Lecture 9:

Regression as a Learning Task

Week of February 13, 2023

University California, Berkeley

Machine Learning Algorithms

MSSE 277B, 3 Units

Spring 2023

Prof. Teresa Head-Gordon

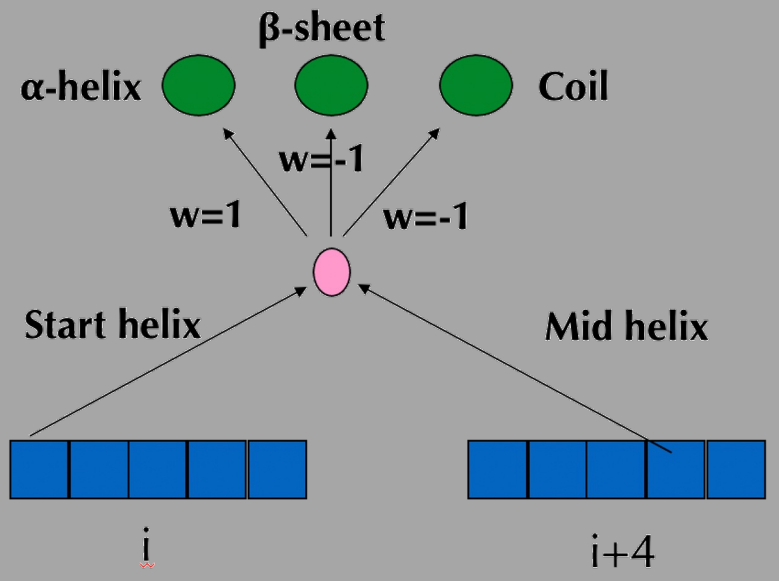
Departments of Chemistry, Bioengineering, Chemical and Biomolecular Engineering

Review of Previous Lecture

Supervised Machine Learning: We showed how ANN is trained with labelled data of input,output variables $\vec{x}_\mu, \vec{y}_\mu$ to learn the general mapping function $\vec{y} = f(\vec{x})$, i.e. to generalize to any new input data $\vec{x}$ to predict $\vec{y}$ for that data.

In the last lecture we used an ANN for classification: every amino acid in a sequence is assigned a discrete secondary structure category. It introduced us to new issues in ML which we will continue grappling with during the semester

- Training and Testing data
- Data quality
- ANN architectures
- Input/Output representation (feature engineering)
- Measuring performance
- Generalization



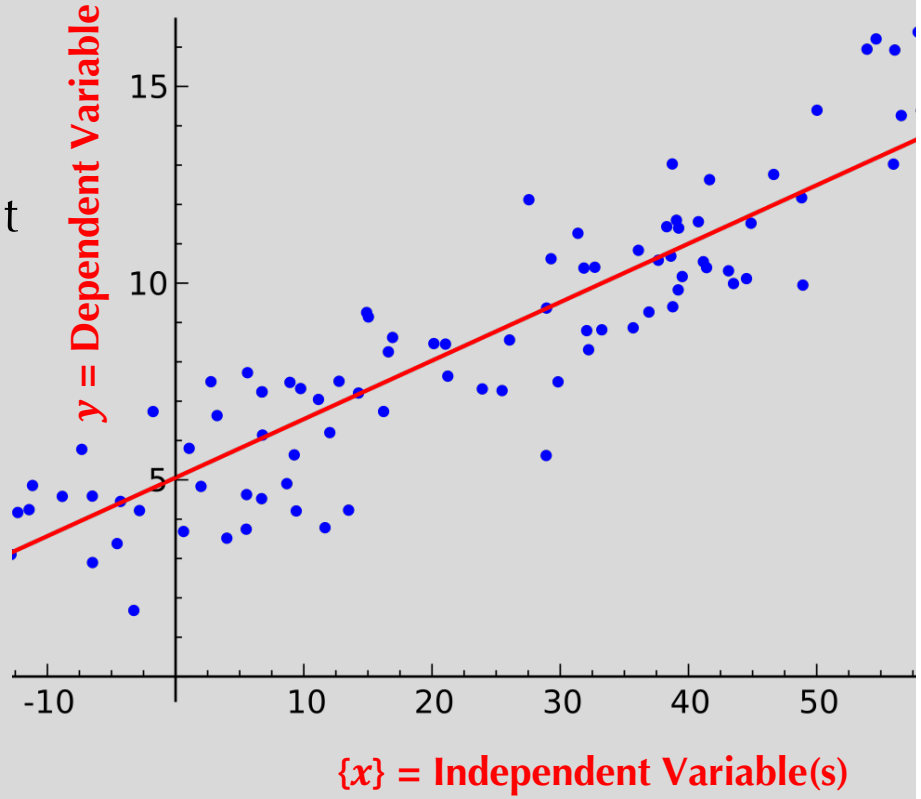
Regression Models

Purpose of Today's Lecture: For classification, we did not care about the values of $\vec{y}$; we just cared that a learning outcome fell into one of the categories.

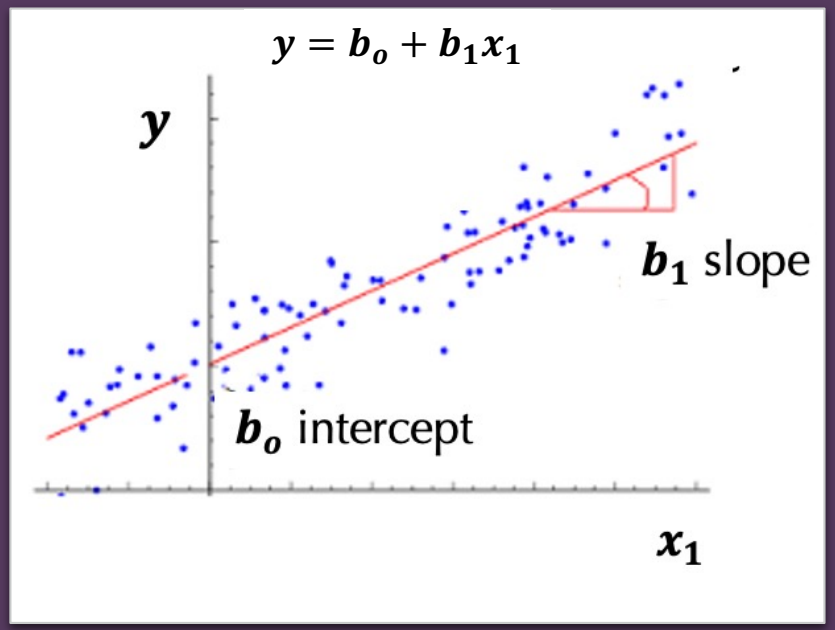
For regression problems, we typically want to learn a real or continuous value for $\vec{y}$.

