&= \operatorname*{argmax}_h \log P(S|h) && \text{because log is monotonic}
\end{aligned}$$

The distribution P(S|*h*) is called the <u>likelihood function</u>. The log likelihood is frequently used as the objective function for learning. It is often written as $\ell(\mathbf{w})$.

The *h* that maximizes the likelihood on the training data is called the <u>maximum likelihood estimator (MLE)</u>

# Computing the Likelihood

- In our framework, we assume that each training example ($\mathbf{x}_i, y_i$) is drawn from the same (but unknown) probability distribution P($\mathbf{x}, y$). This means that the log likelihood of S is the sum of the log likelihoods of the individual training examples:

$$\begin{aligned} \log P(S|h) &= \log \prod_i P(\mathbf{x}_i, y_i|h) \\ &= \sum_i \log P(\mathbf{x}_i, y_i|h) \end{aligned}$$

# Computing the Likelihood (2)

- Recall that *any* joint distribution P(a,b) can be factored as P(a|b) P(b). Hence, we can write

$$\begin{aligned}\operatorname*{argmax}_h \log P(S|h) &= \operatorname*{argmax}_h \sum_i \log P(\mathbf{x}_i, y_i|h) \\ &= \operatorname*{argmax}_h \sum_i \log P(y_i|\mathbf{x}_i, h) P(\mathbf{x}_i|h)\end{aligned}$$

- In our case, P(**x** | *h*) = P(**x**), because it does not depend on *h*, so

$$\begin{aligned}\operatorname*{argmax}_h \log P(S|h) &= \operatorname*{argmax}_h \sum_i \log P(y_i|\mathbf{x}_i, h) P(\mathbf{x}_i|h) \\ &= \operatorname*{argmax}_h \sum_i \log P(y_i|\mathbf{x}_i, h)\end{aligned}$$

# Log Likelihood for Conditional Probability Estimators

- We can express the log likelihood in a compact form known as the cross entropy.
- Consider an example ($\mathbf{x}_i, y_i$)
  - If $y_i = 0$, the log likelihood is log $[1 - p_1(\mathbf{x}; \mathbf{w})]$
  - if $y_i = 1$, the log likelihood is log $[p_1(\mathbf{x}; \mathbf{w})]$
- These cases are mutually exclusive, so we can combine them to obtain:

  $\ell(y_i; \mathbf{x}_i, \mathbf{w}) = \log P(y_i \mid \mathbf{x}_i, \mathbf{w}) = (1 - y_i) \log[1 - p_1(\mathbf{x}_i; \mathbf{w})] + y_i \log p_1(\mathbf{x}_i; \mathbf{w})$
- The goal of our learning algorithm will be to find **w** to maximize

  $J(\mathbf{w}) = \sum_i \ell(y_i; \mathbf{x}_i, \mathbf{w})$

# Fitting Logistic Regression by Gradient Ascent

$$\frac{\partial J(\mathbf{w})}{\partial w_j} = \sum_i \frac{\partial}{\partial w_j} \ell(y_i; \mathbf{x}_i, \mathbf{w})$$

