Logistic Regression

Fraida Fund

Contents

In this lecture

- Linear classifiers
- Logistic regression
- Fitting logistic regression

Classification

- Suppose we have a series of data points $\{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x_n}, y_n)\}$ and there is some (unknown) relationship between $\mathbf{x_i}$ and y_i.
- **Classification**: The output variable y is constrained to be $\in 1, 2, \cdots, K$
- **Binary classification**: The output variable y is constrained to be $\in 0, 1$

Linear classifiers

Binary classification with linear decision boundary

- Plot training data points
- Draw a line (**decision boundary**) separating 0 class and 1 class
- If a new data point is in the **decision region** corresponding to class 0, then $\hat{y} = 0$.
- If it is in the decision region corresponding to class 1, then $\hat{y} = 1$.

Binary classification with linear decision boundary: illustration

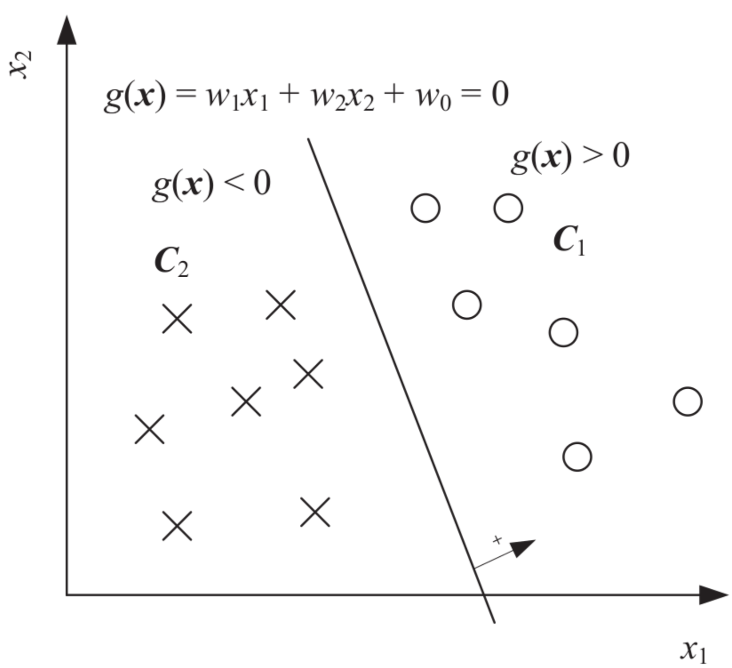

Figure 1: Binary classification problem with linear decision boundary.

Linear classification rule

- Given a **weight vector**: $\mathbf{w} = (w_0, \cdots, w_d)$

- Compute linear combination $z = w_0 + \sum_{j=1}^{d} w_d x_d$
- Predict class:

$$\hat{y} = \begin{cases} 1, z > 0 \\ 0, z \le 0 \end{cases}$$

Multi-class classification with linear decision boundary: illustration

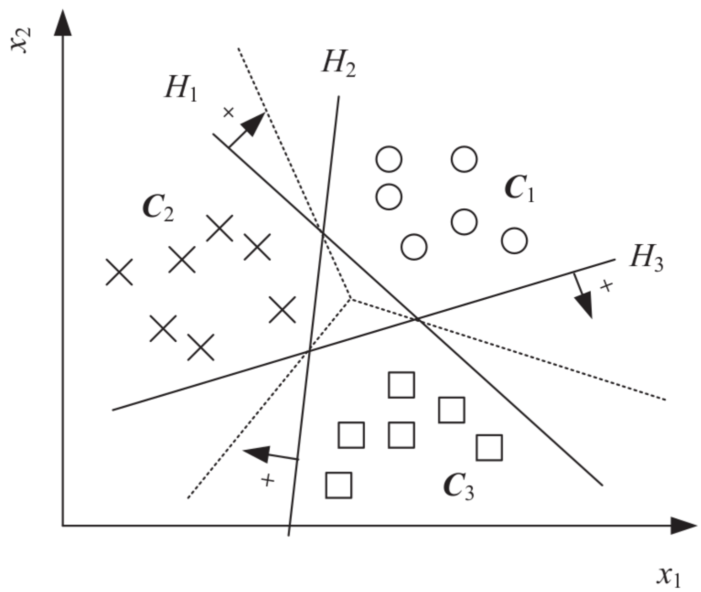

Figure 2: Each hyperplane H_i separates the examples of C_i from the examples of all other classes.