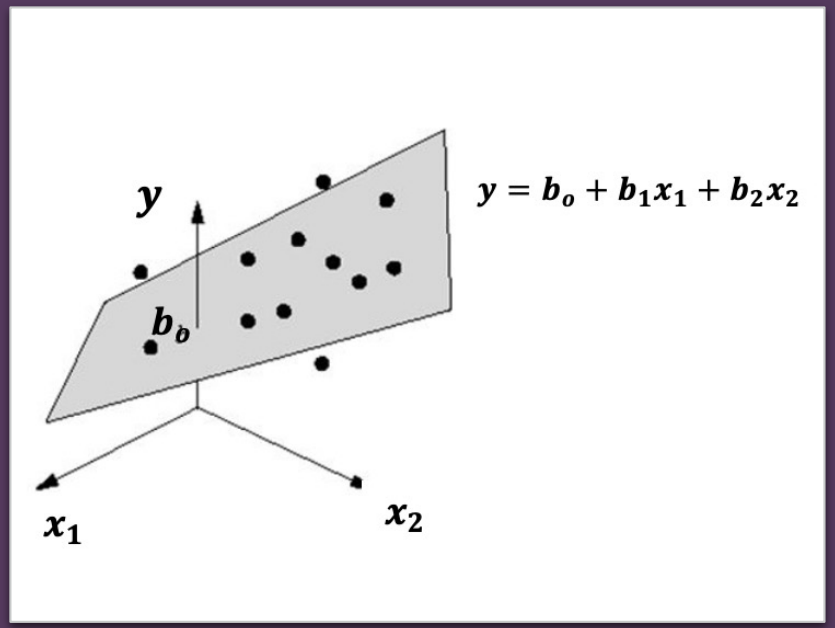
$$\boldsymbol{y = f(x_1, x_2, x_3)}$$

- Suppose we model dependent variable y with limited data, x_1, x_2, x_3. Typically we will not have enough data to directly estimate the true function f.
- Therefore, we have to assume that it has a restricted form, such as linear, often assumed in standard linear regression!



Machine Learning borrows tools from not only optimization but statistical models such as regression, but solves it in different ways!





Linear Regression

Linear regression assumes a relationship between y and x that are linear in the unknown fitting parameters.

Consider a univariate case

$$y = b_o + b_1 x_1$$

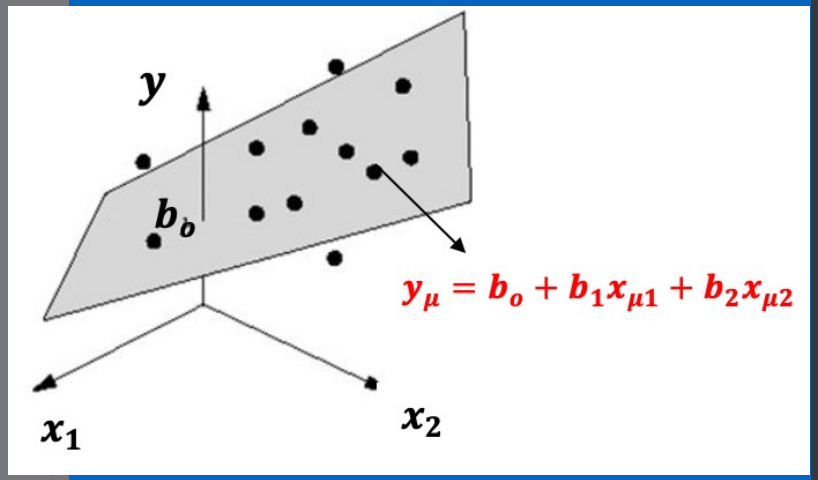
where b_o and b_1 values are model coefficients.

But we may have more "features" $\{x_i\}$ that help explain the y observations. Hence we can do multivariate linear least squares

$$y = b_o + b_1 x_1 + b_2 x_2 + \cdots + b_p x_p$$

If you need to brush up on linear regression basics
https://www.stat.berkeley.edu/~stark/SticiGui/Text/regression.htm