$$\begin{aligned}
\frac{\partial}{\partial w_j} \ell(y_i; \mathbf{x}_i, \mathbf{w}) &= \frac{\partial}{\partial w_j}((1-y_i)\log[1-p_1(\mathbf{x}_i;\mathbf{w})] + y_1 \log p_1(\mathbf{x}_i;\mathbf{w})) \\
&= (1-y_i)\frac{1}{1-p_1(\mathbf{x}_i;\mathbf{w})}\left(-\frac{\partial p_1(\mathbf{x}_i;\mathbf{w})}{\partial w_j}\right) + y_i \frac{1}{p_1(\mathbf{x}_i;\mathbf{w})}\left(\frac{\partial p_1(\mathbf{x}_i;\mathbf{w})}{\partial w_j}\right) \\
&= \left[\frac{y_i}{p_1(\mathbf{x}_i;\mathbf{w})} - \frac{(1-y_i)}{1-p_1(\mathbf{x}_i;\mathbf{w})}\right]\left(\frac{\partial p_1(\mathbf{x}_i;\mathbf{w})}{\partial w_j}\right) \\
&= \left[\frac{y_i(1-p_1(\mathbf{x}_i;\mathbf{w})) - (1-y_i)p_1(\mathbf{x}_i;\mathbf{w})}{p_1(\mathbf{x}_i;\mathbf{w})(1-p_1(\mathbf{x}_i;\mathbf{w}))}\right]\left(\frac{\partial p_1(\mathbf{x}_i;\mathbf{w})}{\partial w_j}\right) \\
&= \left[\frac{y_i - p_1(\mathbf{x}_i;\mathbf{w})}{p_1(\mathbf{x}_i;\mathbf{w})(1-p_1(\mathbf{x}_i;\mathbf{w}))}\right]\left(\frac{\partial p_1(\mathbf{x}_i;\mathbf{w})}{\partial w_j}\right)
\end{aligned}$$

# Gradient Computation (continued)

- Note that $p_1$ can also be written as

$$p_1(\mathbf{x}_i;\mathbf{w}) = \frac{1}{(1+\text{exp}[-\mathbf{w}\cdot\mathbf{x}_i])}.$$

- From this, we obtain:

$$\begin{aligned}
\frac{\partial p_1(\mathbf{x}_i;\mathbf{w})}{\partial w_j} &= -\frac{1}{(1+\text{exp}[-\mathbf{w}\cdot\mathbf{x}_i])^2}\frac{\partial}{\partial w_j}(1+\text{exp}[-\mathbf{w}\cdot\mathbf{x}_i]) \\
&= -\frac{1}{(1+\text{exp}[-\mathbf{w}\cdot\mathbf{x}_i])^2}\text{exp}[-\mathbf{w}\cdot\mathbf{x}_i]\frac{\partial}{\partial w_j}(-\mathbf{w}\cdot\mathbf{x}_i) \\
&= -\frac{1}{(1+\text{exp}[-\mathbf{w}\cdot\mathbf{x}_i])^2}\text{exp}[-\mathbf{w}\cdot\mathbf{x}_i](-x_{ij}) \\
&= p_1(\mathbf{x}_i;\mathbf{w})(1-p_1(\mathbf{x}_i;\mathbf{w}))x_{ij}
\end{aligned}$$

# Completing the Gradient Computation

- The gradient of the log likelihood of a single point is therefore

$$\begin{aligned}\frac{\partial}{\partial w_j}\ell(y_i;\mathbf{x}_i,\mathbf{w}) &= \left[\frac{y_i - p_1(\mathbf{x}_i;\mathbf{w})}{p_1(\mathbf{x}_i;\mathbf{w})(1-p_1(\mathbf{x}_i;\mathbf{w}))}\right]\left(\frac{\partial p_1(\mathbf{x}_i;\mathbf{w})}{\partial w_j}\right)\\ &= \left[\frac{y_i - p_1(\mathbf{x}_i;\mathbf{w})}{p_1(\mathbf{x}_i;\mathbf{w})(1-p_1(\mathbf{x}_i;\mathbf{w}))}\right] p_1(\mathbf{x}_i;\mathbf{w})(1-p_1(\mathbf{x}_i;\mathbf{w}))x_{ij}\\ &= (y_i - p_1(\mathbf{x}_i;\mathbf{w}))x_{ij}\end{aligned}$$

- The overall gradient is

$$\frac{\partial J(\mathbf{w})}{\partial w_j} = \sum_i (y_i - p_1(\mathbf{x}_i;\mathbf{w}))x_{ij}$$

