CSE 6363 - Machine Learning

Logistic Regression

Alex Dillhoff

University of Texas at Arlington

Classification

With linear regression, we fit a model to the data.

This allowed us to make predictions about the observations paired with the input features.

Classification

In the regression example, both the inputs and outputs were continuous values.

We now turn to the classification task: **we want to classify some input vector as being a part of 1 of K distinct classes.**

Classification

In the binary case, the target variable takes on either a 0 or 1.

For $K > 2$, we use a K-dimensional vector that has as 1 corresponding to the class.

Classification

Given the classes

- car
- truck
- person

A target vector for person is $\hat{\mathbf{y}} = [0, 0, 1]^T$

Linear Models (again)

Again, we start with a linear model $y = f(\mathbf{x}; \mathbf{w})$.

The output should be some discrete value:

- 0 and 1
- -1 and +1
- 1, 2, 3, … ???

Linear Models (again)

The logistic model is often approached by introducing the **odds** of an event occurring:

$$\frac{p}{1-p},$$

where p is the probability of the event happening.

Linear Models (again)

Our input p represents the probability in range $(0, 1)$ which we want to map to the real number space.

To approximate this, we apply the natural logarithm to the odds.

Linear Models (again)

The logistic model assumes a linear relationship between the linear model $\mathbf{w}^T\mathbf{x}$ and the logit function

$$\text{logit}(p) = \ln \frac{p}{1-p}.$$

This function maps a value in range $(0, 1)$ to the space of real numbers.

Linear Models (again)

Under this assumption, we can write

$$\text{logit}(p) = \mathbf{w}^T\mathbf{x}.$$

This assumption is reasonable because we ultimately want to predict the **probability** that an event occurs.

The output should then be in the range of $(0, 1)$.

Linear Models (again)

If the logit function produces output in the range of real numbers, as does our linear model $\mathbf{w}^T\mathbf{x}$, then we ultimately want a function that maps **from** the range of real numbers to **to** $(0, 1)$.

We can achieve this using the **inverse** of the logit function.

Linear Models (again)

The model should produce some likelihood of whether or not the sample belongs to class 1 or 2.

This is commonly accomplished with the **logistic sigmoid function**.

$$\sigma(z) = \frac{1}{1 + \exp(-z)},$$

where $z = \mathbf{w}^T\mathbf{x}$.

Logistic Sigmoid Function

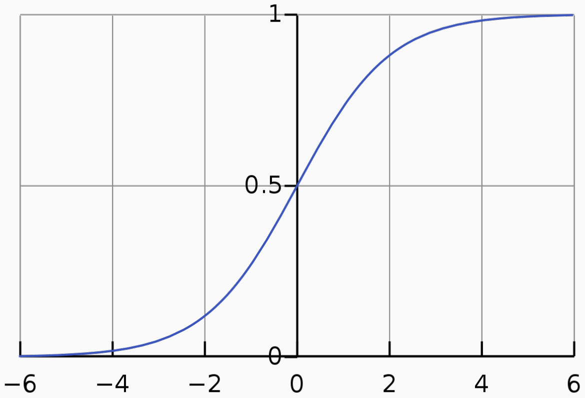

Figure 1: Plot of the logistic sigmoid function.

Logistic Sigmoid Function

The logistic sigmoid function also has a convenient derivative, which is useful when solving for the model parameters via gradient descent.

$$\frac{d}{dx} = \sigma(x)(1 - \sigma(x))$$

Linear Models (again)

Applying this function to the raw output of our model yields the form

$$f(\mathbf{x}; \mathbf{w}) = h(\mathbf{w}^T \mathbf{x}),$$

where h is our choice of activation function.

Binary Classification

We begin with binary classification.

Come up with the parameters of a line that separate the data.

We will assume linearly separable data.

Binary Classification

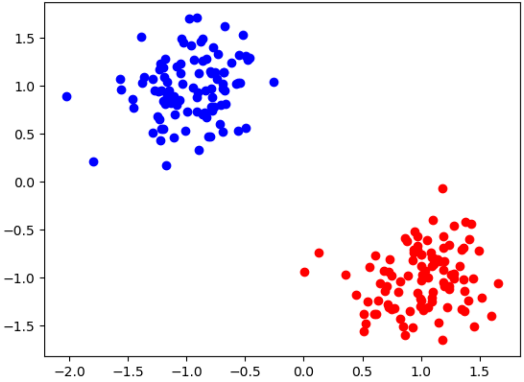

Figure 2: Two groups of data separated into the red and blue class.

Binary Classification

This toy example is chosen to focus on the core concepts.

We could easily come up with parameters that draw a line between them.

One example is the line $y = x$.

