Machine learning - HT 2016
6. Classification: Logistic Regression

Varun Kanade

University of Oxford
February 10, 2016

Outline

Today we'll discuss classification using logistic regression.

- Discriminative vs Generative Models
- Likelihood of Logistic Regression
- Using convex optimization to the obtain MLE
- Logistic Regression in torch

Classification : Generative Models

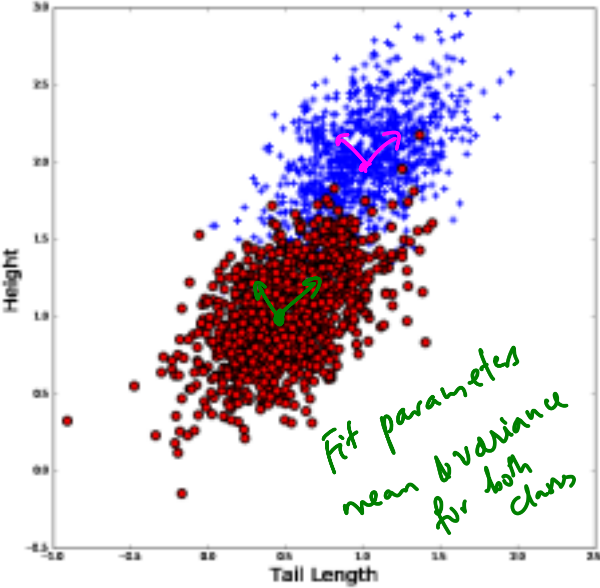
How are the inputs, tail length and height, distributed given the class?

Model $\Pr(\mathbf{x} \mid y = \text{zebra})$

Model $\Pr(\mathbf{x} \mid y = \text{donkey})$

Example: Model both distributions are multivariate normal with same covariance matrix but different mean

Classification : Generative Models



$\mathbb{P}(x|c=1)$

$\mathbb{P}(x|c=2)$.

Predict class using Bayes Rule.

Discriminative Approach

Don't try to model the inputs $\mathbf{x}$ at all

Model the output y given the input $\mathbf{x}$ and the parameters for the model $\mathbf{w}$

$$y \sim p(\mathbf{x}, \mathbf{w})$$

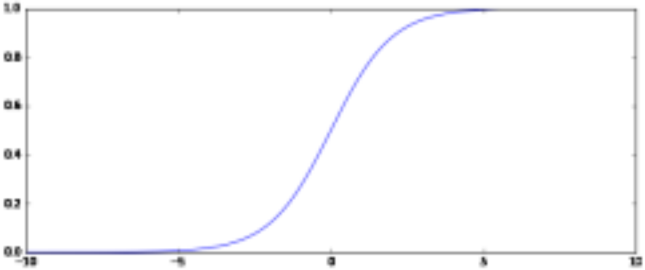
Pros and cons for both approaches (see Murphy Chapter 8.6)

Focus on discriminative classification

Logistic Regression: Sigmoid Function

The sigmoid function, or σ, (a.k.a. logistic or logit) is defined as

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



Binary Classification: Logistic Regression

As in the case of linear regression, we model y given $\mathbf{x} \in \mathbb{R}^n$ and parameters $\mathbf{w} \in \mathbb{R}^n$

Linear model parametrized by $\mathbf{w} \in \mathbb{R}^n$ composed with sigmoid filter

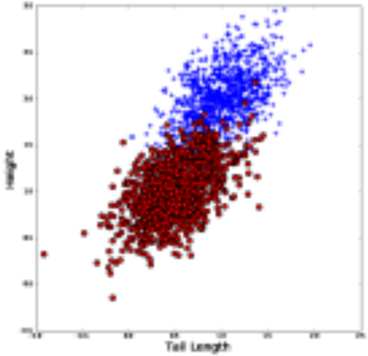
We have,

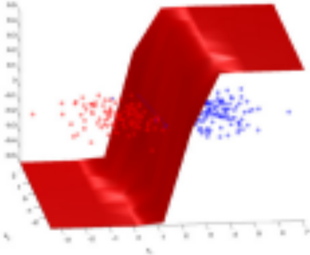
$$\Pr(y = 1 \mid \mathbf{x}, \mathbf{w}) = \sigma(\mathbf{x}^T \mathbf{w})$$