# Batch Gradient Ascent for Logistic Regression

**Given:** training examples $(\mathbf{x}_i, y_i)$, $i = 1 \ldots N$

**Let** $\mathbf{w} = (0, 0, 0, 0, \ldots, 0)$ be the initial weight vector.

**Repeat** until convergence

  **Let** $\mathbf{g} = (0, 0, \ldots, 0)$ be the gradient vector.

  **For** $i = 1$ **to** $N$ **do**

    $p_i = 1/(1 + \exp[-\mathbf{w} \cdot \mathbf{x}_i])$

    $\text{error}_i = y_i - p_i$

    **For** $j = 1$ **to** $n$ **do**

      $g_j = g_j + \text{error}_i \cdot x_{ij}$

  $\mathbf{w} := \mathbf{w} + \eta \mathbf{g}$   step in direction of increasing gradient

- An online gradient ascent algorithm can be constructed, of course
- Most statistical packages use a second-order (Newton-Raphson) algorithm for faster convergence. Each iteration of the second-order method can be viewed as a weighted least squares computation, so the algorithm is known as Iteratively-Reweighted Least Squares (IRLS)

# Logistic Regression Implements a Linear Discriminant Function

- In the 2-class 0/1 loss function case, we should predict ŷ = 1 if

$$
\begin{aligned}
E_{y|\mathbf{x}}[L(0,y)] &> E_{y|\mathbf{x}}[L(1,y)] \\
\sum_y P(y|\mathbf{x})L(0,y) &> \sum_y P(y|\mathbf{x})L(1,y) \\
P(y=0|\mathbf{x})L(0,0) + P(y=1|\mathbf{x})L(0,1) &> P(y=0|\mathbf{x})L(1,0) + P(y=1|\mathbf{x})L(1,1) \\
P(y=1|\mathbf{x}) &> P(y=0|\mathbf{x}) \\
\frac{P(y=1|\mathbf{x})}{P(y=0|\mathbf{x})} &> 1 \qquad \text{if } P(y=0|X) \neq 0 \\
\log \frac{P(y=1|\mathbf{x})}{P(y=0|\mathbf{x})} &> 0 \\
\mathbf{w} \cdot \mathbf{x} &> 0
\end{aligned}
$$

- A similar derivation can be done for arbitrary L(0,1) and L(1,0).

# Extending Logistic Regression to K > 2 classes

- Choose class K to be the "reference class" and represent each of the other classes as a logistic function of the odds of class *k* versus class K:

$$
\begin{aligned}
\log \frac{P(y=1|\mathbf{x})}{P(y=K|\mathbf{x})} &= \mathbf{w}_1 \cdot \mathbf{x} \\
\log \frac{P(y=2|\mathbf{x})}{P(y=K|\mathbf{x})} &= \mathbf{w}_2 \cdot \mathbf{x} \\
&\vdots \\
\log \frac{P(y=K-1|\mathbf{x})}{P(y=K|\mathbf{x})} &= \mathbf{w}_{K-1} \cdot \mathbf{x}
\end{aligned}
$$

- Gradient ascent can be applied to simultaneously train all of these weight vectors $\mathbf{w}_k$

# Logistic Regression for K > 2 (continued)

- The conditional probability for class k ≠ K can be computed as

$$P(y = k|\mathbf{x}) \;=\; \frac{\exp(\mathbf{w}_k \cdot \mathbf{x})}{1 + \sum_{\ell=1}^{K-1} \exp(\mathbf{w}_\ell \cdot \mathbf{x})}$$

- For class K, the conditional probability is

$$P(y = K|\mathbf{x}) \;=\; \frac{1}{1 + \sum_{\ell=1}^{K-1} \exp(\mathbf{w}_\ell \cdot \mathbf{x})}$$

# Summary of Logistic Regression

- Learns conditional probability distribution P(*y* | **x**)
- Local Search
  - begins with initial weight vector.  Modifies it iteratively to maximize the log likelihood of the data
- Eager
  - the classifier is constructed from the training examples, which can then be discarded
- Online or Batch
  - both online and batch variants of the algorithm exist