Regression as a Learning Task

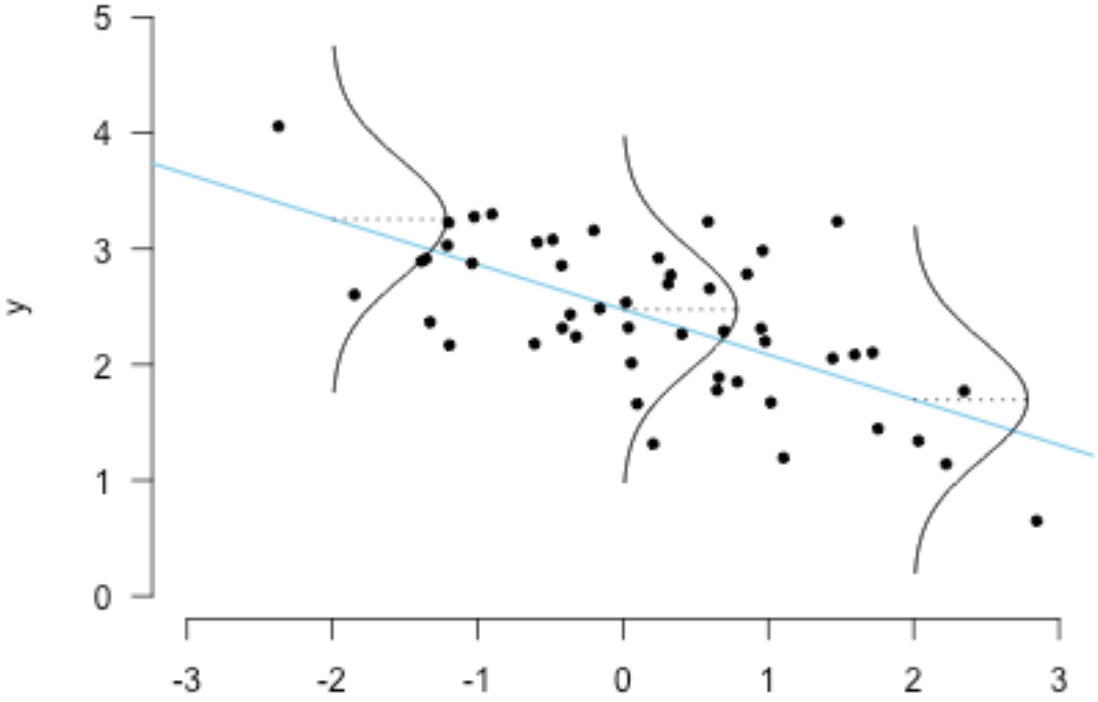


The {b} values are "learned" during the model fitting/training step by presenting it with labeled data $[y_\mu, \{x_{\mu 1}, \ldots x_{\mu p}\}]$.

$$y_\mu = b_o + b_1 x_{\mu 1} + b_2 x_{\mu 2} + \cdots + b_p x_{\mu p}$$

- Supervised learning of linear regression model uses a cost function to minimize error on labelled data
- We can use our favorite gradient descent based optimization method to determine the {b} coefficients

We judge the quality of the linear regression *model:* do particular features raise or lower the cost function; is linear appropriate?



$$\mathbf{L}(m, \sigma^2|\mathbf{y}) = \prod_{\mu}^{N} P_y(y_\mu|m, \sigma^2) = \prod_{\mu}^{N} \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{(y_\mu - m)^2}{2\sigma^2}\right)$$

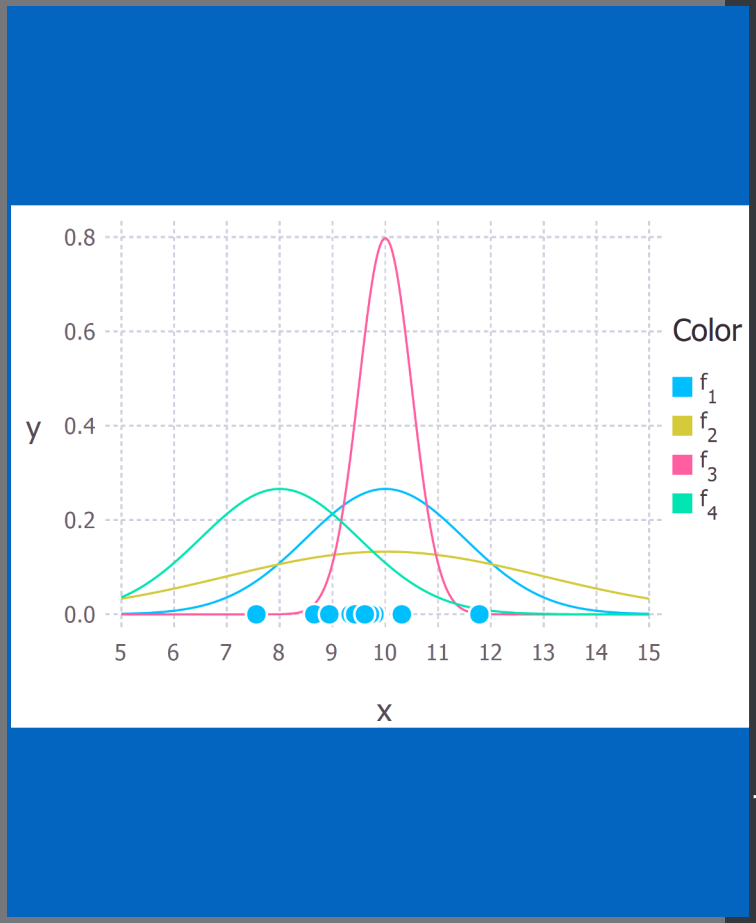
Regression as a Statistical Model

Regression is a statistical model which can be used for either prediction (new value of y given x) or can quantify strength of the relationship between y and x (is it linear?).

- uncertainty in y is modelled as a Gaussian (normal) distribution with a mean m and variance σ^2. The probability of $\{y\}$ points being drawn defines the likelihood function, which is just the multiplication of N normal probability density functions (PDF).

- A good estimate of our unknown statistical model parameters, m and σ^2, is to find them by maximizing the likelihood function for the given data points

Maximizing Likelihood



$L(m, \sigma^2|y)$ achieves same maximum as its logarithm at same point (consider m first)

$$m = \max \mathrm{L}(m, \sigma^2|y) = max \log [\, L(m, \sigma^2|y)]$$

and changes the multiplications to additions

$$m = max \sum_{\mu}^{N} log \frac{1}{\sigma\sqrt{2\pi}} log\, exp\left(-\frac{(y_\mu - m)^2}{2\sigma^2}\right)$$

we can remove any additive or multiplicative constants because they do not affect the *maximum* likelihood

$$m = max \sum_{\mu}^{N} -(y_\mu - m)^2$$

And thus we minimize sum of the squares of the residuals (SSR)!

$$SSR = min \sum_{\mu}^{N} (y_\mu - m)^2$$

This makes sense!

$$\frac{\partial}{\partial m} \sum_{\mu}^{N} (y_\mu - m)^2 = 0;\; m = \frac{1}{N} \sum_{\mu}^{N} y_\mu$$

Ordinary Least Squares

The problem now is trying to find estimates for the $p\{b\}$ parameters instead of just a single m value

$$P_y(y|m,\sigma^2) = P_y\left(b_0 + b_1x_1 + b_2x_2 \ldots + b_px_p|\sigma^2\right)$$

$$m = min\sum_{\mu}^{N}\left(y_\mu - [b_0\cdot 1 + b_1x_{1\mu} + b_2x_{2\mu} \ldots + b_px_{p\mu}]\right)^2$$

$$\{b\} = min\sum_{\mu}^{N}\left(y_\mu - \widehat{y_\mu}\right)^2$$

Where $\widehat{y_\mu}$ is the predicted value of the maximum likelihood, and the set of $\{b\}$ are those that minimize the difference of the training examples $\{y_\mu, x_\mu\}$.

