Probabilistic Linear Classifier: Logistic Regression

Three Main Approaches to learning a Classifier

- Learn a classifier: a function f, $\hat{y} = f(\mathbf{x})$
- Learn a probabilistic ***discriminative*** model, i.e., the conditional distribution $P(y \mid \mathbf{x})$
- Learn a probabilistic ***generative*** model, i.e., the joint probability distribution: $P(\mathbf{x}, y)$
- Examples:
  - Learn a classifier: Perceptron, LDA (projection with threshold view)
  - Learn a conditional distribution: ***Logistic regression***
  - Learn the joint distribution: a probabilistic view of Linear Discriminant Analysis (LDA)

Notation Shift

- $S=\{(\mathbf{x}^i, y^i): i=1,\ldots, N\}$ --- superscript for example index. N is the total number of examples
- Subscript for element index within a vector, i.e., x^i_j represents the jth element of the ith training example
- Class labels are 0 and 1 (not +1 and -1)

Logistic Regression

- Given training set D, logistic regression learns the conditional distribution P(*y* | **x**)
- We will assume only two classes *y* = 0 and *y* = 1 and a parametric form for P(y = 1 | x, w) were w is the parameter vector

$$p(y=1|\mathbf{x};\mathbf{w}) = p_1(\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{x}}}$$

$$p(y=0|\mathbf{x};\mathbf{w}) = 1 - p_1(\mathbf{x})$$

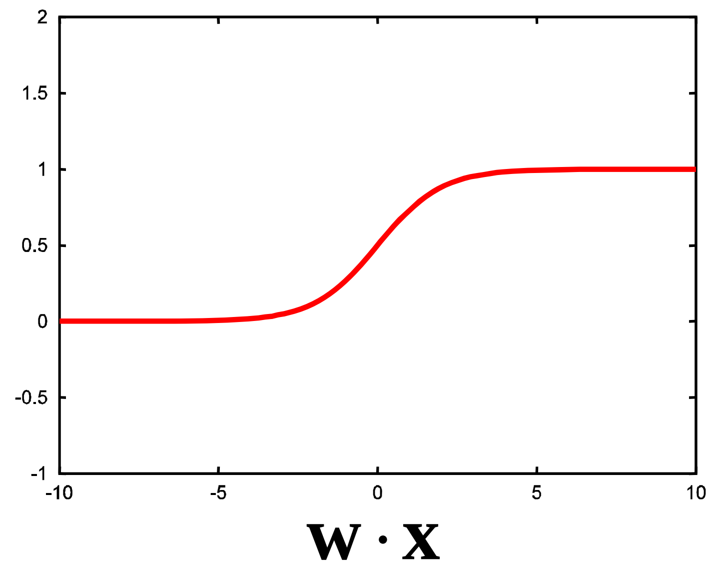
- It is easy to show that this is equivalent to

$$\log\frac{p(y=1|\mathbf{x};\mathbf{w})}{p(y=0|\mathbf{x};\mathbf{w})} = \mathbf{w}\cdot\mathbf{x}$$

- i.e. the ***log odds*** of class 1 is a linear function of **x**.

Why the Logistic (Sigmoid) Function

$$g(\mathbf{x},\mathbf{w}) = \frac{1}{1+\exp(-\mathbf{w}\cdot\mathbf{x})}$$



A linear function has a range from $[-\infty,\infty]$, the logistic function transforms the range to [0,1] to be a probability.

Logistic Regression Yields Linear Classifier

- Recall that given P(y | x) we predict ŷ = 1 if the expected loss of predicting 0 is greater than predicting 1
(for now assume L(0,1) = L(1,0))

$$E_{y|x}[L(0,y)] > E_{y|x}[L(1,y)] \Leftrightarrow$$

$$\sum_y P(y \mid \mathbf{x}) L(0,y) > \sum_y P(y \mid \mathbf{x}) L(1,y) \Leftrightarrow$$

$$P(y=0 \mid \mathbf{x}) L_{00} + P(y=1 \mid \mathbf{x}) L_{01} > P(y=0 \mid \mathbf{x}) L_{10} + P(y=1 \mid \mathbf{x}) L_{11} \Leftrightarrow$$

$$P(y=1 \mid \mathbf{x}) > P(y=0 \mid \mathbf{x}) \Leftrightarrow$$

$$\frac{P(y=1 \mid \mathbf{x})}{P(y=0 \mid \mathbf{x})} > 1 \Leftrightarrow \log \frac{P(y=1 \mid \mathbf{x})}{P(y=0 \mid \mathbf{x})} > 0 \Leftrightarrow$$

$$\mathbf{w} \cdot \mathbf{x} > 0$$

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

Maximum Likelihood Learning

- Recall that the likelihood function is the probability of the data **D** given the parameters – p(**D**|**w**)
- It is a function of the parameters
- Maximum likelihood learning finds the parameters that maximize this likelihood function
- A common trick is to work with log-likelihood, i.e., take the logarithm of the likelihood function – log p(**D**|**w**)

