Lecture 8
Logistic Regression

CS 6320

Logistic Regression

- Logistic regression is a discrimination classifier – it distinguishes classes based on discriminative features.
- For document classification

  $$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \, \mathrm{P(c|d)}$$

  computes directly $\mathrm{P(c|d)}$
- Components of probabilistic ML classifier
  1. For each input observation x^j form a feature vector $[x_1, x_2, \dots x_n]$ denoted as f_i, or x_i^j
  2. A classification function that computes $\hat{y}$ sigmoid and softmax.
  3. An objective function for learning. It minimizes training error. Use cross-entropy loss function.
  4. An algorithm for optimizing the objective function. Use stochastic gradient descent.

Classification – the Sigmoid

- We want to calculate

$P(y=1|x)$ and $P(y=0|x)$

$$z = \left(\sum_{i=1}^{n} w_i x_i\right) + b$$

$$z = w \cdot x + b$$

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

Features of sigmoid

1. Maps a real value number into range [0,1]
2. It is differentiable
3. It is a probability – sums to 1

Classification – the Sigmoid

$$
\begin{aligned}
P(y=1) &= \sigma(w \cdot x + b) \\
&= \frac{1}{1+e^{-(w \cdot x + b)}} \\
P(y=0) &= 1 - \sigma(w \cdot x + b) \\
&= 1 - \frac{1}{1+e^{-(w \cdot x + b)}} \\
&= \frac{e^{-(w \cdot x + b)}}{1+e^{-(w \cdot x + b)}}
\end{aligned}
$$

Decision boundary

$$
\hat{y} = \begin{cases} 1 & if\ P(y=1|x) > 0.5 \\ 0 & \text{otherwise} \end{cases}
$$

Sentiment Classification

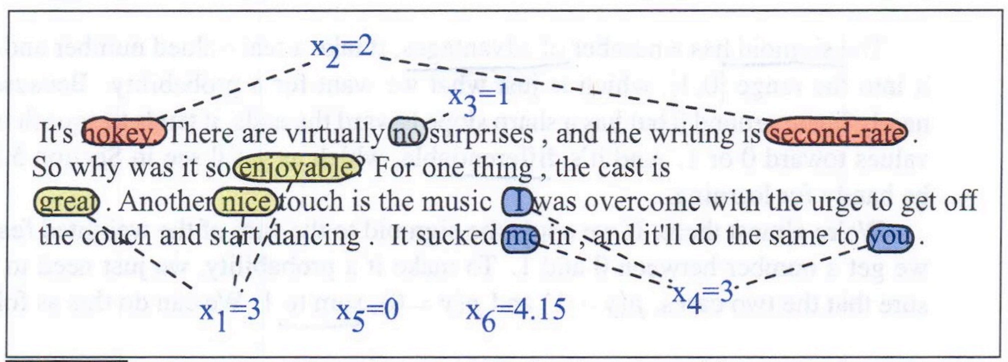


Figure 5.2 A sample mini test document showing the extracted features in the vector *x*.

Var	**Definition**	**Value in Fig 5.2**
x_1	count (positive lexicon) $\in doc$)	3
x_2	count (negative lexicon) $\in doc$)	2
x_3	$\begin{cases} 1\ if\ \text{"no"} \in doc \\ 0\ otherwise \end{cases}$	1
x_4	count (1st and 2nd pronouns $\in doc$)	3
x_5	$\begin{cases} 1\ if\ \text{"!"} \in doc \\ 0\ otherwise \end{cases}$	0
x_6	log(word count of doc)	ln(64) = 4.15

Sentiment Classification

- Assume we learned the 6 weights

$$[w_1 w_2 w_3 w_4 w_5 w_6] = [2.5, -5.0, -1.2, 0.5, 2.0, 0.7]$$

$$P(+|x) = P(Y = 1|x) = \sigma(w \cdot x + b)$$

$$= \sigma([2.5, -5.0 - 1.2, 0.5, 2.0, 0.7] \cdot [3,2,1,3,0,4.15] + 0.1)$$

$$= \sigma(1.805) = 0.86$$

$$P(-|x) = P(Y = 0|x) = 1 - 0.86 = 0.14$$

Designing features

- Any property of the input can be a feature; ie uppercase, punctuation, St. John, etc.

- A feature can also express a complex combination of properties, including exceptions.

- Source of features: linguistic intuitions and linguistic literature on the subject.

