Multi-Class Logistic Regression and Perceptron

Instructor: Wei Xu

Some slides adapted from Dan Jurfasky, Brendan O'Connor and Marine Carpuat

MultiClass Classification

- Q: what if we have more than 2 categories?
  - Sentiment: Positive, Negative, Neutral
  - Document topics: Sports, Politics, Business, Entertainment, …

Q: How to easily do Multi-label classification?

Two Types of MultiClass Classification

- Multi-label Classification
  - each instance can be assigned more than one labels

- Multinominal Classification
  - each instance appears in exactly one class (classes are exclusive)

Multinominal Classification

- Pretty straightforward with Naive Bayes.

$$P(\mathrm{spam}|D) \propto P(\mathrm{spam}) \prod_{w \in D} P(w|\mathrm{spam})$$

Log-Linear Models

$$P(y|x) \propto e^{w \cdot f(x,y)}$$

$$P(y|x) = \frac{1}{Z(w)} e^{w \cdot f(x,y)}$$

Multinominal Logistic Regression

$$P(y|x) \propto e^{w \cdot f(x,y)}$$

$$P(y|x) = \frac{1}{Z(w)} e^{w \cdot f(x,y)}$$

$$\boxed{P(y|x) = \frac{e^{w \cdot f(x,y)}}{\sum_{y' \in Y} e^{w \cdot f(x,y')}}}$$

normalization term (Z) so that probabilities sum to 1

(a.k.a) Softmax Regression

Not logged in Talk Contributions Create account Log in

Article Talk Read Edit View history Search Wikipedia

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Softmax function

From Wikipedia, the free encyclopedia

In mathematics, the **softmax function**, or **normalized exponential function**,[1]:198 is a generalization of the logistic function that "squashes" a K-dimensional vector $\mathbf{z}$ of arbitrary real values to a K-dimensional vector $\sigma(\mathbf{z})$ of real values in the range (0, 1) that add up to 1. The function is given by

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

Q: what if there are only 2 categories?

$$P(y = j|x_i) = \frac{e^{w_j \cdot x_i}}{\sum_k e^{w_k \cdot x_i}}$$

Q: what if there are only 2 categories?

$$P(y=1|x) = \frac{e^{w_1 \cdot x}}{e^{w_0 \cdot x + w_1 \cdot x - w_1 \cdot x} + e^{w_1 \cdot x}}$$

Q: what if there are only 2 categories?

$$P(y=1|x) = \frac{e^{w_1 \cdot x}}{e^{w_0 \cdot x - w_1 \cdot x} e^{w_1 \cdot x} + e^{w_1 \cdot x}}$$

Q: what if there are only 2 categories?

$$P(y=1|x) = \frac{e^{w_1 \cdot x}}{e^{w_1 \cdot x}(e^{w_0 \cdot x - w_1 \cdot x} + 1)}$$

Q: what if there are only 2 categories?

$$P(y = 1|x) = \frac{1}{e^{w_0 \cdot x - w_1 \cdot x} + 1}$$

Q: what if there are only 2 categories?

$$P(y = 1|x) = \frac{1}{e^{-w' \cdot x} + 1}$$

Sigmoid (logistic) function

Multinominal Logistic Regression

- Binary (two classes):
  - We have one feature vector that matches the size of the vocabulary
- Multi-class in practice:
  - one weight vector for each category

$$w_{\text{pos}} \qquad w_{\text{neg}} \qquad w_{\text{neut}}$$

In practice, can represent this with one giant weight vector and repeated features for each category.

Maximum Likelihood Estimation

$$w_{\mathrm{MLE}} = \mathrm{argmax}_w \log P(y_1, \ldots, y_n | x_1, \ldots, x_n; w)$$

$$= \mathrm{argmax}_w \sum_i \log P(y_i | x_i; w)$$

$$= \mathrm{argmax}_w \sum_i \log \frac{e^{w \cdot f(x_i, y_i)}}{\sum_{y' \in Y} e^{w \cdot f(x_i, y')}}$$

