Outline

Logistic Discrimination

- In logistic discrimination (or logistic regression or logit regression), we do not model the class-conditional densities $p(\mathbf{x} \mid C_i)$ but rather their ratio.
- Unlike the parametric classification approach (the likelihood-based approach studied before) which learns the classifier by estimating the parameters of $p(\mathbf{x} \mid C_i)$, logistic discrimination, a discriminant-based approach, estimates the parameters of the discriminant directly.

Two Classes

- Let us start with two classes and assume that the log likelihood ratio is linear:

$$\log \frac{p(\mathbf{x} \mid C_1)}{p(\mathbf{x} \mid C_2)} = \mathbf{w}^T \mathbf{x} + w_0^0$$

- Using Bayes' rule, we have

$$\begin{aligned}\operatorname{logit}(P(C_1 \mid \mathbf{x})) &= \log \frac{P(C_1 \mid \mathbf{x})}{1 - P(C_1 \mid \mathbf{x})} = \log \frac{P(C_1 \mid \mathbf{x})}{P(C_2 \mid \mathbf{x})} = \log \frac{p(\mathbf{x} \mid C_1)}{p(\mathbf{x} \mid C_2)} + \log \frac{P(C_1)}{P(C_2)} \\ &= \mathbf{w}^T \mathbf{x} + w_0\end{aligned}$$

  where $w_0 = w_0^0 + \log \frac{P(C_1)}{P(C_2)}$
- Then we have

$$y = \hat{P}(C_1 \mid \mathbf{x}) = \operatorname{sigmoid}(\mathbf{w}^T \mathbf{x} + w_0) = \frac{1}{1 + \exp[-(\mathbf{w}^T \mathbf{x} + w_0)]}$$

  - equivalent to the case when class-conditional densities are normal
  - but logistic discrimination is more general, e.g., $\mathbf{x}$ may take discrete attributes

Parameter Learning – I

- Training set $\mathcal{X} = \{(\mathbf{x}^t, r^t)\}_{t=1}^N$ where

$$r^t = \begin{cases} 1 & \text{if } \mathbf{x}^t \in C_1 \\ 0 & \text{if } \mathbf{x}^t \in C_2 \end{cases}$$

- Given an input $\mathbf{x}$, we assume that r is Bernoulli with parameter $y = P(C_1 \mid \mathbf{x})$:

$$r \mid \mathbf{x} \sim \text{Ber}(r; y)$$

- Different from the likelihood-based methods where we modeled $p(\mathbf{x} \mid C_i)$, in the discriminant-based approach, we model directly $r \mid \mathbf{x}$, i.e., $P(C_1 \mid \mathbf{x})$.
- Likelihood:

$$L(\mathbf{w}, w_0 \mid \mathcal{X}) = \prod_{t=1}^{N} (y^t)^{r^t} (1 - y^t)^{1-r^t}$$

Parameter Learning – II

- Maximizing the likelihood function $L(\mathbf{w}, w_0 \mid \mathcal{X})$ is equivalent to minimizing an error function (corresponding to the cross-entropy) $E(\mathbf{w}, w_0 \mid \mathcal{X})$ defined as

$$
\begin{aligned}
E(\mathbf{w}, w_0 \mid \mathcal{X}) &= -\log L(\mathbf{w}, w_0 \mid \mathcal{X}) \\
&= -\sum_{t=1}^{N} \left[r^t \log y^t + (1 - r^t) \log(1 - y^t) \right]
\end{aligned}
$$

- Due to the nonlinearity of the sigmoid function, we cannot solve it directly. Iterative algorithms such as gradient descent or other methods can be used.
- Please keep in mind once a suitable model and an error function is defined, the (numerical) optimization of the model parameters to minimize the error function can be done by using one of many possible techniques.