Linear separability

Given training data

$$(\mathbf{x}_i, y_i), i = 1, \cdots, N$$

The problem is **perfectly linearly separable** if there exists a **separating hyperplane** H_i such that all $\mathbf{x} \in C_i$ lie on its positive side, and all $\mathbf{x} \in C_j, j \neq i$ lie on its negative side.

Non-uniqueness of separating hyperplane

When a separating hyperplane exists, it is not unique

Non-existence of perfectly separating hyperplane

Many datasets *not* linearly separable - some points will be misclassified by separating hyperplane.

Choosing a hyperplane

We will try to find the hyperplane that minimizes loss according to some **loss function**.

Will revisit several times this semester.

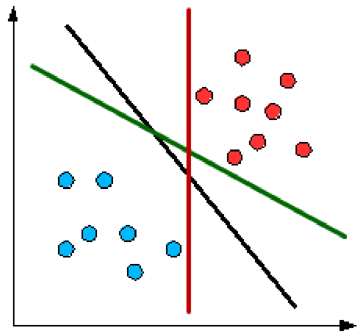

Figure 3: Several separating hyperplanes.

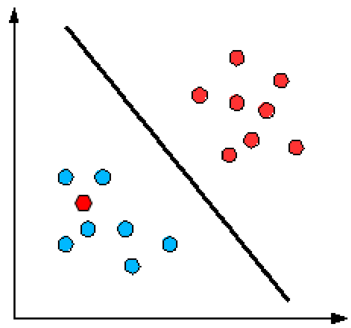

Figure 4: Not separable.

Logistic regression

Probabilistic model for binary classification with linear decision boundary

- Binary classification problem: $y = 0, 1$
- Linear classification: $z = w_0 + \sum_{j=1}^{k} w_j x_j$
- Rather than predicting y directly, let us predict a probability:

$$P(y = i|\mathbf{x}) = f(z)?$$

Logistic regression model as log of odds ratio

$$\ln \frac{p}{1-p} = w_0 + \sum_{j=1}^{k} w_k x_k$$

$$p = \frac{e^{w_0 + w_1 x_1 + \dots}}{1 + e^{w_0 + w_1 x_1 + \dots}} = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + \dots)}}$$

Logistic function

- $\sigma(z) = \frac{1}{1+e^{-z}}$ is a classic "S"-shaped function
- logistic (also called sigmoidal) function

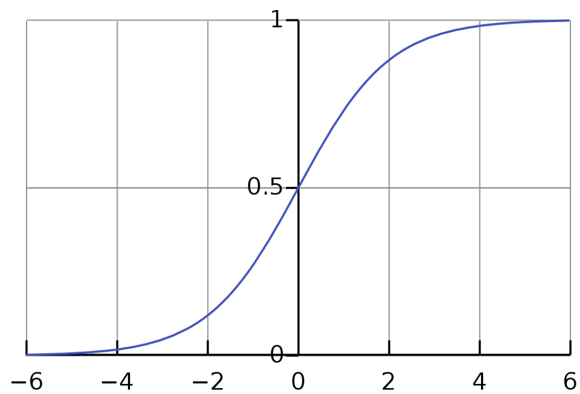

Figure 5: Shape of sigmoid function.

- takes a real value and maps it to range $[0, 1]$.

For classification: $\hat{y} = \sigma(\mathbf{w}^{\mathbf{T}}\mathbf{x} + w_0)$

Logistic function for binary classification

$$P(y = 1|\mathbf{x}) = \frac{1}{1 + e^{-z}}, \quad P(y = 0|\mathbf{x}) = \frac{e^{-z}}{1 + e^{-z}}$$

(note: $P(y = 1) + P(y = 0) = 1$)

Logistic function with threshold

Choose a threshold t, then

$$\hat{y} = \begin{cases} 1, & P(y = 1|\mathbf{x}) > t \\ 0, & P(y = 1|\mathbf{x}) \leq t \end{cases}$$

Logistic model as a "soft" classifier

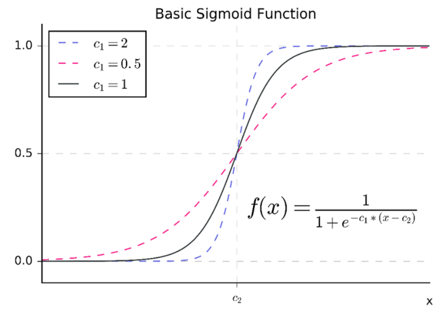

Figure 6: Plot of $P(y=1|x)=\frac{1}{1+e^{-z}}, z=w_1x$. As $w_1 \rightarrow \infty$ the logistic model becomes a "hard" rule.