Multiclass LR Gradient

$$\frac{\partial \mathcal{L}}{\partial w_j} = \sum_{i=1}^{D} f_j(y_i, d_i) - \sum_{i=1}^{D} \sum_{y \in Y} f_j(y, d_i) P(y|d_i)$$

empirical feature count

expected feature count

(a.k.a) Maximum Entropy Classifier

- or MaxEnt

- Math proof of “LR=MaxEnt”:
  - [Klein and Manning 2003]
  - [Mount 2011]

http://www.win-vector.com/dfiles/LogisticRegressionMaxEnt.pdf

Perceptron Algorithm

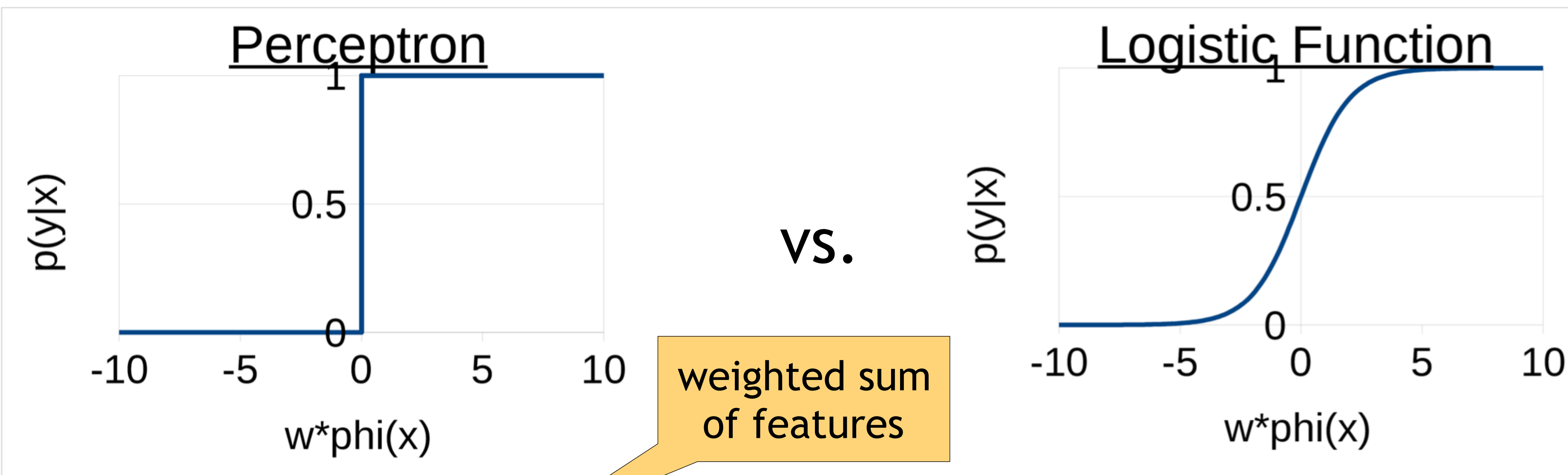
- Very similar to logistic regression
- Not exactly computing gradient

[Rosenblatt 1957]

http://www.peterasaro.org/writing/neural_networks.html

Perceptron Algorithm

- Very similar to logistic regression
- Not exactly computing gradient (simpler)



$$P(y=1|x)=1 \text{ if } w\cdot\varphi(x)\geq 0$$
$$P(y=1|x)=0 \text{ if } w\cdot\varphi(x)<0$$

$$P(y=1|x)=\frac{e^{w\cdot\varphi(x)}}{1+e^{w\cdot\varphi(x)}}$$

Perceptron vs. LR

- The Perceptron is an online learning algorithm.
- Standard Logistic Regression is not

Online Learning

- The Perceptron is an online learning algorithm.
- Logistic Regression is not:

this update is effectively the same as “w += y_i * x_i”

$$
\begin{aligned}
w_{\text{MLE}} &= \operatorname{argmax}_w \log P(y_1, \ldots, y_d | x_1, \ldots, x_d; w) \\
&= \operatorname{argmax}_w \sum_i y_i \log p_i + (1 - y_i) \log(1 - p_i)
\end{aligned}
$$

(Full) Batch Learning

- update parameters after each pass of training set

```
Initialize weight vector w = 0
Create features
Loop for K iterations
   Loop for all training examples x_i, y_i
        …
   update_weights(w)
```

Online Learning

- update parameters for each training example

```
Initialize weight vector w = 0
Create features
Loop for K iterations
    Loop for all training examples x_i, y_i
        …
        update_weights(w, x_i, y_i)
```

Perceptron Algorithm

- Very similar to logistic regression
- Not exactly computing gradient

features of a training example x

$$w \leftarrow w + y\,\varphi(x)$$

weights

label

If y = 1, increase the weights for features in $\varphi(x)$

If y = -1, decrease the weights for features in $\varphi(x)$

Perceptron Algorithm

- Very similar to logistic regression
- Not exactly computing gradient

```
Initialize weight vector w = 0
Loop for K iterations
    Loop For all training examples x_i
        if sign(w * x_i) != y_i        Error-driven!
            w += y_i * x_i
```

The Intuition

- For a given example, makes a prediction, then checks to see if this prediction is correct.