Gradient Descent Learning – I

- If $y = \text{sigmoid}(a) = \frac{1}{1+\exp(-a)}$, its derivative is

$$\frac{dy}{da} = y(1-y)$$

- Update equations for w_j ($j = 0, \ldots, d$):

$$\Delta w_j = -\eta \frac{\partial E}{\partial w_j} = \eta \sum_{t=1}^{N} \left(\frac{r^t}{y^t} \frac{\partial y^t}{\partial w_j} - \frac{1-r^t}{1-y^t} \frac{\partial y^t}{\partial w_j} \right)$$

$$= \eta \sum_{t=1}^{N} \left(\frac{r^t}{y^t} - \frac{1-r^t}{1-y^t} \right) \frac{\partial y^t}{\partial a^t} \frac{\partial a^t}{\partial w_j}$$

Gradient Descent Learning – II

- Update equations for w_0 and w_j ($j = 1, \ldots, d$):
  Since $a = \mathbf{w}^T\mathbf{x} + w_0$, we have

$$\frac{\partial a}{\partial w_0} = 1, \qquad \frac{\partial a}{\partial w_j} = x_j, \quad j = 1, \ldots, d$$

  So

$$\Delta w_0 = -\eta \frac{\partial E}{\partial w_0} = \eta \sum_{t=1}^{N} (r^t - y^t)$$

$$\Delta w_j = -\eta \frac{\partial E}{\partial w_j} = \eta \sum_{t=1}^{N} \left(\frac{r^t}{y^t} - \frac{1 - r^t}{1 - y^t} \right) y^t(1 - y^t) x_j^t$$

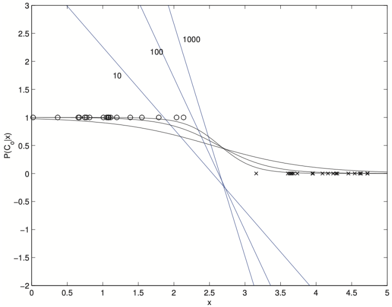
$$= \eta \sum_{t=1}^{N} (r^t - y^t) x_j^t, \quad j = 1, \ldots, d$$

Gradient Descent Algorithm

For $j = 0, \ldots, d$
    $w_j \leftarrow$rand(-0.01,0.01)
Repeat
    For $j = 0, \ldots, d$
        $\Delta w_j \leftarrow 0$
    For $t = 1, \ldots, N$
        $o \leftarrow 0$
        For $j = 0, \ldots, d$
            $o \leftarrow o + w_j x_j^t$
        $y \leftarrow \mathrm{sigmoid}(o)$
        $\Delta w_j \leftarrow \Delta w_j + (r^t - y) x_j^t$
    For $j = 0, \ldots, d$
        $w_j \leftarrow w_j + \eta \Delta w_j$
Until convergence

For w_0, we assume that there is an extra input x_0, which is always 1: $x_0^t = 1, \forall t$.

A Univariate Two-Class Example



Both $wx + w_0$ *and sigmoid*$(wx + w_0)$ *are shown as the learning develops.*

- We see that to get outputs of 0 and 1, the sigmoid hardens, which is achieved by increasing the magnitude of w, or $\|\mathbf{w}\|$ in the multivariate case.
- After training, during testing, given $\mathbf{x}^t$, we calculate $y^t = \text{sigmoid}(\mathbf{w}^T\mathbf{x}^t + w_0)$ and choose C_1 if $y^t > 0.5$ and choose C_2 otherwise.

Remarks on Parameter Learning

- To minimize # of misclassifications, we do not need to continue learning until all y^t are 0 or 1, but only until y^t are less than or greater than 0.5 (i.e., on the correct side of the decision boundary).
- If we do continue training beyond this point, cross-entropy will continue decreasing ($|w_j|$ will continue increasing to harden the sigmoid), but the number of misclassifications will not decrease.
- We continue training until the number of misclassifications does not decrease (which will be 0 if the classes are linearly separable).
- Actually stopping early before we have 0 training error is a form of regularization.
  - Because we start with weights almost 0 and they move away as training continues, stopping early corresponds to a model with more weights close to 0 and effectively fewer parameters.