Logistic classifier properties (1)

- Class probabilities depend on distance from separating hyperplane
- Points far from separating hyperplane have probability ≈ 0 or ≈ 1
- When $||\mathbf{w}||$ is larger, class probabilities go towards extremes (0,1) more quickly

Logistic classifier properties (2)

- Unlike linear regression, weights do *not* correspond to change in output associated with one-unit change in input
- Sign of weight *does* tell us about relationship between a given feature and target variable

Logistic regression - illustration

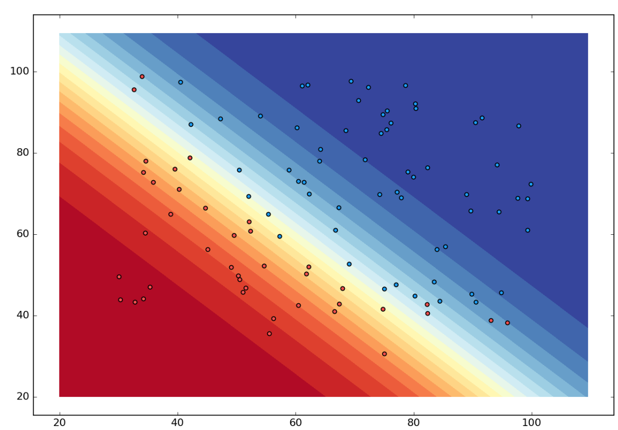

Figure 7: Logistic regression, illustrated with contour plot.

Multi-class linear regression

Suppose $y \in 1, \dots, K$. We can formulate the multi-class logistic regression using:

- $\mathbf{W} \in R^{K \times d}, \mathbf{w}_0 \in R^k$ (slope matrix and bias vector)
- $\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{w}_0$ (K linear functions)

Softmax function

$$g_k(\mathbf{z}) = \frac{e^{z_k}}{\sum_{\ell=1}^{K} e^{z_\ell}}$$

- Takes as input a vector of K numbers
- Outputs K probabilities proportional to the exponentials of the input numbers.

Softmax function as a PMF

Acts like a probability mass function:

- $g_k(\mathbf{z}) \in [0, 1]$ for each k
- $\sum_{k=1}^{K} g_k(\mathbf{z}) = 1$
- larger input corresponds to larger "probability"

Softmax function for multi-class logistic regression (1)

Class probabilities are given by

$$P(y = k|\mathbf{x}) = \frac{e^{z_k}}{\sum_{\ell=1}^{K} e^{z_\ell}}$$

Softmax function for multi-class logistic regression (2)

When $z_k \gg z_\ell$ for all $\ell \neq k$:

- $g_k(\mathbf{z}) \approx 1$
- $g_\ell(\mathbf{z}) \approx 0$ for all $\ell \neq k$

Assign highest probability to class k when z_k is largest.

Fitting logistic regression model

Learning logistic model parameters

Let $\mathbf{W} \in R^{K \times p}$ be a weight matrix including bias term $(p = d + 1)$.

Linear weights are unknown **model parameters**:

$$\mathbf{z} = \mathbf{W}\mathbf{x}, \mathbf{W} \in R^{K \times p}$$

$$P(y = k|\mathbf{x}) = g_k(\mathbf{z}) = g_k(\mathbf{W}\mathbf{x})$$

Given training data $(\mathbf{x}_i, y_i), i = 1, \dots, N$, we must learn $\mathbf{W}$.

Maximum likelihood estimation (1)

Let $P(\mathbf{y}|\mathbf{X}, \mathbf{W})$ be the probability of class labels $\mathbf{y} = (y_1, \dots, y_N)^T$ given inputs $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)^T$ and weights $\mathbf{W}$.

