Logistic Regression

Dit-Yan Yeung

Department of Computer Science and Engineering
Hong Kong University of Science and Technology

COMP 4211: Machine Learning (Fall 2022)

Linear Regression vs. Logistic Regression

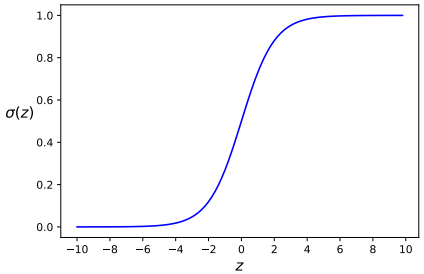
- While linear regression solves regression problems, logistic regression solves classification problems.
- Logistic regression transforms the output of a linear regression model using the logistic function (for binary classification) or the softmax function (for multiclass classification).
- Binary classification:
  - Two classes
  - Training set $\mathcal{S} = \left\{(\mathbf{x}^{(\ell)}, y^{(\ell)})\right\}_{\ell=1}^{N}$ where $y^{(\ell)} \in \{0, 1\}$
- Multiclass classification:
  - Three or more classes
  - Training set $\mathcal{S} = \left\{(\mathbf{x}^{(\ell)}, \mathbf{y}^{(\ell)})\right\}_{\ell=1}^{N}$ where $\mathbf{y}^{(\ell)} \in \{0, 1\}^K$ uses one-hot encoding such that all dimensions of $\mathbf{y}^{(\ell)}$ are equal to 0 except $y_j^{(\ell)} = 1$ if $\mathbf{x}^{(\ell)}$ belongs to class j (denoted by C_j)

Logistic Function

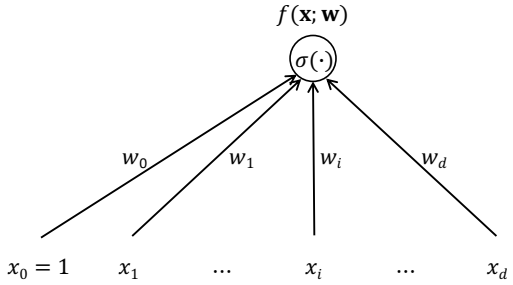
- Logistic function (a.k.a. sigmoid function):

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

- For any real-valued z, $\sigma(z) \in (0, 1)$.



Network Model



Class Probabilities

- Logistic regression represents the probability that $\mathbf{x}$ belongs to C_1 as:

$$f(\mathbf{x}; \mathbf{w}) = \sigma(\mathbf{w}^\top \tilde{\mathbf{x}}),$$

  where $\mathbf{w} = (w_0, \ldots, w_d)^\top$ and $\tilde{\mathbf{x}} = (x_0 = 1, x_1, \ldots, x_d)^\top$.
- There exists a probabilistic interpretation of the model, but here we only present a less formal treatment which is intuitively easier to understand.
- A good model corresponds to one with $\mathbf{w}$ such that $f(\mathbf{x}^{(\ell)}; \mathbf{w})$ is as close to 1 as possible if $\mathbf{x}^{(\ell)} \in \mathcal{S}$ belongs to C_1 (i.e., $y^{(\ell)} = 1$), and it is as close to 0 as possible if $\mathbf{x}^{(\ell)} \in \mathcal{S}$ belongs to C_0 (i.e., $y^{(\ell)} = 0$).