- If the prediction is correct, do nothing.
- If the prediction is incorrect, change its parameters so that it would do better on this example next time around.

Perceptron (vs. LR)

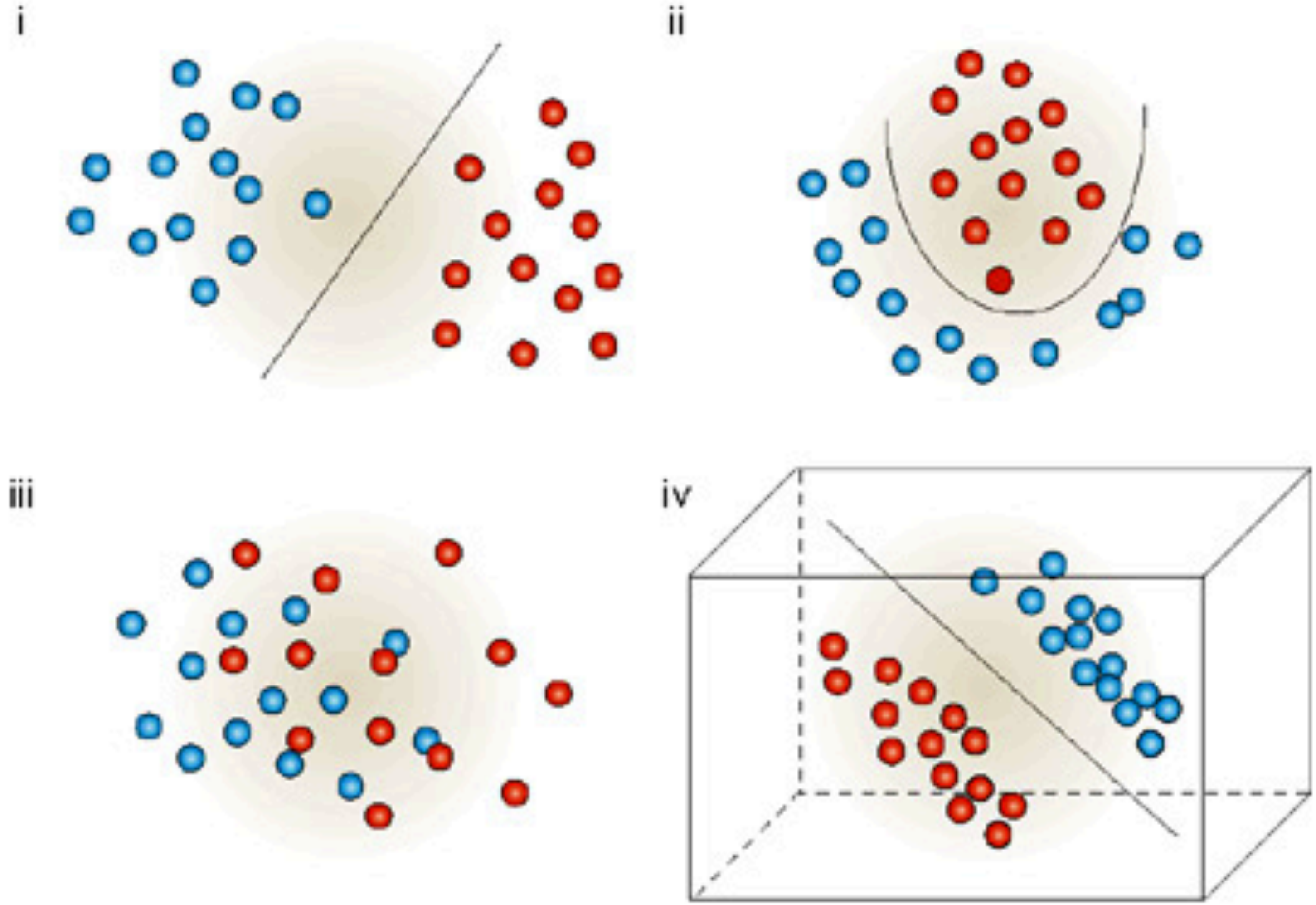
- Only hyperparameter is maximum number of iterations (LR also needs learning rate)

Perceptron (vs. LR)

- Only hyperparameter is maximum number of iterations (LR also needs learning rate)

- Guaranteed to converge if the data is linearly separable (LR always converge)

Linear Separability



What does “converge” mean?

- It means that it can make an entire pass through the training data without making any more updates.
- In other words, it has correctly classified every training example.
- Geometrically, this means that it was found some hyperplane that correctly segregates the data into positive and negative examples

What if non linearly separable?

- In real-world problem, this is nearly always the case.
- The perceptron will not be able to converge.

Q: Then, when to stop?