This defines the ordinary least squares method,

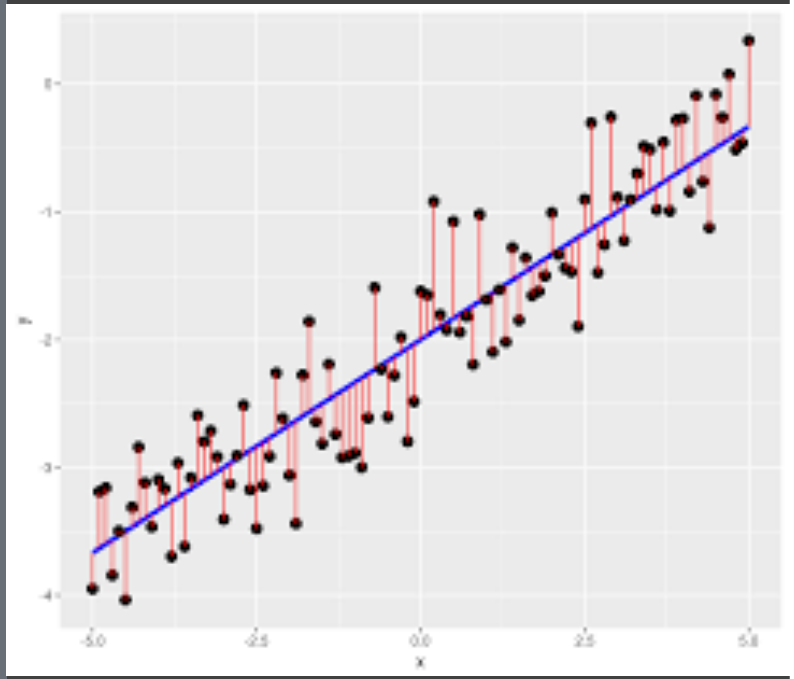
$$SSR = \sum_{\mu}^{N}\sum_{j}^{p}\left(y_\mu - x_{j\mu}b_{j\mu}\right)^2$$

$$= \left(\vec{y} - \vec{x}\vec{b}\right)^T\left(\vec{y} - \vec{x}\vec{b}\right)$$

…and solved as a minimization problem

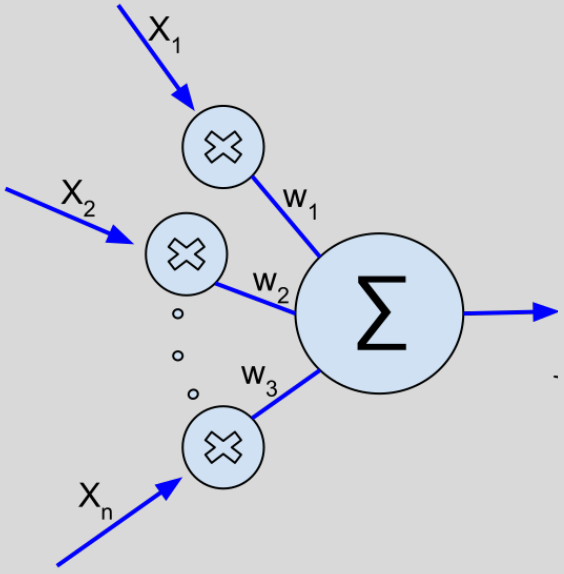
$$\frac{\partial SSR}{\partial \vec{b}} = -2\vec{x}^T\left(\vec{y} - \vec{x}\vec{b}\right) = 0$$

$$\hat{\vec{b}} = (\vec{x}^T\vec{x})^{-1}\vec{x}^T\vec{y}$$



While this is our normal von Neumann machine, we can also solve it using machine learning methods!

Simple Perceptron is Linear Regression



I.e we can solve linear regression as a simple perceptron that uses a linear combination of features to produce one output.

$$\hat{y} = a(\vec{w}^T \cdot \vec{x})$$

Now our task is to find the weights that provide the best fit to our training data!

We had said that the way to measure our fit is to calculate the least squares error (or loss) over our dataset:

$$C = \frac{1}{2}\sum_{\mu}^{N}\sum_{j}^{J}\left[O_j^{\mu} - \hat{y}_j^{\mu}\right]^2$$

Which we now see is our maximum likelihood model for a Gaussian PDF

Simple Perceptron is Linear Regression

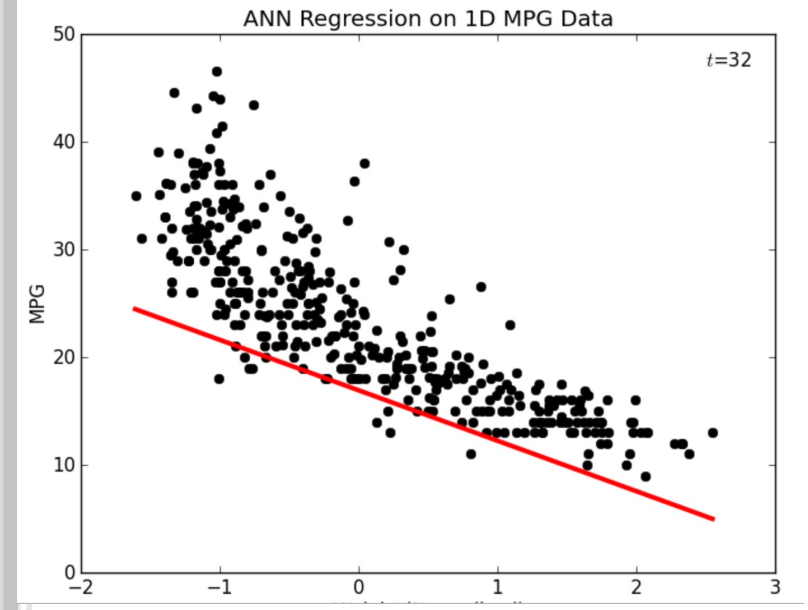
In order to find the line, plane, or hyperplane of best fit, we must minimize C. While we saw that this has a closed-form solution for OLS, in general we can minimize using gradient descent!