- Careful error analysis can provide insights into features.

Choosing a classifier

- Naïve Bayes feature independence requirement is often violated which overestimates evidence.
- Logistic regression is far more robust to correlated features
- Logistic regression works better on large documents, and Naïve Bayes works well on small datasets.
- Naïve Bayes is easy to implement and fast to train.

Learning in Logistic Regression

- Loss function or cost function measures the distance between the system output and the gold output

- Cross-entropy loss function

  $\hat{y} = \sigma(w \cdot x + b)$

  $L(\hat{y}, y)$ $-$ How much $\hat{y}$ differs from the true y

- Mean squared error

  $L_{MSE}(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$

  Not useful for probabilistic classification.

- We seek a convex function.

Cross entropy loss

- Instead of computing $\hat{y} - y$ we maximize $P(y|x)$

- Bernoulli distribution

$$P(y|x) = \hat{y}^{y}(1-\hat{y})^{1-y}$$

$$\log P(y|x) = \log[\hat{y}^{y}(1-\hat{y})^{1-y}] = y\log\hat{y} + (1-y)\log(1-\hat{y})$$

Define cross-entropy loss L_{CE}

$$L_{CE}(\hat{y}, y) = -\log P(y|x) = -[y\log\hat{y} + (1-y)\log(1-\hat{y})]$$

But $\hat{y} = \sigma(w \cdot x + b)$

$$L_{CE}(w, b) = -\big[y\log(w \cdot x + b) + (1-y)\log\big(1 - \sigma(w \cdot x + b)\big)\big]$$

Cross entropy loss

- Expand from one example to the whole training set
  (x^i, y^i) pairs of training features and training label.

  Assume training examples are independent

$$\log P(\text{training labels}) = \log \prod_{i=1}^{m} P\left(y^i | x^i\right)$$

$$= \sum_{i=1}^{m} \log p(y^i \mid x^i)$$

$$= -\sum_{i=1}^{m} L_{ce} p(\hat{y}^i, y^i)$$

Cross entropy loss

- The cost function for the whole dataset is

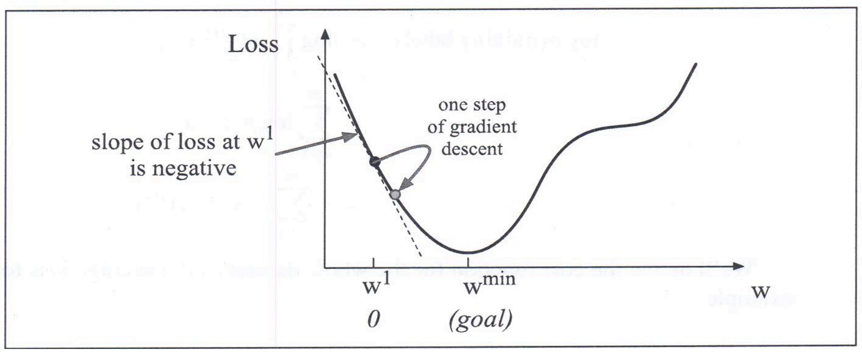
$$Cost(w,b) = \frac{1}{m}\sum_{i=1}^{m} L_{CE}(\hat{y}^i, y^i)$$

$$= -\frac{1}{m}\sum_{i=1}^{m}[y^i \log\sigma\left(w \cdot x^{(i)} + b\right) + \left(1 - y^{(i)}\right)\log\left(1 - \sigma\left(w \cdot x^{(i)} + b\ \right)\right)]$$

Gradient Descent

- Denote $\theta = w, b$ that we need to find

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} L_{CE}(y^{(i)}, x^{(i)}; \theta)$$



- For logistic regression, the loss function is convex.
- To find optimum w, b use an iterative process; start at w^1 and iterate.

$$w^{t+1} = w^t - \eta \frac{d}{dw} f(x; w)$$

Gradient Descent

- Extend from a scalar variable w to many variables.
  The gradient is a vector

$$\nabla_\theta L(f(x;\theta),y)) = \begin{bmatrix} \frac{\partial}{\partial w_1} L\big(f(x;\theta),y\big) \\ \frac{\partial}{\partial w_2} L\big(f(x:\theta),y\big) \\ . \\ . \\ . \\ \frac{\partial}{\partial w_n} L\big(f(x;\theta),y\big) \end{bmatrix}$$

$$\theta^{t+1} = \theta^t - \eta \nabla L(f(x,\theta),y)$$

Gradient Descent

But

$$L_{CE}(w, b) = -\left[y\log\sigma(w \cdot x + b) + (1 - y)\log\left(1 - \sigma(w \cdot b)\right)\right]$$

