Learning from Data: Learning Logistic Regressors

Amos Storkey, School of Informatics

November 1, 2005

http://www.anc.ed.ac.uk/∼amos/lfd/

Learning Logistic Regressors

- $P(t|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + b)$. Want to learn $\mathbf{w}$ and b using training data.
- As before:
  - Write out the model and hence the likelihood.
  - Find the derivatives of the log likelihood w.r.t the parameters.
  - Adjust the parameters to maximize the log likelihood.

Likelihood

- Assume data is independent and identically distributed.
- The likelihood is

$$p(D) = \prod_{i=1}^{N} P(t^i|\mathbf{x}^i) = \prod_{i=1}^{N} P(t=1|\mathbf{x}^i)^{t^i} \left(1 - P(t=1|\mathbf{x}^i)\right)^{1-t^i} \qquad (1)$$

- Hence the log likelihood is

$$\log P(D) = \sum_{i=1}^{N} t^i \log P(t=1|\mathbf{x}^i) + (1-t^i) \log\left(1 - P(t=1|\mathbf{x}^i)\right) \qquad (2)$$

Logistic Regression Log Likelihood

- Using our assumed logistic regression model, the log likelihood becomes

$$L = \log P(D|\mathbf{w}, b) = \sum_{i=1}^{N} t^i \log \sigma(b + \mathbf{w}^T \mathbf{x}^i) + (1 - t^i) \log \left(1 - \sigma(b + \mathbf{w}^T \mathbf{x}^i)\right) \quad (3)$$

- We wish to maximise this value w.r.t the parameters **w** and *b*.
- Cannot do this explicitly as before. Use an iterative procedure.

Gradients

- As before we can calculate the gradients of the log likelihood.
- Gradient of sigmoid is $\sigma'(x) = \sigma(x)(1 - \sigma(x))$.

$$\nabla_{\mathbf{w}} L = \sum_{i=1}^{N} (t^i - \sigma(b + \mathbf{w}^T \mathbf{x}^i))\mathbf{x}^i \tag{4}$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{N} (t^i - \sigma(b + \mathbf{w}^T \mathbf{x}^i)) \tag{5}$$

- This cannot be solved directly to find the maximum.
- Have to revert to an iterative procedure. - E.g. Gradient Ascent

Gradient Ascent

- Consider the likelihood as a surface: a function of the parameters.
- Want to find the maximum likelihood value. In other words we want to find the highest point of the likelihood surface - the top of the hill.
- We propose a dumb hill climbing approach. Make sure you take each step in the steepest direction (locally).
- Eventually we will get to a point where whatever direction we step in will take us down. We are at *a* top.
- Note we are not necessarily at *the* top, but are at *a* top. We ignore this issue for the moment.

Gradient Ascent for Logistic Regression

- Choose some step size (or more accurately a learning rate) η.
- Initialise at some position in parameter space. Presume we are in position $(\mathbf{w}, b)$.
- At each step, move to position

$$\mathbf{w}^{new} = \mathbf{w} + \eta \nabla_{\mathbf{w}} L \quad (6)$$

$$b^{new} = b + \eta \frac{\partial L}{\partial b} \quad (7)$$

- Iterate the stepping until some stopping criterion is reached. This might be when **w** and *b* don't change much anymore (equivalently all the partial derivatives are nearly zero).

Problems

- Local minima: luckily there are none for logistic regression, but there can be for other models.
- Need to set the learning rate:
  - Too small: never get there.
  - Too large: gradient information ceases to be of much use. Keep jumping about somewhat randomly.
- A learning rate of 0.1 is a good starting point.
- Naively this approach might seem like a good idea.
- In fact a pretty bad optimization approach. Will discuss conjugate gradient and pseudo-Newton methods.