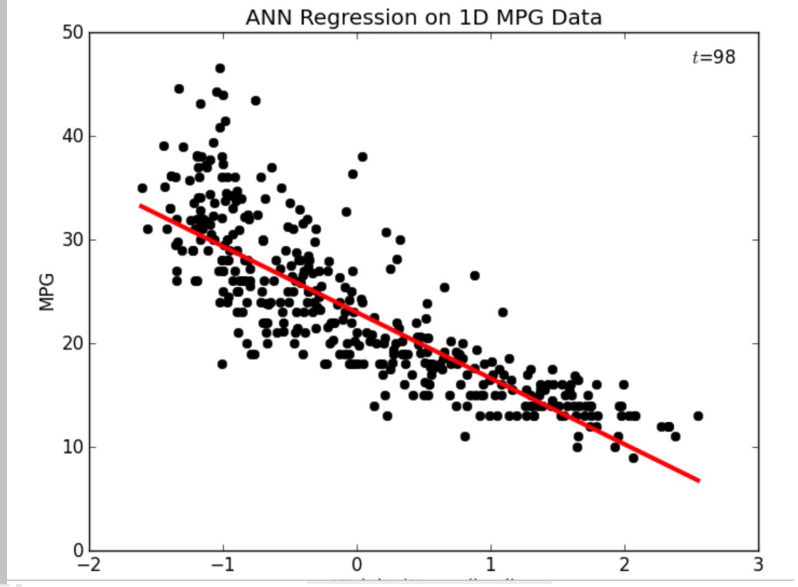
For the simple perceptron with a linear activation function then

$$\delta w_j = -\varepsilon \frac{\partial C}{\partial w_j} = -\varepsilon \sum_{\mu} \left[O_j^{\mu} - \hat{y}\left(a_j^{\mu}\right) \right] x_j$$

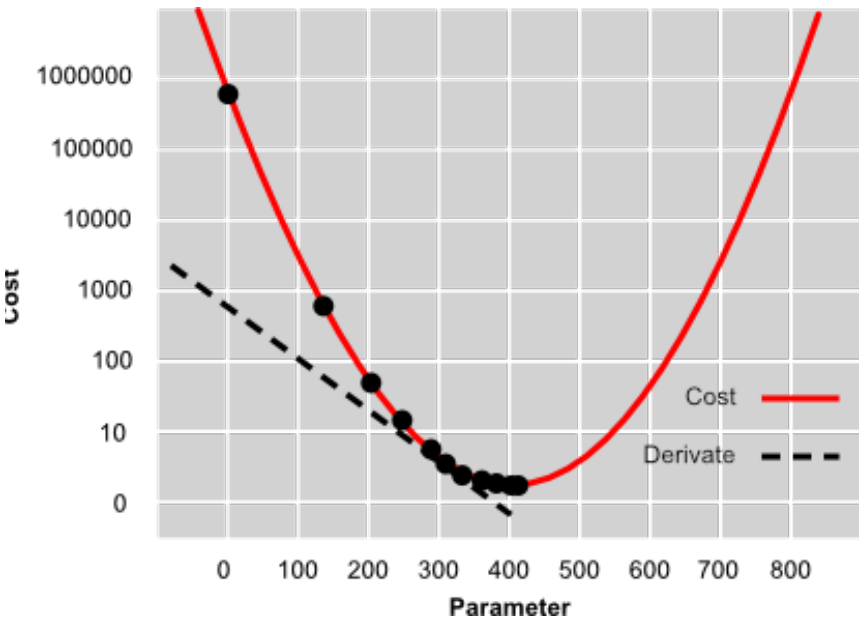
After a set amount of epochs, $\vec{w}$ will define a line or hyperplane of best-fit.

$$a(\vec{w}^T \cdot \vec{x}) = \vec{w}^T \cdot \vec{x}$$





Linear vs. Non-linear Regression



A regression model is linear when all terms in the model are one of the following:

- A constant, b_o
- A parameter b_i multiplied by an independent variable, x_i
- Then, you build the equation by only adding the terms together. These rules defines a regression equation that is linear in the parameters,

$$y = b_o + b_1 x_1 + b_2 x_2 + \cdots + b_p x_p$$

It can be modified to describe curvature through transformations of independent variables, and yet continues to be linear in the parameters.

$$y = \{b\}, \{f(x)\} = b_o + b_1 x_1 + b_2 (x_2)^2 + b_3 (1/x_3)^2 + \cdots + b_p log x_p$$