Likelihood and Cross-Entropy Loss

- Assuming that the examples in $\mathcal{S}$ are independent, finding a good model corresponds to maximizing the following quantity w.r.t. the model parameters $\mathbf{w}$:

$$\prod_{\ell=1}^{N} f(\mathbf{x}^{(\ell)}; \mathbf{w})^{y^{(\ell)}} (1 - f(\mathbf{x}^{(\ell)}; \mathbf{w}))^{1-y^{(\ell)}},$$

  which is called the likelihood.
- Since the logarithmic function is an increasing function, it is equivalent to maximizing the logarithm of the likelihood, called the log likelihood, or minimizing the negative logarithm of the likelihood, called the cross-entropy loss or log loss:

$$L(\mathbf{w}; \mathcal{S}) = -\sum_{\ell=1}^{N} \left[y^{(\ell)} \log f(\mathbf{x}^{(\ell)}; \mathbf{w}) + (1 - y^{(\ell)}) \log(1 - f(\mathbf{x}^{(\ell)}; \mathbf{w})) \right].$$

Minimization of Cross-Entropy Loss

- Although the cross-entropy loss function $L(\mathbf{w}; \mathcal{S})$ is convex, no closed-form solution exists.
- Iterative algorithms are needed for finding the model parameters $\mathbf{w}$ that minimize $L(\mathbf{w}; \mathcal{S})$.
- A simple gradient-based iterative algorithm is called gradient descent, which iteratively updates the parameter vector $\mathbf{w}$ in a direction opposite to the gradient of the loss function at $\mathbf{w}$.

A Preliminary Result: Derivative of Sigmoid Function

- Sigmoid function:

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

- Derivative of sigmoid function:

$$\begin{aligned}
\sigma'(z) &= \frac{e^{-z}}{(1 + e^{-z})^2} \\
&= \frac{1}{1 + e^{-z}} \left(1 - \frac{1}{1 + e^{-z}}\right) \\
&= \sigma(z)\left(1 - \sigma(z)\right),
\end{aligned}$$

allowing $\sigma'(z)$ to be computed entirely based on $\sigma(z)$.

Gradient Descent Learning

- Recall that

$$f(\mathbf{x}; \mathbf{w}) = \sigma(\mathbf{w}^\top \tilde{\mathbf{x}}).$$

- It follows that

$$\begin{aligned}\frac{\partial f}{\partial w_i} &= \sigma'(\mathbf{w}^\top \tilde{\mathbf{x}})\, x_i \\ &= \sigma(\mathbf{w}^\top \tilde{\mathbf{x}})\big(1 - \sigma(\mathbf{w}^\top \tilde{\mathbf{x}})\big)\, x_i \\ &= f(\mathbf{x}; \mathbf{w})\,(1 - f(\mathbf{x}; \mathbf{w}))\, x_i,\end{aligned}$$

for w_i $(i = 0, \ldots, d)$.

Gradient Descent Learning (2)

- Recall that the cross-entropy loss is

$$L(\mathbf{w}; \mathcal{S}) = -\sum_{\ell=1}^{N} \left[y^{(\ell)} \log f(\mathbf{x}^{(\ell)}; \mathbf{w}) + (1 - y^{(\ell)}) \log(1 - f(\mathbf{x}^{(\ell)}; \mathbf{w})) \right].$$

- Gradient of cross-entropy loss function:

$$\begin{aligned}
\frac{\partial L}{\partial w_i} &= -\sum_{\ell=1}^{N} \left(\frac{y^{(\ell)}}{f(\mathbf{x}^{(\ell)}; \mathbf{w})} \frac{\partial f}{\partial w_i} - \frac{1 - y^{(\ell)}}{1 - f(\mathbf{x}^{(\ell)}; \mathbf{w})} \frac{\partial f}{\partial w_i} \right) \\
&= -\sum_{\ell=1}^{N} \frac{y^{(\ell)} - f(\mathbf{x}^{(\ell)}; \mathbf{w})}{f(\mathbf{x}^{(\ell)}; \mathbf{w}) \left(1 - f(\mathbf{x}^{(\ell)}; \mathbf{w})\right)} \frac{\partial f}{\partial w_i} \\
&= -\sum_{\ell=1}^{N} \left(y^{(\ell)} - f(\mathbf{x}^{(\ell)}; \mathbf{w}) \right) x_i^{(\ell)}.
\end{aligned}$$

Gradient Descent Learning (3)

- Weight update for (batch) gradient descent:

$$\Delta w_i = -\eta \frac{\partial L}{\partial w_i} = \eta \sum_{\ell=1}^{N} \left(y^{(\ell)} - f(\mathbf{x}^{(\ell)}; \mathbf{w}) \right) x_i^{(\ell)}, \qquad i = 0, \ldots, d,$$

  where η is the learning rate.