For prediction:

$$\hat{y} = \mathbb{I}(\sigma(\mathbf{x}^T \mathbf{w}) \geq \tfrac{1}{2})$$

Binary Classification : Logistic Regression





$$\hat{y} = \mathbb{I}\left(\frac{1}{1+e^{-x\cdot w}} \geq \frac{1}{2}\right) = \mathbb{I}\left(\bar{x}\cdot\bar{w} \geq 0\right)$$

separating surface is a hyperplane

Bernoulli Random Variables

Bernoulli random variable X takes value in $\{0, 1\}$. We parametrize using $\theta \in [0, 1]$.

$$p(1 \mid \theta) = \theta$$
$$p(0 \mid \theta) = 1 - \theta$$

More succinctly, we can write

$$p(x \mid \theta) = \theta^x (1 - \theta)^{1-x}$$

Logistic Regression

y given $\mathbf{x}$ and parameter $\mathbf{w}$ is modelled as Bernoulli variable

$$y \sim \mathrm{Bernoulli}(\sigma(\mathbf{x}^T \mathbf{w}))$$

Likelihood of Logistic Regression

Given data $\mathcal{D} = \langle(\mathbf{x}_i, y_i)\rangle_{i=1}^{m}$ we can compute the likelihood of observing $\mathbf{y}$ under the logistic regression model

$y \in \{0,1\}^m$ $\quad X = \begin{bmatrix} x_1^T \\ \vdots \\ x_m^T \end{bmatrix}$

$$p(\mathbf{y} \mid \mathbf{X}, \mathbf{w}) = \prod_{i=1}^{m} \mathrm{Bernoulli}(y_i \mid \sigma(\mathbf{x}_i^T \mathbf{w}))$$
$$= \prod_{i=1}^{m} \left(\frac{1}{1 + e^{-\mathbf{x}_i^T \mathbf{w}}}\right)^{y_i} \left(1 - \frac{1}{1 + e^{-\mathbf{x}_i^T \mathbf{w}}}\right)^{1-y_i}$$

$\pi_i = \frac{1}{1 + e^{-x_i^T w}}$

Let's look at the negative log likelihood for a single data point $(\mathbf{x}_i, y_i)$

$$\mathrm{L}(\mathbf{w}; \mathbf{x}_i, y_i) = -\log(p(y_i \mid \sigma(\mathbf{x}_i^T \mathbf{w})))$$
$$= -(y_i \log(\pi_i) + (1 - y_i)\log(1 - \pi_i))$$

cross entropy $\quad KL(P_y \,\|\, \pi)$

$P_y(y) = 1$
$P_y(1-y) = 0$
point distribution.

Gradient and Hessian of NLL

The negative log likelihood is given by

$$\mathrm{L}(\mathbf{w}) = \mathrm{NLL}(\mathbf{y} \mid \mathbf{X}, \mathbf{w}) = -\sum_{i=1}^{m} (y_i \log(\pi_i) + (1 - y_i) \log(1 - \pi_i))$$

The gradient and the Hessian (with respect to $\mathbf{w}$) can be computed as:

$$\mathbf{g} = \nabla_{\mathbf{w}} \mathrm{L} = \sum_{i=1}^{m} \mathbf{x}_i(\pi_i - y_i) = \mathbf{X}^T(\boldsymbol{\pi} - \mathbf{y})$$

"Residual terms"

$$\mathbf{H} = \nabla^2_{\mathbf{w}} \mathrm{L} = \sum_{i=1}^{m} \pi_i(1 - \pi_i)\mathbf{x}^i \mathbf{x}_i^T = \mathbf{X}^T \mathrm{diag}(\pi_i(1 - \pi_i))\mathbf{X}$$

Homework: Show that $\mathbf{H}$ is positive definite.