Binary Classification

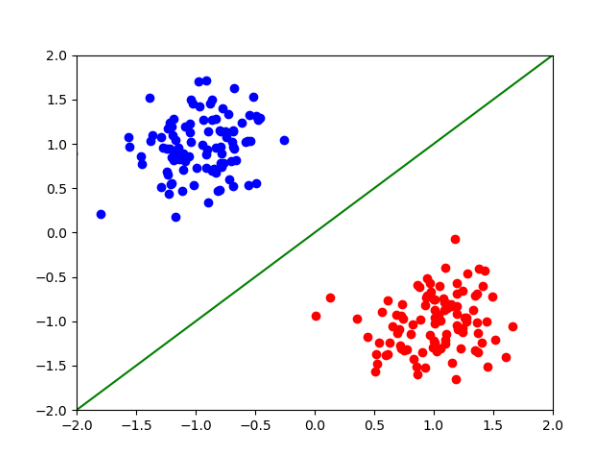

Figure 3: The line $y = x$ separates the data perfectly.

Binary Classification

The parameter vector $\mathbf{w}$ is orthogonal to the decision boundary.

The model output is 0 when $\mathbf{x}$ lies on the line.

Binary Classification

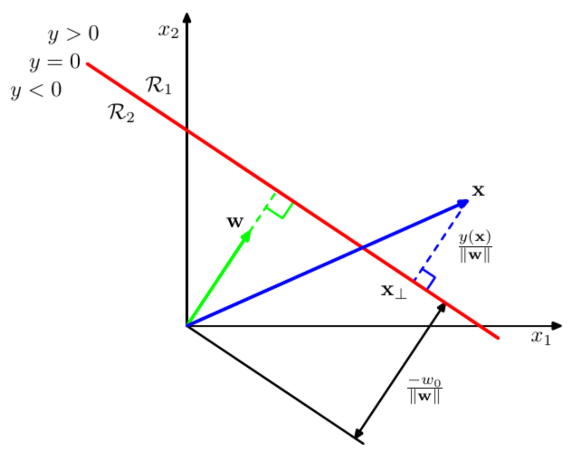

Figure 4: Geometry of the decision boundary in 2D. Source: Bishop

Binary Classification

In the binary case, we are approximating $p(C_1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x})$.

Then $p(C_2|\mathbf{x}) = 1 - p(C_1|\mathbf{x})$.

Measuring Performance

Qualitatively, we can see that our dataset is perfectly classified.

How can we measure this quantitatively?

Measuring Performance

How can we measure this quantitatively?

A common choice for binary classification is to use $L1$ loss:

$$L_1 = \sum_i |\hat{y}_i - y_i|,$$

where $\hat{y}_i$ is the ground truth and y_i is the model output for input $\mathbf{x}_i$.

Maximum Likelihood

To fit our model to the data, we can take a **maximum likelihood** approach.

This will reveal some very useful functions when dealing with any classification problem.

Maximum Likelihood

Let $y_i \in \{0, 1\}$ be the target for binary classification and $\hat{y}_i \in (0, 1)$ be the output of a logistic regression model.

The likelihood function is

$$p(\mathbf{y}|\mathbf{w}) = \prod_{i=1}^{n} \hat{y}_i^{y_i}(1 - \hat{y}_i)^{1-y_i}.$$

Maximum Likelihood

Since the output is restricted within the range $(0, 1)$, the model will never produce 0 or 1.

If the target $y_i = 0$, then we can evaluate the subexpression $1 - \hat{y}_i$. In this case, the likelihood increases as $\hat{y}_i$ decreases.

If the target $y_i = 1$, then we evaluate the subexpression $\hat{y}_i$.

Maximum Likelihood

When fitting this model, we want to define an error measure based on the above function.

This is done by taking the negative logarithm of $p(\mathbf{y}|\mathbf{w})$.

$$E(\mathbf{w}) = -\ln p(\mathbf{y}|\mathbf{w}) = -\sum_{i=1}^{n} y_i \ln \hat{y}_i + (1 - y_i) \ln(1 - \hat{y}_i)$$

Maximum Likelihood

$$E(\mathbf{w}) = -\ln p(\mathbf{y}|\mathbf{w}) = -\sum_{i=1}^{n} y_i \ln \hat{y}_i + (1 - y_i) \ln(1 - \hat{y}_i)$$

This function is commonly referred to as the **cross-entropy** function.

Maximum Likelihood

If we use this as an objective function for gradient descent with the understanding that $\hat{y}_i = \sigma(\mathbf{w}^T\mathbf{x})$, then the gradient of the error function is

$$\nabla E(\mathbf{w}) = \sum_{i=1}^{n}(\hat{y}_i - y_i)\mathbf{x}_i.$$

Multiple Classes

In multiclass logistic regression, we are dealing with target values that can take on one of K values $y \in \{1, 2, \ldots, K\}$.