Computing the Likelihood

- In our framework, we assume each training example ($\mathbf{x}^i$, y^i) is drawn independently from the same (but unknown) distribution $P(\mathbf{x}, y)$ (the famous **i.i.d** assumption), hence we can write

$$\log P(D \mid \mathbf{w}) = \log \prod_i P(\mathbf{x}^i, y^i \mid \mathbf{w}) = \sum_i \log P(\mathbf{x}^i, y^i \mid \mathbf{w})$$

- Joint distribution P(a,b) can be factored as P(a | b)P(b)

$$\arg\max_{w} \log P(D \mid \mathbf{w}) = \arg\max_{\mathbf{w}} \sum_i \log P(\mathbf{x}^i, y^i \mid \mathbf{w})$$
$$= \arg\max_{\mathbf{w}} \sum_i \log P(y^i \mid \mathbf{x}^i, \mathbf{w}) P(\mathbf{x}^i \mid \mathbf{w})$$

- Further, P(**x** | **w**) = P(**x**) because it does not depend on w, so:

$$\arg\max_{\mathbf{w}} \log P(D \mid \mathbf{w}) = \arg\max_{\mathbf{w}} \sum_i \log P(y^i \mid \mathbf{x}^i, \mathbf{w})$$

Computing the Likelihood

$$\arg\max_{\mathbf{w}} \log P(D \mid \mathbf{w}) = \arg\max_{\mathbf{w}} \sum_i \log P(y^i \mid \mathbf{x}^i, \mathbf{w})$$

Recall

$$p(y=1 \mid \mathbf{x}, \mathbf{w}) = g(\mathbf{x}, \mathbf{w}) = \frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{x}}} = \hat{y}$$

$$p(y=0 \mid \mathbf{x}, \mathbf{w}) = 1 - g(\mathbf{x}, \mathbf{w}) = 1 - \hat{y}$$

So: $P(y^i \mid \mathbf{x}^i, \mathbf{w}) = \begin{cases} \hat{y} & \text{if } y^i = 1 \\ 1-\hat{y} & \text{otherwsie} \end{cases}$

This can be compactly written as

$$p(y^i \mid \mathbf{x}^i, \mathbf{w}) = \hat{y}^{y^i}(1-\hat{y})^{(1-y^i)}$$

We will take our learning objective function to be:

$$\begin{aligned} L(\mathbf{w}) = \log P(D \mid \mathbf{w}) &= \sum_i \log P(y^i \mid \mathbf{x}^i, \mathbf{w}) \\ &= \sum_i [y^i \log \hat{y}^i + (1-y^i)\log(1-\hat{y}^i)] \end{aligned}$$

Fitting Logistic Regression by Gradient Ascent

$$L(\mathbf{w}) = \sum_i \log P(y^i \mid \mathbf{x}^i, \mathbf{w}) = \sum_i [y^i \log \hat{y}^i + (1-y^i)\log(1-\hat{y}^i)]$$

$$\frac{\partial L(\mathbf{w})}{\partial w_j} = \frac{\partial \log P(y^i \mid \mathbf{x}^i, \mathbf{w})}{\partial w_j} = \frac{\partial}{\partial w_j}[y^i \log \hat{y}^i + (1-y^i)\log(1-\hat{y}^i)]$$

$$= \frac{y^i}{\hat{y}^i}\left(\frac{\partial \hat{y}^i}{\partial w_j}\right) + \frac{1-y^i}{1-\hat{y}^i}\left(-\frac{\partial \hat{y}^i}{\partial w_j}\right) = \left[\frac{y^i}{\hat{y}^i} - \frac{1-y^i}{1-\hat{y}^i}\right]\frac{\partial \hat{y}^i}{\partial w_j}$$

$$= \left[\frac{y^i - y^i\hat{y}^i - \hat{y}^i + y^i\hat{y}^i}{\hat{y}^i(1-\hat{y}^i)}\right]\frac{\partial \hat{y}^i}{\partial w_j} = \left[\frac{y^i - \hat{y}^i}{\hat{y}^i(1-\hat{y}^i)}\right]\frac{\partial \hat{y}^i}{\partial w_j}$$

Recall that $\hat{y}^i = \hat{y}(\mathbf{x}^i, \mathbf{w}) = \dfrac{1}{1+\exp(-\mathbf{w}\cdot\mathbf{x}^i)}$

for $g(t) = \dfrac{1}{1+\exp(-t)}$ we have

$$g'(t) = \frac{\exp(-t)}{(1+\exp(-t))^2} = g(t)(1-g(t))$$

So $\dfrac{\partial \hat{y}^i}{\partial w_j} = \hat{y}^i(1-\hat{y}^i)\dfrac{\partial(\mathbf{w}\cdot\mathbf{x}^i)}{\partial w_j} = \hat{y}^i(1-\hat{y}^i)x_j^i$

$$\frac{\partial L(\mathbf{w})}{\partial w_j} = \sum_{i=1}^{N}(y^i - \hat{y}^i)x_j^i \qquad \nabla L(\mathbf{w}) = \sum_{i=1}^{N}(y^i - \hat{y}^i)\mathbf{x}^i$$