The **maximum likelihood estimate**

$$\hat{\mathbf{W}} = \underset{W}{\text{argmax}}\, P(\mathbf{y}|\mathbf{X}, \mathbf{W})$$

is the estimate of parameters for which these observations are most likely.

Maximum likelihood estimation (2)

Assume outputs y_i are independent of one another,

$$P(\mathbf{y}|\mathbf{X}, \mathbf{W}) = \prod_{i=1}^{N} P(y_i|\mathbf{x_i}, \mathbf{W})$$

Maximum likelihood estimation (3)

Define the **negative log likelihood**:

$$L(\mathbf{W}) = -\ln P(\mathbf{y}|\mathbf{X}, \mathbf{W})$$

$$= -\sum_{i=1}^{N} \ln P(y_i|\mathbf{x_i}, \mathbf{W})$$

(also called cross-entropy)

Maximum likelihood estimation (4)

Then we can re-write max likelihood estimator using a loss function to minimize:

$$\hat{\mathbf{W}} = \underset{W}{\text{argmax}}\, P(\mathbf{y}|\mathbf{X}, \mathbf{W}) = \underset{W}{\text{argmin}}\, L(\mathbf{W})$$

Binary cross-entropy loss (1)

For binary classification with class labels $0, 1$:

$$
\begin{aligned}
\ln P(y_i|\mathbf{x_i}, \mathbf{w}) & \\
&= y_i \ln P(y_i = 1|\mathbf{x_i}, \mathbf{w}) + (1 - y_i) \ln P(y_i = 0|\mathbf{x_i}, \mathbf{w}) \\
&= y_i \ln \sigma(z_i) + (1 - y_i) \ln(1 - \sigma(z_i)) \\
&= y_i(\ln \sigma(z_i) - \ln \sigma(-z_i)) + \ln \sigma(-z_i) \\
&= y_i \ln \frac{\sigma(z_i)}{\sigma(-z_i)} + \ln \sigma(-z_i) \\
&= y_i \ln \frac{1 + e^{z_i}}{1 + e^{-z_i}} + \ln \sigma(-z_i) \\
&= y_i \ln \frac{e^{z_i}(e^{-z_i} + 1)}{1 + e^{-z_i}} + \ln \sigma(-z_i) \\
&= y_i z_i - \ln(1 + e^{z_i})
\end{aligned}
\tag{1}
$$

(Note: $\sigma(-z) = 1 - \sigma(z)$)

Binary cross-entropy loss (2)

Binary cross-entropy loss function (negative log likelihood):

$$\sum_{i=1}^{N} \ln(1 + e^{z_i}) - y_i z_i$$

Cross-entropy loss for multi-class classification (1)

Define “one-hot” vector - for a sample from class k, all entries in the vector are 0 except for the kth entry which is 1:

$$r_{ik} = \begin{cases} 1 & y_i = k \\ 0 & y_i \neq k \end{cases}$$

$$i = 1, \ldots, N, \quad k = 1, \ldots, K$$

Cross-entropy loss for multi-class classification (2)

Then,

$$\ln P(y_i|\mathbf{x_i}, \mathbf{W}) = \sum_{k=1}^{K} r_{ik} \ln P(y_i = k|\mathbf{x_i}, \mathbf{W})$$

Cross-entropy loss function is

$$\sum_{i=1}^{N} \left[\ln \left(\sum_k e^{z_{ik}} \right) - \sum_k z_{ik} r_{ik} \right]$$

Minimizing cross-entropy loss

To minimize, we would take the partial derivative:

$$\frac{\partial L(W)}{\partial W_{kj}} = 0$$

for all W_{kj}

But, there is no closed-form expression - can only estimate weights via numerical optimization.

"Recipe" for logistic regression

- Choose a **model**: get $P(y = k|\mathbf{x})$ using sigmoid or softmax.
- Get **data** - for supervised learning, we need **labeled** examples: $(x_i, y_i), i = 1, 2, \cdots, N$
- Choose a **loss function** that will measure how well model fits data: cross-entropy loss
- Find model **parameters** that minimize loss: use numerical optimization to find weights
- Use model to **predict** $\hat{y}$ for new, unlabeled samples