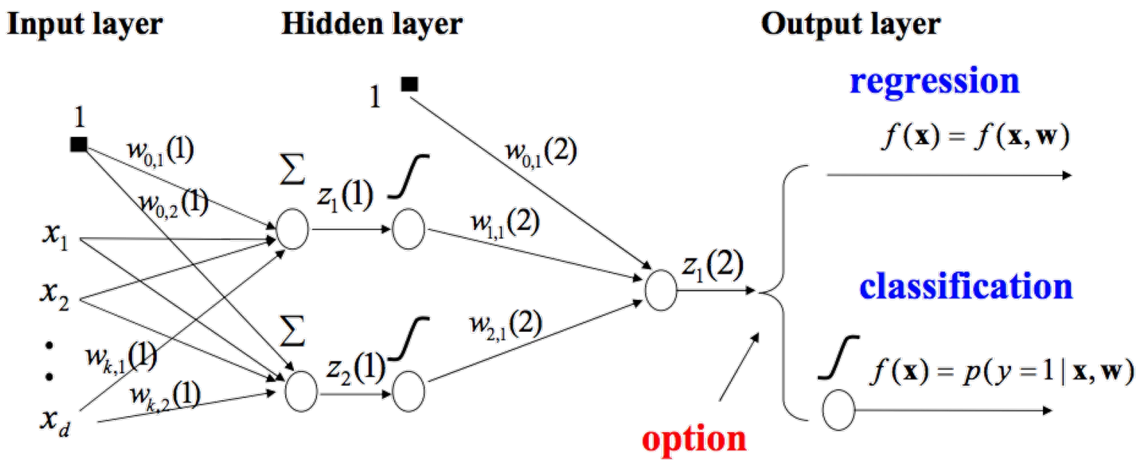
Nonlinear Regression

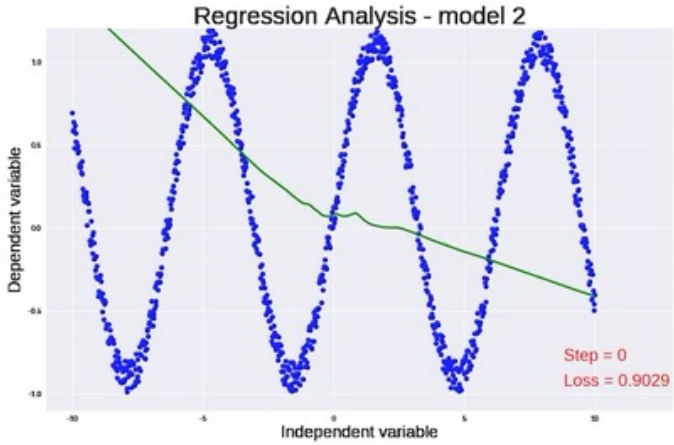
Often linear regression is a poor approximation to the true input and output relationship.

A regression model is non-linear when there are now parameter functions in the model :

$$y = f(\{x\}, \{b\}) = b_o + b_1 \sin(b_2 x_1 + b_3)$$

The beauty of a multi-layer neural network is that we now can build up perceptrons with different activation functions to handle highly non-linear input-output relationships!





ANN's are highly suited to a variety of learning tasks involving either regression or classification (next time!)

Lecture 9 (Part 2):

Classification as a Learning Task

Week of February 14, 2022

University California, Berkeley

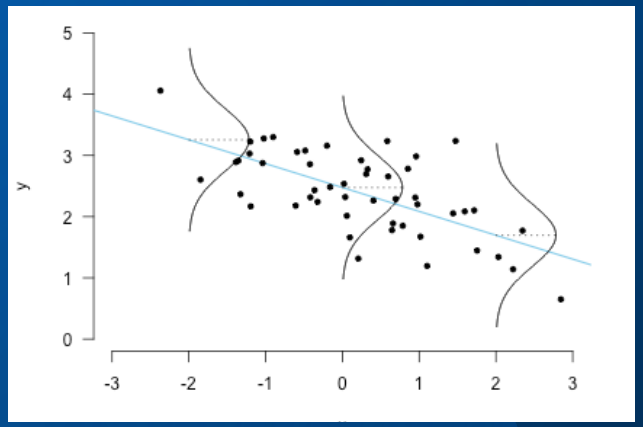
Machine Learning Algorithms

MSSE 277B, 3 Units

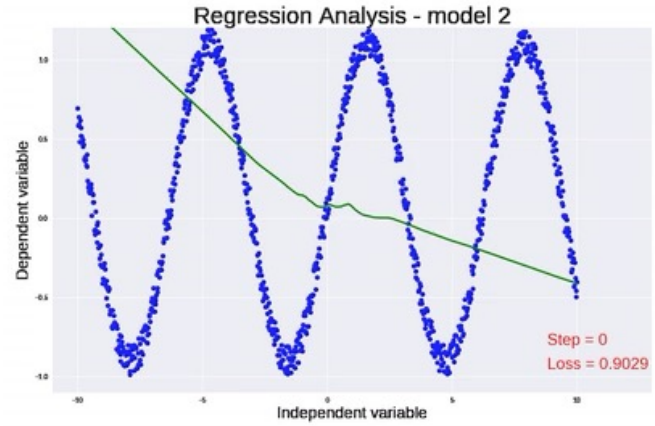
Spring 2022

Prof. Teresa Head-Gordon

Departments of Chemistry, Bioengineering, Chemical and Biomolecular Engineering



Summary of Previous Lecture



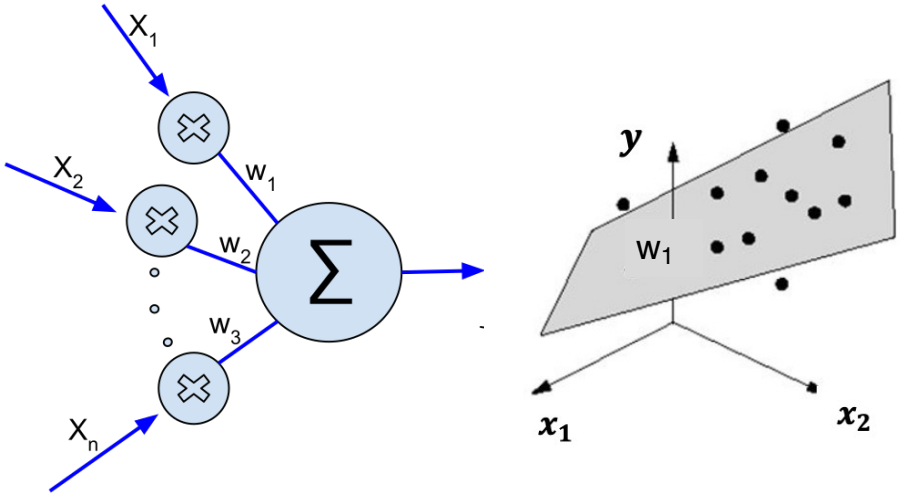
Multivariate linear regression is a way to model input/output relationship based on a statistical model that formulates the degree to which explanatory variables or features {x} can explain the dependent variables {y}.

- For linear regression the underlying statistical model is often Gaussian and leads to an error or cost function that is based on the sum of the squared residuals. The SSR is the cost function that assumes prediction errors are Gaussian distributed. Linear regression has a well-defined analytical solution.

- We showed that a simple perceptron can equivalently solve the linear regression problem.
- But what makes an ANN so appealing is that, with choices of different *activation functions*, and with use of multi-layer networks, we can use the same machinery (cost function, back-propagation) for non-linear regression!