Multiple Classes

- One of the $K > 2$ classes, e.g., C_K, is taken as the reference class.
- Assume that

$$\log \frac{p(\mathbf{x} \mid C_i)}{p(\mathbf{x} \mid C_K)} = \mathbf{w}_i^T \mathbf{x} + w_{i0}^0, \quad i = 1, \ldots, K-1$$

So we have

$$\begin{aligned} \frac{P(C_i \mid \mathbf{x})}{P(C_K \mid \mathbf{x})} &= \frac{p(\mathbf{x} \mid C_i) P(C_i)}{p(\mathbf{x} \mid C_K) P(C_K)} \\ &= \exp(\mathbf{w}_i^T \mathbf{x} + w_{i0}^0) \cdot \exp\left(\log \frac{P(C_i)}{P(C_K)}\right) \\ &= \exp(\mathbf{w}_i^T \mathbf{x} + w_{i0}) \end{aligned} \tag{1}$$

where $w_{i0} = w_{i0}^0 + \log \frac{P(C_i)}{P(C_K)}$

Generalization of Sigmoid Function

- Summing (1) over $i = 1, \ldots, K-1$:

$$\sum_{i=1}^{K-1} \frac{P(C_i \mid \mathbf{x})}{P(C_K \mid \mathbf{x})} = \sum_{i=1}^{K-1} \exp(\mathbf{w}_i^T \mathbf{x} + w_{i0}) = \frac{1 - P(C_K \mid \mathbf{x})}{P(C_K \mid \mathbf{x})}$$

  we get

$$P(C_K \mid \mathbf{x}) = \frac{1}{1 + \sum_{i=1}^{K-1} \exp(\mathbf{w}_i^T \mathbf{x} + w_{i0})} \tag{2}$$

- From (1) and (2), we get

$$P(C_i \mid \mathbf{x}) = \frac{\exp(\mathbf{w}_i^T \mathbf{x} + w_{i0})}{1 + \sum_{j=1}^{K-1} \exp(\mathbf{w}_j^T \mathbf{x} + w_{j0})}, \quad i = 1, \ldots, K-1$$

Softmax Function

- If we want to treat all classes uniformly without having to choose a reference class, we can use the following expression instead for the posterior class probabilities:

$$y_i = \hat{P}(C_i \mid \mathbf{x}) = \frac{\exp(\mathbf{w}_i^T \mathbf{x} + w_{i0})}{\sum_{j=1}^K \exp(\mathbf{w}_j^T \mathbf{x} + w_{j0})}, \quad i = 1, \ldots, K$$

  where we call

$$y_i = \text{softmax}_i(\mathbf{a}) = \frac{\exp(a_i)}{\sum_{j=1}^K \exp(a_j)}$$

  the softmax function.
  - If $\mathbf{w}_i^T \mathbf{x} + w_{i0}$ for one class is sufficiently larger than for the others, its corresponding y_i will be close to 1 and the others will be close to 0.
- The softmax function behaves like taking a maximum, but it has the advantage of being differentiable.
- Softmax functions also guarantee that $\sum_{i=1}^K y_i = 1$.

Parameter Learning

- Each sample point is a generalized Bernoulli or multinomial trial with one draw, i.e.

$$\mathbf{r} \mid \mathbf{x} \sim \mathrm{Mult}(\mathbf{r}; 1, \mathbf{y})$$

  where $y_i \equiv P(C_i \mid \mathbf{x})$
- Likelihood:

$$L(\{\mathbf{w}_i, w_{i0}\}_i \mid \mathcal{X}) = \prod_t \prod_i (y_i^t)^{r_i^t}$$

- Cross-entropy error function:

$$E(\{\mathbf{w}_i, w_{i0}\}_i \mid \mathcal{X}) = -\sum_t \sum_i r_i^t \log y_i^t$$

Gradient Descent Learning

- If $y_i = \exp(a_i)/\sum_j \exp(a_j)$, its derivative is

$$\frac{\partial y_i}{\partial a_j} = y_i(\delta_{ij} - y_j)$$

  where δ_{ij} is the Kronecker delta, which is 1 if $i = j$ and 0 if $i \neq j$.