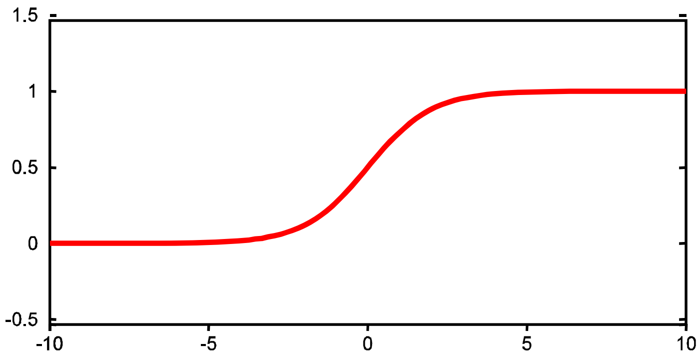
Batch Gradient Ascent for LR

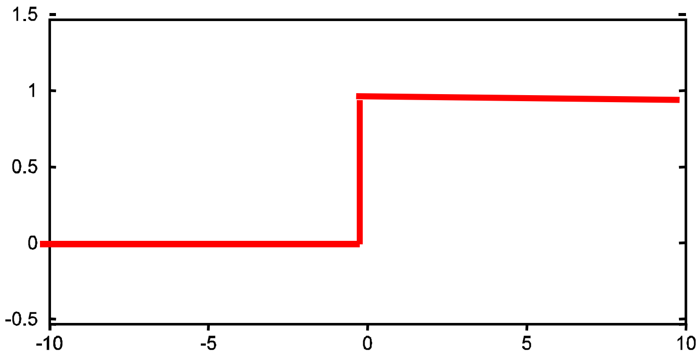
$$
\begin{aligned}
&\text{Given: training examples } (\mathbf{x}^i\ , y^i),\ i = 1,\ldots,N \\
&\text{Let } \mathbf{w} \leftarrow (0,0,0,\ldots,0) \\
&\text{Repeat until convergence} \\
&\qquad \mathbf{d} \leftarrow (0,0,0,\ldots,0) \\
&\qquad \text{For } i\ =\ 1 \text{ to } N \text{ do} \\
&\qquad\qquad \widehat{y}^i \leftarrow \frac{1}{1+e^{-\mathbf{w}\,\cdot\,\mathbf{x}^i}} \\
&\qquad\qquad error = y^i - \widehat{y}^i \\
&\qquad\qquad \mathbf{d} = \mathbf{d} + error \cdot \mathbf{x}^i \\
&\qquad \mathbf{w} \leftarrow \mathbf{w} + \eta\,\mathbf{d}
\end{aligned}
$$

- Online gradient *ascent* algorithm can be easily constructed

Connection Between
Logistic Regression & Perceptron Algorithm

If we replace the logistic function with a step function:





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

$$h_{\mathbf{w}}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w}\cdot\mathbf{x} > 0 \\ 0 & \text{otherwise} \end{cases}$$

Both algorithms uses the same updating rule :

$$\mathbf{w} = \mathbf{w} + \eta(y^i - h_{\mathbf{w}}(\mathbf{x}^i))\mathbf{x}^i$$

Multi-Class Cases

- 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:

$$\log \frac{P(y=1 \mid \mathbf{x})}{P(y=K \mid \mathbf{x})} = \mathbf{w}_1 \cdot \mathbf{x}$$

$$\log \frac{P(y=2 \mid \mathbf{x})}{P(y=K \mid \mathbf{x})} = \mathbf{w}_2 \cdot \mathbf{x}$$

$$\vdots$$

$$\log \frac{P(y=K-1 \mid \mathbf{x})}{P(y=K \mid \mathbf{x})} = \mathbf{w}_{K-1} \cdot \mathbf{x}$$

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

Multi-Class Cases

- Conditional probability for class k $\neq$ K can be computed as

$$P(y = k \mid \mathbf{x}) = \frac{\exp(\mathbf{w}_k \cdot \mathbf{x})}{1 + \sum_{l=1}^{K-1} \exp(\mathbf{w}_l \cdot \mathbf{x})}$$

- For class K, the conditional probability is

$$P(y = K \mid \mathbf{x}) = \frac{1}{1 + \sum_{l=1}^{K-1} \exp(\mathbf{w}_l \cdot \mathbf{x})}$$

Summary of Logistic Regression

- Learns conditional probability distribution $P(y \mid \mathbf{x})$
- Local Search
  - begins with initial weight vector. Modifies it iteratively to maximize the log likelihood of the data
- Online or Batch
  - both online and batch variants of the algorithm exist