Activation Functions for Regression

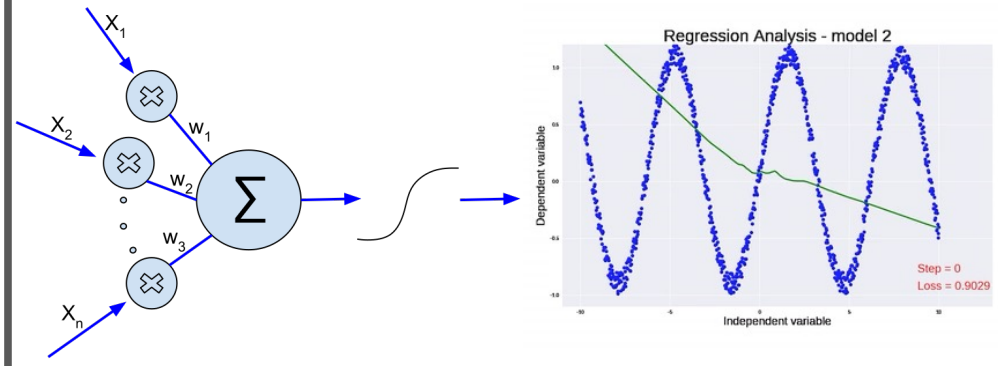
A simple perceptron solves the linear regression problem (linear in the free parameters, i.e. {w})



and activation function, a, is just

$$\hat{y} = a(\vec{w}^T \cdot \vec{x}) = \vec{w}^T \cdot \vec{x}$$

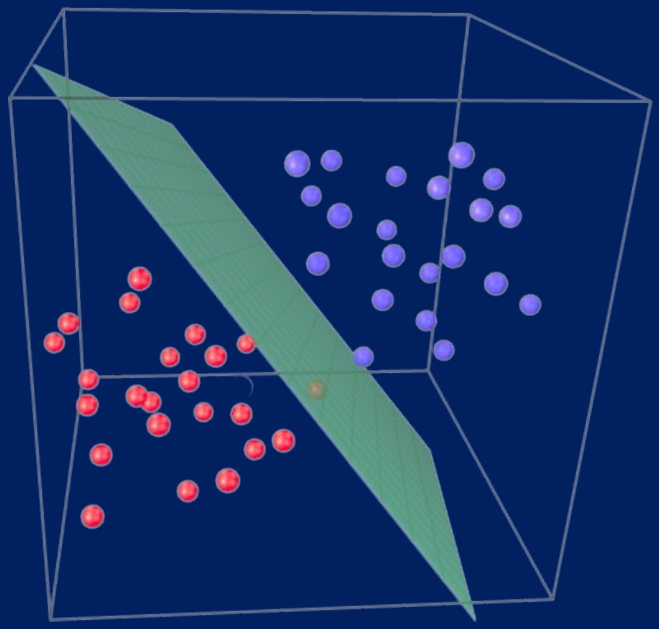
without a *non-linear* activation function, any deep NN would behave just like a simple perceptron (because summing over layer outputs would just give another linear function).



Hence we need to use a nonlinear activation to represent non-linear transformations of the weight parameters

$$\hat{y} = a(\vec{w}^T \cdot \vec{x}) = \tanh(\vec{w}^T \cdot \vec{x})$$

Classification as a Learning Task

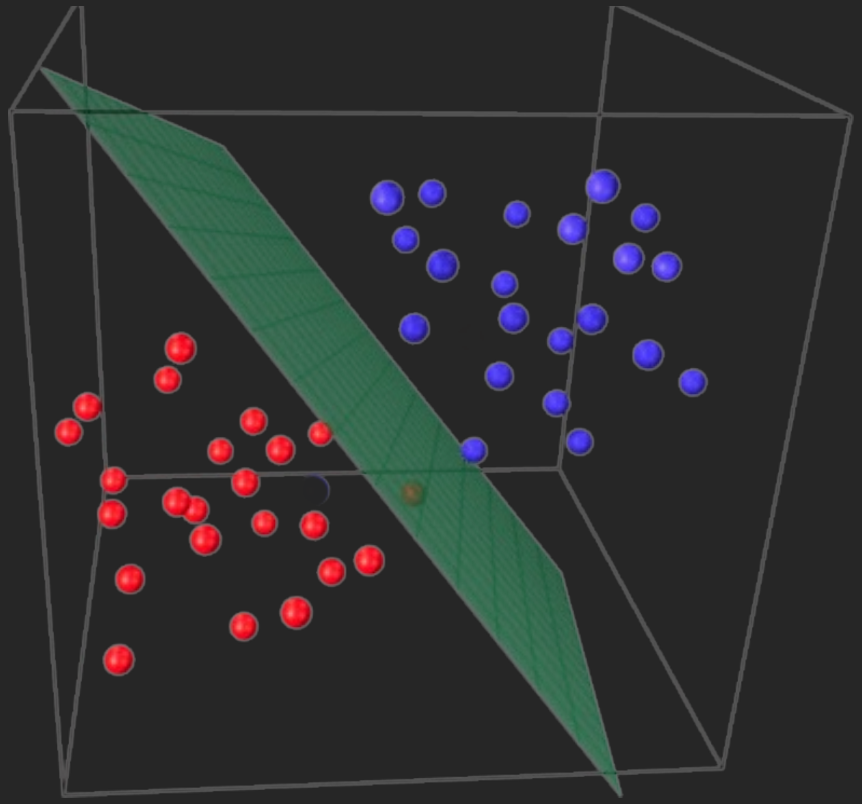


Example cases: Which A_xB_y compounds are chiral (or achiral)? Does patient improve or not under new drug, etc

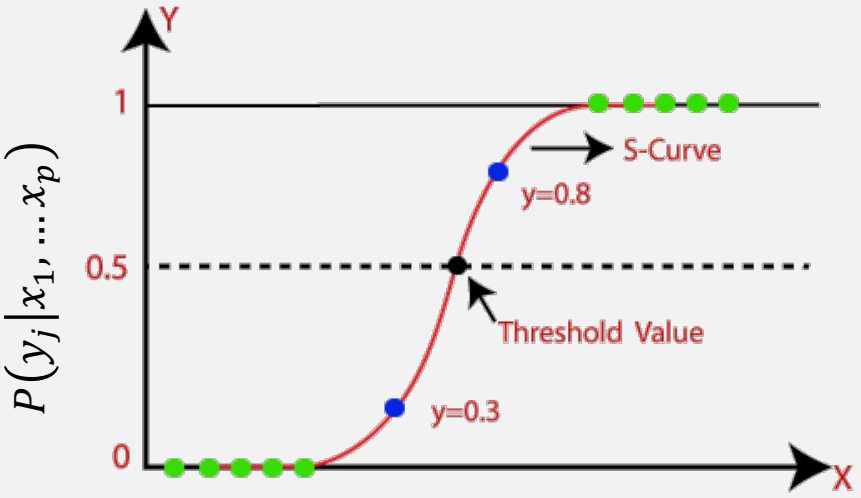
Purpose of Today's Lecture: Logistic Regression is a classification technique that gives not only the predicted classification, but the probability of it occurring.