Batch or Online

- Batch: update using all the training data.

$$\mathbf{w}^{new} = \mathbf{w} + \eta \sum_{i=1}^{N} (t^i - \sigma(b + \mathbf{w}^T \mathbf{x}^i))\mathbf{x}^i \quad (8)$$

$$b^{new} = b + \eta \sum_{i=1}^{N} (t^i - \sigma(b + \mathbf{w}^T \mathbf{x}^i)) \quad (9)$$

- Online: make an update using one training example at a time.

$$\mathbf{w}^{new} = \mathbf{w} + \eta/N(t^i - \sigma(b + \mathbf{w}^T \mathbf{x}^i))\mathbf{x}^i \quad (10)$$

$$b^{new} = b + \eta/N(t^i - \sigma(b + \mathbf{w}^T \mathbf{x}^i)) \quad (11)$$

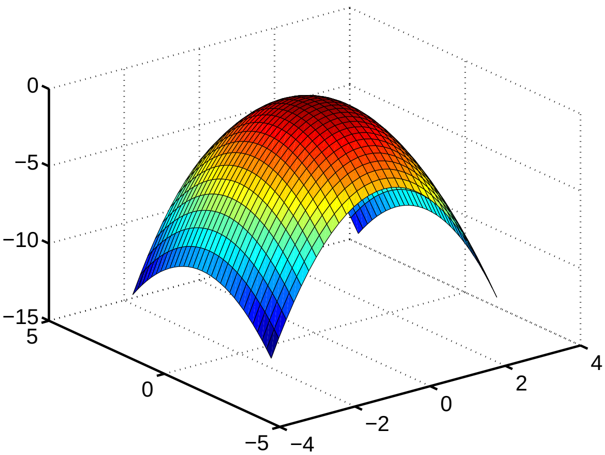
What shape is the likelihood surface

- Calculate the Hessian (matrix of second derivatives)

$$H_{ij} = \frac{\partial^2 L}{\partial w_i w_j} = -\sum_{ij\mu} \mathbf{x}_i^\mu \mathbf{x}_j^\mu \sigma(b + \mathbf{w}^T \mathbf{x}^\mu)(1 - \sigma(b + \mathbf{w}^T \mathbf{x}^\mu))$$

- Always negative definite: second derivatives in any direction at any point are negative.
- Hence likelihood surface is convex: only one peak. No local maxima.
- Bowl shaped (upside down!).

Convex Likelihood Surface

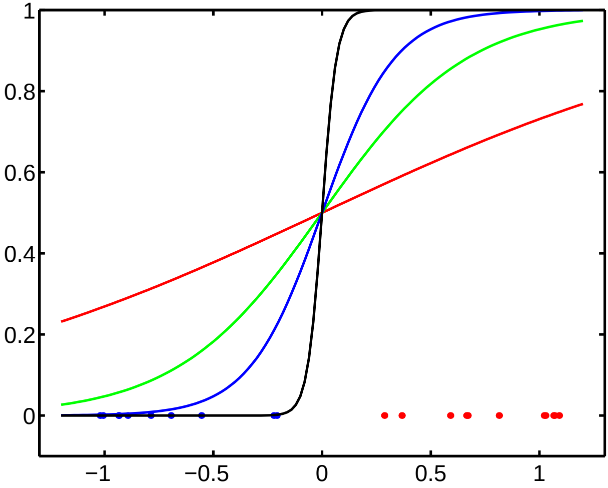


The likelihood surface has no local minima

Linear separability

- The decision boundary is a hyperplane
- Data is *linearly separable* if some hyperplane can divide the two classes perfectly.
- The maximum likelihood logistic regressor for linearly separable training data is a perceptron. The firmer the decision, the more probable the data.
- Linear separability might occur just because of limited training data

Maximum Likelihood for Linear Seperability



Regularisation and prior belief

- What if we believe that the classification should not be certain.
- For example we could know that in general the data would not be linearly separable: just that a finite training set might be.
- This is prior information about the parameter.
- Hence we have some model $P(\mathbf{w})$, which is low for large $|\mathbf{w}|$. E.g. $P(\mathbf{w})$ is Gaussian.
- This actually amounts to adding a penalty term $-\alpha \mathbf{w}^T \mathbf{w}$ to the likelihood.
- This is called *regularisation*.

Summary

- Likelihood for logistic regression.
- Derivatives of the log likelihood
- Using derivatives for gradient ascent.
- Perceptron
- Regularisation