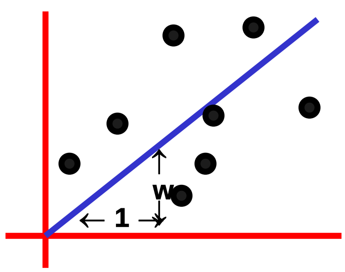
Linear and Nonlinear Regression and Classification

Note to other teachers and users of these slides. Andrew would be delighted if you found this source material useful in giving your own lectures. Feel free to use these slides verbatim, or to modify them to fit your own needs. PowerPoint originals are available. If you make use of a significant portion of these slides in your own lecture, please include this message, or the following link to the source repository of Andrew's tutorials: http://www.cs.cmu.edu/~awm/tutorials . Comments and corrections gratefully received.

Ronald J. Williams
CSG220, Spring 2007

Containing a number of slides adapted from the Andrew Moore tutorial "Regression and Classification with Neural Networks"

Linear Regression



DATASET

inputs	outputs
$x_1 = 1$	$y_1 = 1$
$x_2 = 3$	$y_2 = 2.2$
$x_3 = 2$	$y_3 = 2$
$x_4 = 1.5$	$y_4 = 1.9$
$x_5 = 4$	$y_5 = 3.1$

Linear regression assumes that the expected value of the output given an input, *E[y|x]*, is linear.

Simplest case: Out(*x*) = *wx* for some unknown *w*.

Given the data, we can estimate *w*.

1-parameter linear regression

Assume that the data is formed by

$$y_i = wx_i + \text{noise}_i$$

where…

- the noise signals are independent
- the noise has a normal distribution with mean 0 and unknown variance σ^2

$P(y|w,x)$ has a normal distribution with

- mean wx
- variance σ^2

Bayesian Linear Regression

$$P(y|w,x) = \text{Normal (mean } wx\text{, var } \sigma^2)$$

We have a set of datapoints (x_1,y_1) (x_2,y_2) … (x_R,y_R) which are EVIDENCE about w.

We want to infer w from the data.

$$P(w|x_1, x_2, x_3,\ldots x_R, y_1, y_2 \ldots y_R)$$

- You can use BAYES rule to work out a posterior distribution for w given the data.
- Or you could do Maximum Likelihood Estimation

Maximum likelihood estimation of *w*

Asks the question:

"For which value of *w* is this data most likely to have happened?"

<=>

For what *w* is

$P(y_1, y_2 \ldots y_R \mid x_1, x_2, x_3, \ldots x_R, w)$ maximized?

<=>

For what *w* is

$$\prod_{i=1}^{n} P(y_i \mid w, x_i) \text{ maximized?}$$

For what *w* is

$$\prod_{i=1}^{R} P(y_i \mid w, x_i) \text{ maximized?}$$

For what *w* is

$$\prod_{i=1}^{R} \exp\left(-\frac{1}{2}\left(\frac{y_i - wx_i}{\sigma}\right)^2\right) \text{ maximized?}$$

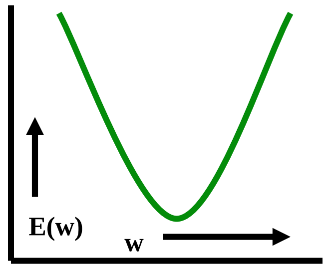
For what *w* is

$$\sum_{i=1}^{R} -\frac{1}{2}\left(\frac{y_i - wx_i}{\sigma}\right)^2 \text{ maximized?}$$

For what *w* is

$$\sum_{i=1}^{R} \left(y_i - wx_i\right)^2 \text{ minimized?}$$

Linear Regression



The maximum likelihood *w* is the one that minimizes sum-of-squares of <u>residuals</u>

$$E = \sum_i (y_i - wx_i)^2$$

$$= \sum_i y_i^2 - \left(2\sum x_i y_i\right)w + \left(\sum x_i^2\right)w^2$$

We want to minimize a quadratic function of *w*.

Linear Regression

Easy to show the sum of squares is minimized when

$$w = \frac{\sum x_i y_i}{\sum x_i^2}$$

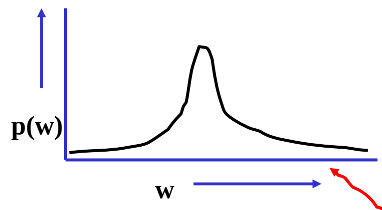
The maximum likelihood model is $\text{Out}(x) = wx$

We can use it for prediction

Linear Regression

Easy to show the sum of squares is minimized when

$$w = \frac{\sum x_i y_i}{\sum x_i^{\ 2}}$$



The maximum likelihood model is $\text{Out}(x) = wx$

We can use it for prediction

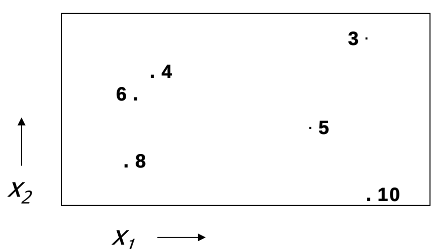
Note: In Bayesian stats you'd have ended up with a prob dist of *w*

And predictions would have given a prob dist of expected output

Often useful to know your confidence. Max likelihood can give some kinds of confidence too.

Multivariate Regression

What if the inputs are vectors?