- Update equations given $\sum_i r_i^t = 1$:

$$\begin{aligned}
\Delta \mathbf{w}_j &= \eta \sum_t \sum_i r_i^t(\delta_{ij} - y_j^t)\mathbf{x}^t = \eta \sum_t \Big[\sum_i r_i^t \delta_{ij} - y_j^t \sum_i r_i^t\Big]\mathbf{x}^t \\
&= \eta \sum_t (r_j^t - y_j^t)\mathbf{x}^t, \quad j = 1, \ldots, K \\
\Delta w_{j0} &= \eta \sum_t (r^t - y^t)
\end{aligned}$$

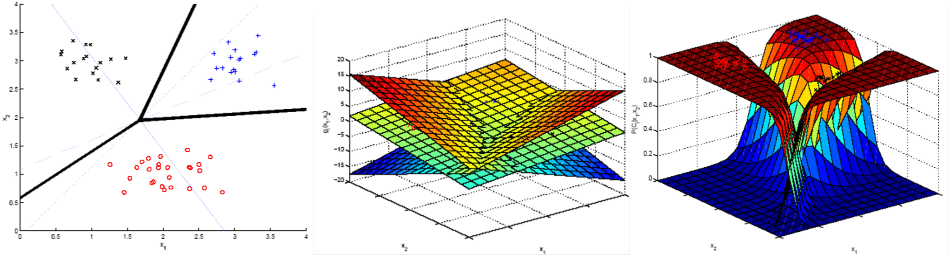
  Note that because of the normalization in softmax, $\mathbf{w}_j$ and w_{j0} are affected not only by $\mathbf{x}^t \in C_j$ but also by $\mathbf{x}^t \in C_i$, $i \neq j$.

Gradient Descent Algorithm

```
For $i = 1, \ldots, K$, For $j = 0, \ldots, d$, $w_{ij} \leftarrow$ rand$(-0.01, 0.01)$
Repeat
    For $i = 1, \ldots, K$, For $j = 0, \ldots, d$, $\Delta w_{ij} \leftarrow 0$
    For $t = 1, \ldots, N$
        For $i = 1, \ldots, K$
            $o_i \leftarrow 0$
            For $j = 0, \ldots, d$
                $o_i \leftarrow o_i + w_{ij} x_j^t$
        For $i = 1, \ldots, K$
            $y_i \leftarrow \exp(o_i) / \sum_k \exp(o_k)$
        For $i = 1, \ldots, K$
            For $j = 0, \ldots, d$
                $\Delta w_{ij} \leftarrow \Delta w_{ij} + (r_i^t - y_i) x_j^t$
    For $i = 1, \ldots, K$
        For $j = 0, \ldots, d$
            $w_{ij} \leftarrow w_{ij} + \eta \Delta w_{ij}$
Until convergence
```

We take $x_0^t = 1, \forall t$.

A Two-dimensional Three-class Example



Remarks on Parameter Learning

- We do not need to continue training to minimize cross-entropy as much as possible; we train only until the correct class has the highest weighted sum, and therefore we can stop training earlier by checking the number of misclassifications.

Logistic Discriminant

- When data are normally distributed, the logistic discriminant has a comparable performance to the parametric, normal-based linear discriminant.
- Logistic discrimination can still be used when the class-conditional densities are non-normal or when they are not unimodal, as long as classes are linearly separable.

Generalizing the Linear Model

- The ratio of class-conditional densities is of course not restricted to be linear.
- Quadratic discriminant:

$$\log \frac{p(\mathbf{x} \mid C_i)}{p(\mathbf{x} \mid C_K)} = \mathbf{x}^T \mathbf{W}_i \mathbf{x} + \mathbf{w}_i^T \mathbf{x} + w_{i0}$$

  which corresponds to parametric discrimination with multivariate normal class-conditionals having different covariance matrices.
- Sum of nonlinear basis functions:

$$\log \frac{p(\mathbf{x} \mid C_i)}{p(\mathbf{x} \mid C_K)} = \mathbf{w}_i^T \boldsymbol{\Phi}(\mathbf{x}) + w_{i0}$$

  where $\boldsymbol{\Phi}(\cdot)$ are basis functions which transform the original input variables to a new set of variables.
- Basis functions are related to:
  - Hidden units like sigmoid function in neural networks (studied later)
  - Kernels in kernel methods such as support vector machines (SVM) (studied later).