- Vector form of weight update equation:

$$\Delta \mathbf{w} = \eta \sum_{\ell=1}^{N} \left(y^{(\ell)} - f(\mathbf{x}^{(\ell)}; \mathbf{w}) \right) \tilde{\mathbf{x}}^{(\ell)}.$$

  In other words, the weight change is proportional to the product of the input and the difference between the actual output and the predicted output summed over all the examples in the training set.

Algorithm for Gradient Descent Learning

for $i = 0$ to d **do**
  $w_i \leftarrow \text{rand}(-0.01, 0.01)$
end for
repeat
  for $i = 0$ to d **do**
    $\Delta w_i \leftarrow 0$
    for $\ell = 1$ to N **do**
      $s \leftarrow 0$
      for $i' = 0$ to d **do**
        $s \leftarrow s + w_{i'} \, x_{i'}^{(\ell)}$
      end for
      $a \leftarrow \text{sigmoid}(s)$
      $\Delta w_i \leftarrow \Delta w_i + \eta \left(y^{(\ell)} - a \right) x_i^{(\ell)}$
    end for
  end for
  for $i = 0$ to d **do**
    $w_i \leftarrow w_i + \Delta w_i$
  end for
until convergence

Softmax Function

- Let $\mathbf{z} = (z_1, \ldots, z_K)^\top \in \mathbb{R}^K$ be a K-dimensional vector.
- Softmax function:

$$\mathbf{r} = (r_1, \ldots, r_K)^\top = \text{softmax}(\mathbf{z}),$$

where

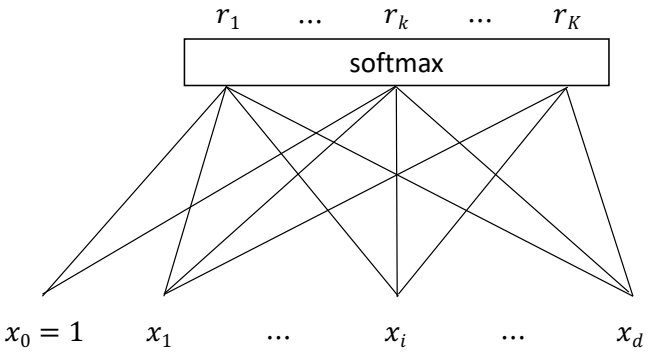
$$r_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \in (0, 1).$$

- Note that $\sum_{j=1}^{K} r_j = 1$ and (with the derivation skipped)

$$\frac{\partial r_j}{\partial z_k} = r_j \left(\delta_{jk} - r_k\right),$$

where δ_{jk} is the Kronecker delta such that $\delta_{jk} = 1$ if $j = k$ and 0 otherwise.
- The softmax function behaves like taking the maximum, but it has the advantage of being differentiable.

Network Model



Class Probabilities

- Let $\mathbf{W} = [w_{ki}] \in \mathbb{R}^{K \times (1+d)}$ denote the parameters of the model such that the k-th row of $\mathbf{W}$, denoted by $\mathbf{w}_k$, corresponds to the output for class C_k.
- For multiclass classification, logistic regression represents the probability distribution over K classes as:

$$f(\mathbf{x}; \mathbf{W}) = \text{softmax}(\mathbf{W}\,\tilde{\mathbf{x}}) = \mathbf{r} \in (0,1)^K.$$

- Intuitively, a good model corresponds to one with $\mathbf{W}$ such that $r_k^{(\ell)}$ is as close to 1 as possible if $\mathbf{x}^{(\ell)} \in \mathcal{S}$ belongs to C_k (i.e., $y_k^{(\ell)} = 1$).