2-d input example

Dataset has form

$\mathbf{x}_1$	y_1
$\mathbf{x}_2$	y_2
$\mathbf{x}_3$	y_3
.:	:
.	
$\mathbf{x}_R$	y_R

Multivariate Regression

Write matrix X and Y thus:

$$\mathbf{X} = \begin{bmatrix} \ldots\ldots\mathbf{x}_1\ldots\ldots \\ \ldots\ldots\mathbf{x}_2\ldots\ldots \\ \vdots \\ \ldots\ldots\mathbf{x}_R\ldots\ldots \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \ldots & x_{1m} \\ x_{21} & x_{22} & \ldots & x_{2m} \\ & & \vdots & \\ x_{R1} & x_{R2} & \ldots & x_{Rm} \end{bmatrix} \quad \mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_R \end{bmatrix}$$

(There are R datapoints. Each input has m components)

The linear regression model assumes a vector $\boldsymbol{w}$ such that

$$\text{Out}(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} = w_1x[1] + w_2x[2] + \ldots. w_mx[m]$$

The max. likelihood $\boldsymbol{w}$ is $\boldsymbol{w} = (X^TX)^{-1}(X^TY)$

Multivariate Regression (con't)

The max. likelihood $\boldsymbol{w}$ is $\boldsymbol{w} = (X^TX)^{-1}(X^TY)$

X^TX is an m x m matrix: i,j'th elt is $\sum_{k=1}^{R} x_{ki}x_{kj}$

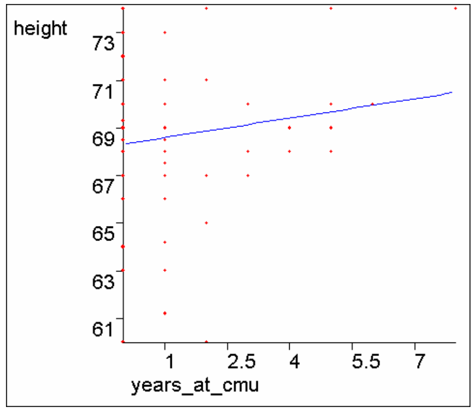
X^TY is an m-element vector: i'th elt is $\sum_{k=1}^{R} x_{ki}y_k$

What about a constant term?

We may expect linear data that does not go through the origin.

Statisticians and Neural Net Folks all agree on a simple obvious hack.

Can you guess??



The constant term

- The trick is to create a fake input "X_0" that always takes the value 1

X_1	X_2	Y
2	4	16
3	4	17
5	5	20

Before:

$Y=w_1X_1+ w_2X_2$

…has to be a poor model

X_0	X_1	X_2	Y
1	2	4	16
1	3	4	17
1	5	5	20

After:

$Y= w_0X_0+w_1X_1+ w_2X_2$

$= w_0+w_1X_1+ w_2X_2$

…has a fine constant term

In this example, You should be able to see the MLE w_0, w_1 and w_2 by inspection

What about higher-order terms?

Maybe we suspect a higher-order polynomial function like

$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$

would fit the data better.

In that case, we can simply perform multivariate linear regression using additional dimensions for all higher-order terms.

Higher-order terms

Linear Fit

1	*X*	*Y*
1	1	2
1	2	5
1	3	10
1	5	26

Quadratic Fit

1	*X*	X^2	*Y*
1	1	1	2
1	2	4	5
1	3	9	10
1	5	25	26

Maximum Likelihood Nonlinear Regression

Assume correct function is $\boldsymbol{y} = f(\boldsymbol{x}, \boldsymbol{w})$, where f is any function of the input $\boldsymbol{x}$ parameterized by $\boldsymbol{w}$, and observations are corrupted by additive Gaussian noise (with some fixed variance σ^2).

For example, f could be the function computed by a multilayer neural network whose weights are $\boldsymbol{w}$.

As before, we would like to determine for what $\boldsymbol{w}$

$$P(\boldsymbol{y}_1, \boldsymbol{y}_2 \ldots \boldsymbol{y}_R \mid \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \ldots \boldsymbol{x}_R, \boldsymbol{w})$$

is maximized.

And just as before, this translates into:

For what ***w*** is

$$\prod_{i=1}^{R} P(\mathbf{y}_i | \mathbf{w}, \mathbf{x}_i) \text{ maximized?}$$

For what ***w*** is

$$\prod_{i=1}^{R} \exp\left(-\frac{1}{2}\left(\frac{\|\mathbf{y}_i - f(\mathbf{x}_i,\mathbf{w})\|}{\sigma}\right)^2\right) \text{ maximized?}$$

For what ***w*** is

$$\sum_{i=1}^{R} -\frac{1}{2}\left(\frac{\|\mathbf{y}_i - f(\mathbf{x}_i,\mathbf{w})\|}{\sigma}\right)^2 \text{ maximized?}$$

For what ***w*** is

$$\sum_{i=1}^{R} \left(\| \mathbf{y}_i - f(\mathbf{x}_i, \mathbf{w}) \|\right)^2 \text{ minimized?}$$

- So, for example, with the usual squared-error measure, backpropagation can be viewed as a technique for searching for a maximum-likelihood fit of a neural network to a given set of training data.
- This applies when neural networks are used for regression, assuming additive Gaussian noise.
- What about for classification?

Maximum Likelihood Probability Estimation

- Consider a 2-class classification problem, and assume that the probability that an instance $\boldsymbol{x}$ is classified as positive has the functional form $y = f(\boldsymbol{x}, \boldsymbol{w})$.
- Then it can be shown that the correct criterion to optimize to generate ML estimates of the probability of belonging to the + class is *not* squared error.

Maximum Cross-Entropy

- Instead the following *cross-entropy* measure should be maximized:

$$\sum_{i=1}^{R} \left(y_i \log f(\mathbf{x}_i, \mathbf{w}) + (1 - y_i) \log(1 - f(\mathbf{x}_i, \mathbf{w})) \right)$$

- In a multilayer neural network, the gradient computation for this measure still follows the backpropagation process.