If our goal is to model the distribution over K classes, a multinomial distribution is the obvious choice.

Multiple Classes

Let $p(y|\mathbf{x};\theta)$ be a distribution over K numbers $w_1, \ldots, w_K$ that sum to 1.

Our parameterized model cannot be represented exactly by a multinomial distribution, so we will derive it so that it satisfies the same constraints.

Multiple Classes

We can start by introducing K parameter vectors $\mathbf{w}_1, \ldots, \mathbf{w}_K \in \mathbb{R}^d$, where d is the number of input features.

Then each vector $\mathbf{w}_k^T\mathbf{x}$ represents $p(C_k|\mathbf{x};\mathbf{w}_k)$.

Multiple Classes

We need to *squash* each $\mathbf{w}_k^T\mathbf{x}$ so that the output sums to 1.

This is accomplished via the **softmax function**

Multiple Classes

$$p(C_k|\mathbf{x}) = \frac{\exp(\mathbf{w}_k^T\mathbf{x})}{\sum_j \exp(\mathbf{w}_j^T\mathbf{x})}.$$

Multiple Classes

For K classes, the output vector looks like

$$\hat{\mathbf{y}} = \begin{bmatrix} \frac{\exp(\mathbf{w}_1^T\mathbf{x})}{\sum_j \exp(\mathbf{w}_j^T\mathbf{x})} \\ \frac{\exp(\mathbf{w}_2^T\mathbf{x})}{\sum_j \exp(\mathbf{w}_j^T\mathbf{x})} \\ \vdots \\ \frac{\exp(\mathbf{w}_K^T\mathbf{x})}{\sum_j \exp(\mathbf{w}_j^T\mathbf{x})} \end{bmatrix}$$

Multiple Classes

The target vector for each sample is $\mathbf{y}_i \in \mathbb{R}^k$. Likewise, the output vector $\hat{\mathbf{y}}_i$ also has k elements.

Multiple Classes

The maximum likelihood function for the multiclass setting is given by

$$p(\mathsf{Y}|\mathsf{W}) = \prod_{i=1}^{n}\prod_{k=1}^{K} p(C_k|\mathbf{x}_i)^{y_{ik}} = \prod_{i=1}^{n}\prod_{k=1}^{K} \hat{y}_{ik}^{y_{ik}}.$$

Multiple Classes

As with the binary case, we can take the negative logarithm of this function to produce an error function.

$E(\mathrm{W}) = -\ln p(\mathrm{Y}|\mathrm{W}) = -\sum_{i=1}^{n} \sum_{k=1}^{K} y_{ik} \ln \hat{y}_{ik}$

This is the **cross-entropy** function for multiclass classification.

Multiple Classes

The gradient of this function is given as

$$\nabla_{\mathbf{w}_j} E(\mathbf{W}) = \sum_{i=1}^{n} (\hat{y}_{ij} - y_{ij}) \mathbf{x}_i.$$

Summary

- Logistic regression is a linear model for classification parameterized by $\mathbf{w}$.
- It is a probabilistic model that uses the sigmoid function to produce a *probability*.

$$\hat{y} = \sigma(\mathbf{w}^T\mathbf{x})$$

- The maximum likelihood function for logistic regression is given by

$$E(\mathbf{w}) = -\ln p(\mathbf{y}|\mathbf{w}) = -\sum_{i=1}^{n} y_i \ln \hat{y}_i + (1 - y_i)\ln(1 - \hat{y}_i)$$

Summary

- The gradient of the maximum likelihood function is given by

$$\nabla E(\mathbf{w}) = \sum_{i=1}^{n} (\hat{y}_i - y_i)\mathbf{x}_i.$$

- The final update rule for gradient descent is given by

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha E(\mathbf{w}).$$

Summary

- Multiclass logistic regression uses the softmax function to produce a probability distribution over K classes.
- The maximum likelihood function for multiclass logistic regression is given by

$$E(\mathbf{W}) = -\ln p(\mathbf{Y}|\mathbf{W}) = -\sum_{i=1}^{n}\sum_{k=1}^{K} y_{ik} \ln \hat{y}_{ik}$$

- The gradient of the maximum likelihood function is given by

$$\nabla_{\mathbf{w}_j} E(\mathbf{W}) = \sum^{n} (\hat{y}_{ij} - y_{ij})\mathbf{x}_i.$$

Summary

- The final update rule for gradient descent is given by

$$\mathbf{w}_j \leftarrow \mathbf{w}_j - \alpha \nabla_{\mathbf{w}_j} E(\mathbf{W}).$$

- The update rule is the same for both binary and multiclass logistic regression.