Likelihood and Cross-Entropy Loss

- Likelihood:

$$\prod_{\ell=1}^{N} \prod_{j=1}^{K} (r_j^{(\ell)})^{y_j^{(\ell)}},$$

  assuming that the examples in $\mathcal{S}$ are independent.

- Cross-entropy loss:

$$L(\mathbf{W}; \mathcal{S}) = -\sum_{\ell=1}^{N} \sum_{j=1}^{K} y_j^{(\ell)} \log r_j^{(\ell)}.$$

- Gradient descent can be used as an iterative algorithm for minimizing the cross-entropy loss function $L(\mathbf{W}; \mathcal{S})$ w.r.t. $\mathbf{W}$.

Gradient Descent Learning

- Gradient of cross-entropy loss function:

$$\begin{aligned}
\frac{\partial L}{\partial w_{ki}} &= -\sum_{\ell=1}^{N}\sum_{j=1}^{K}\frac{y_j^{(\ell)}}{r_j^{(\ell)}}\frac{\partial r_j^{(\ell)}}{\partial w_{ki}} \\
&= -\sum_{\ell=1}^{N}\sum_{j=1}^{K}\frac{y_j^{(\ell)}}{r_j^{(\ell)}} r_j^{(\ell)}\left(\delta_{jk} - r_k^{(\ell)}\right) x_i^{(\ell)} \\
&= -\sum_{\ell=1}^{N}\sum_{j=1}^{K} y_j^{(\ell)}\left(\delta_{jk} - r_k^{(\ell)}\right) x_i^{(\ell)} \\
&= -\sum_{\ell=1}^{N}\left(\sum_{j=1}^{K} y_j^{(\ell)}\delta_{jk} - r_k^{(\ell)}\sum_{j=1}^{K} y_j^{(\ell)}\right) x_i^{(\ell)} = -\sum_{\ell=1}^{N}\left(y_k^{(\ell)} - r_k^{(\ell)}\right) x_i^{(\ell)}.
\end{aligned}$$

Gradient Descent Learning (2)

- Weight update for (batch) gradient descent:

$$\Delta w_{ki} = -\eta \frac{\partial L}{\partial w_{ki}} = \eta \sum_{\ell=1}^{N} \left(y_k^{(\ell)} - r_k^{(\ell)} \right) x_i^{(\ell)}, \;\; i = 0, \ldots, d, \;\; k = 1, \ldots, K.$$

- Vector form of weight update equations:

$$\Delta \mathbf{w}_k^\top = \eta \sum_{\ell=1}^{N} \left(y_k^{(\ell)} - r_k^{(\ell)} \right) \tilde{\mathbf{x}}^{(\ell)}, \qquad k = 1, \ldots, K.$$

Regularization

- Like linear regression, regularizers can be added to the cross-entropy loss function of logistic regression to prevent overfitting.
- L_2 regularization is still the most common choice:

$$\|\mathbf{W}\|_F^2,$$

  where $\|\cdot\|_F$ denotes the Frobenius norm which is a matrix norm.
- As in linear regression, not regularizing the bias terms w_{k0} may give better result.

Confusion Matrix and Accuracy

- Confusion matrix:

		Predicted condition	
		Positive	Negative
Actual condition	Positive	True positive (TP)	False negative (FN)
	Negative	False positive (FP)	True negative (TN)

- Accuracy:

$$\frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}.$$

Precision, Recall, and F1 Score

- Precision:
$$P = \frac{\text{TP}}{\text{TP} + \text{FP}}.$$
- Recall:
$$R = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$
- F1 score:
$$F1 = 2 \cdot \frac{P \cdot R}{P + R},$$
which is the harmonic mean of precision and recall.
- If the classes are very imbalanced, it is better to report the precision, recall, and F1 score rather than accuracy.