It can be proven that:

$$\frac{\partial L_{CE}(w,b)}{\partial w_j} = [\sigma(w \cdot x + b) - y]x_j$$

Gradient Descent

- For the entire data set

$$Cost(w,b) = -\frac{1}{m}\sum_{i=1}^{m}[y^i \log\sigma(w \cdot x^i + b) + (1 - y^i)\log(1 - \sigma(w \cdot x + b))]$$

And the gradient for multiple data points is

$$\frac{\partial Cost(w,b)}{\partial w_j} = \sum_{i=1}^{m} [\sigma(w \cdot x^i + b) - y^i]x_j^i$$

Example

From previous text

$x_1 = 3$ count of positive lexicon words
$x_2 = 2$ count of negative lexicon words
$y = 1$

Assume θ^0: $w_1 = w_2 = b = 0$

η=0.1

$\theta^{t+1} = \theta^t - \eta \nabla_\theta L(f(x^{(i)};\theta), y^{(i)})$

$$\nabla_{w,b} = \begin{bmatrix} \frac{\partial L_{CE}(w,b)}{\partial w_1} \\ \frac{\partial L_{CE}(w,b)}{\partial w_2} \\ \frac{\partial L_{CE}(w,b)}{\partial w_3} \end{bmatrix} = \begin{bmatrix} (\sigma(w\cdot x+b)-y)x_1 \\ (\sigma(w\cdot x+b)-y)x_2 \\ (\sigma(w\cdot x+b)-y) \end{bmatrix} = \begin{bmatrix} (\sigma(0)-1)x_1 \\ (\sigma(0)-1)x_2 \\ (\sigma(0)-1 \end{bmatrix} = \begin{bmatrix} -0.5x_1 \\ -0.5x_2 \\ -0.5 \end{bmatrix} = \begin{bmatrix} -1.5 \\ -1.0 \\ -0.5 \end{bmatrix}$$

$$\theta^2 = \begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix} - \eta \begin{bmatrix} -1.5 \\ -1.0 \\ -0.5 \end{bmatrix} = \begin{bmatrix} .15 \\ .1 \\ .05 \end{bmatrix}$$

Regularization

- There is the problem with overfitting.
- A regularization term is added to the objective function to avoid overfitting, and to penalize large weights.
- L2 regularization

$$R(w) = ||W||_2^2 = \sum_{j=1}^{N} w_j^2$$ - Euclidean distance

Objective function becomes:

$$\hat{w} = \underset{w}{\operatorname{argmax}} \left[\sum_{i=1}^{m} \log P(y^i | x^i) \right] - \alpha \sum_{j=1}^{n} w_j^2$$

- L1 regularization

$$R(W) = ||W||_1 = \sum_{i=1}^{N} |w_i|$$

$$\hat{w} = \underset{w}{\operatorname{argmax}} \left[\sum_{1=i}^{m} \log P(y^i | x^i) \right] - \alpha \sum_{j=1}^{n} |w_j|$$

Multinomial logistic regression

- More than two classes

  $c \in C, \qquad P(y = c|x)$

- Use the softmax function, a generalization of the sigmoid.

  Softmax takes a vector $z = [z_1, z_2, \ldots, z_k]$ of K values and maps them to a probability distribution, with all values summing up to 1.

$$softmax(z_i) = \frac{e^{z_i}}{\sum_{i=1}^{k} e^{z_i}} \qquad 1 \le i \le k$$

For $z = [z_1, z_2, \ldots, z_k]$ the output

$$softmax(z \ ) = \frac{e^{z_1}}{\sum_{i=1}^{k} e^{z_i}}, \ldots, \frac{e^{z_k}}{\sum_{i=1}^{k} e^{z_i}}$$

Multinomial logistic regression

- Example

  z = [0.6, 1.1, -1.5, 1.2, 3.2, -1.1]

  softmax (z) = [0.055, 0.09, 0.0067, 0.10, 0.74, 0.01]

For $c \in C$ classes

$$P(y = c \mid x) = \frac{e^{w_c \cdot x + b_c}}{\sum_{j=1}^{k} e^{w_j \cdot x + b_j}}$$