NLL is convex and has a global minimum

Iteratively Reweighted Least Squares (IRLS)

Apply Newton's method

$$\mathbf{g}_t = \mathbf{X}^T(\boldsymbol{\pi}_t - \mathbf{y}) = -\mathbf{X}^T(\mathbf{y} - \boldsymbol{\pi}_t)$$
$$\mathbf{H}_t = \mathbf{X}^T\mathbf{S}_t\mathbf{X}$$

$S_t = \begin{bmatrix} \ddots & & 0 \\ & \pi_{t,i}(1-\pi_{t,i}) & \\ 0 & & \ddots \end{bmatrix}$

Newton's update says:

$$\begin{aligned}\mathbf{w}_{t+1} &= \mathbf{w}_t - \mathbf{H}_t^{-1}\mathbf{g}_t \\ &= \mathbf{w}_t + (\mathbf{X}^T\mathbf{S}_t\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{y} - \boldsymbol{\pi}_t) \\ &= (\mathbf{X}^T\mathbf{S}_t\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{S}_t\mathbf{X}\mathbf{w}_t + \mathbf{y} - \boldsymbol{\pi}_t) = (\mathbf{X}^T\mathbf{S}_t\mathbf{X})^{-1}(\mathbf{X}^t\mathbf{S}_t\mathbf{z}_t)\end{aligned}$$

Form of least square solution.

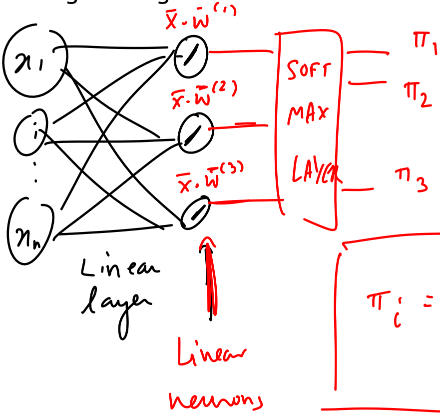
This is a least square solution for the system

$$\sum_{i=1}^{m} S_{t,i}(\mathbf{x}_i^T\mathbf{w} - z_{t,i})^2$$

weighted least square problem

Multi-class, Softmax Formulation, Multinoulli[1]

Logistic Regression as a Neural Network



$y \in \{1, 2, 3\}$

$$\mathbb{I}(y=c) = \begin{cases} 1 & \text{if } y=c \\ 0 & \text{o/w} \end{cases}$$

CROSS ENTROPY

$$-\sum_{c=1}^{3} \mathbb{I}(y=c)\log(\pi_c)$$

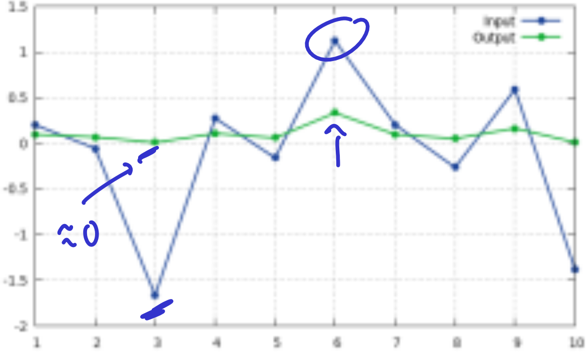
$$\pi_i = \frac{\exp(\bar{x}\cdot\bar{w}^{(i)})}{\sum_j \exp(\bar{x}\cdot\bar{w}^{(j)})}$$

← SOFT MAX

[1] Kevin Murphy's usage

Softmax in Torch

```
nn.SoftMax()
```



Likelihood for multi-class

Classes: $\{1, \ldots, C\}$

Indicator function:

$$\mathbb{I}_c(y) = \begin{cases} 1 & \text{if } y = c \\ 0 & \text{otherwise} \end{cases}$$

The parameters $\mathbf{W}$ is now a $n \times C$ matrix

For a single data point $(\mathbf{x}, y)$ the likelihood is:

$$p(y \mid \mathbf{x}, \mathbf{W}) = \prod_{c=1}^{C} \pi_c^{\mathbb{I}_c(y)}$$

And the negative log likelihood is

$$\mathrm{L}(\mathbf{W}; \mathbf{x}, \mathbf{y}) = -\sum_{c=1}^{C} \mathbb{I}_c(y) \log(\pi_c)$$

Multiclass Logistic Regression in Torch

example-logistic-regression.lua

```lua
require 'nn'; require 'optim';
model = nn.Sequential()
ninputs = 10; noutputs = 3
model:add(nn.Linear(ninputs, noutputs))
model:add(nn.LogSoftMax())
criterion = nn.ClassNLLCriterion()
-- define some input and target
-- to evaluate model
model:forward(input)
-- to evaluate loss
criterion:forward(model:forward(input), target)
-- to compute gradients
model:backward(input, criterion:backward(model.output, target))
```