- The goal of a classifier with p features $\{x_1,\ldots x_p\}$ and k classes $\{y_1,\ldots y_k\}$, is to determine the probability of the features occurring in each class, and to return the most likely class, y_j by calculating $P(y_j|x_1,\ldots x_p)$
- Classification is statistically modeled by a probability function, and the solution is solved numerically/self-consistently. You will show a simple perceptron can accomplish this same learning task in HW!
- (Naïve) Bayes is another statistical model that learns how to solve linear classification problems, but Naive Bayes figures out how the data was generated given the training results (an intermediate result). It uses Baye's theorem for the posterior probability.

Binomial Logistic Regression



Binomial Logistic Regression, is used to predict a binary decision, such as "Yes" or "No."

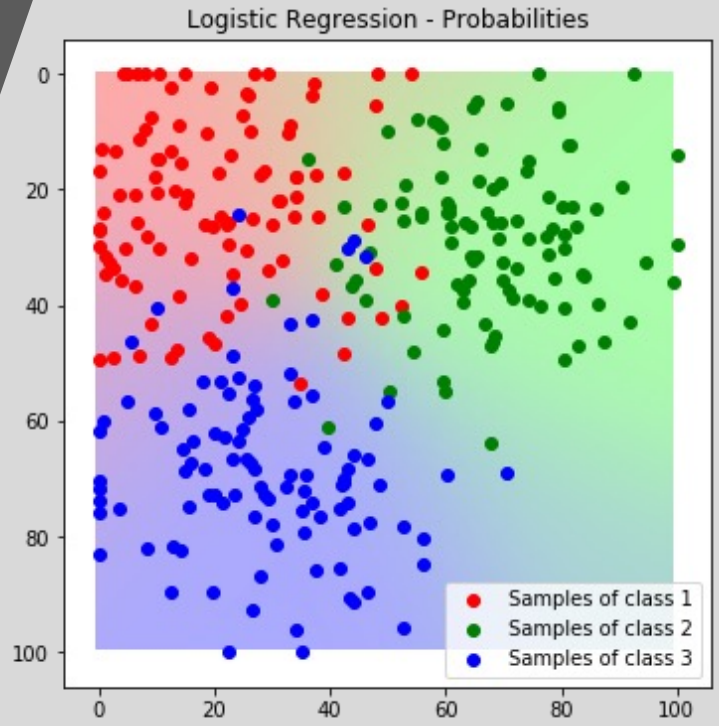


It is the statistical fitting of an “s-curve” logit function or probability to a dataset to estimate the occurrence of a binary category {y} given the values of a set of independent variables {x}. The fit is determined by varying the adjustable parameters {w} to maximize the log likelihood

Multinomial Logistic Regression

Multinomial Logistic Regression is classification for multiple categories to predict (not just two).

In this case it is the statistical fitting of a multinomial logit function to a dataset in order to calculate the probability of the occurrence of a multi-category dependent variable {y} given the independent data {x}. Again the fit is determined by varying the adjustable parameters {w} to maximize the log likelihood.



Example is the wine cultivar data set!

This now requires a different logit function

$$P(y_j|x_1, \ldots x_p) = \frac{1}{Z} exp[-\sum w_j x_j]$$

$$Z = \sum_i P_i$$

Calculated probabilities, P_j will be in the range of 0 to 1; the sum of all the probabilities is equals to 1

For a given pattern, μ, the conditional probability is assumed to be equal to a linear combination of the input variables $\{x_{j\mu}\}$ and unknown {w}'s

$$z_\mu = \sum_j^p w_j x_{j\mu}$$

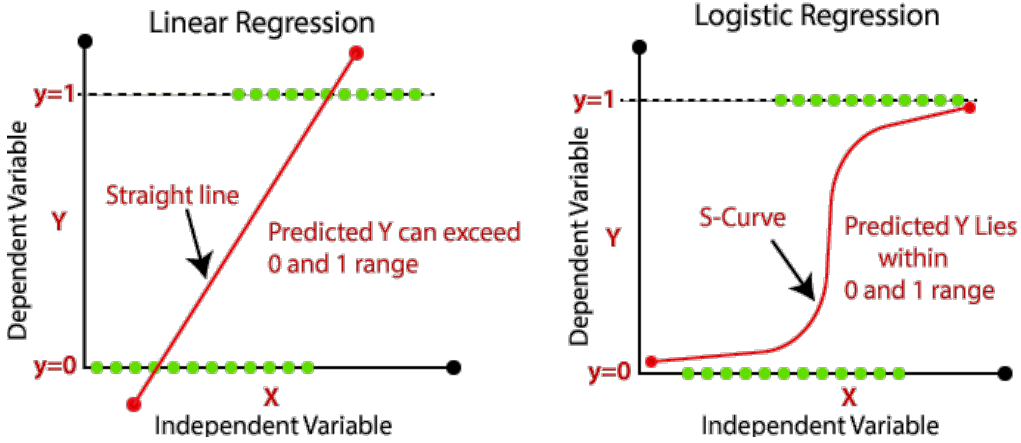
$$z_\mu = \mathrm{w}_0 + w_1 x_\mu \text{ (univariate)}$$

$$z_\mu = \mathrm{w}_0 + w_1 x_{1\mu} + w_2 x_{2\mu} \ \ldots + w_p x_{p\mu} \text{ (multivariate)}$$

To determine one of the two possible realizations of the calculated output variable $\widehat{y_\mu}$, we transform by a logistic function, or "s-curve" function.

$$\widehat{y_\mu} = \begin{cases} 1 \ if \quad z_\mu \geq 0.5 \\ 0 \ if \quad z_\mu < 0.5 \end{cases}$$

that changes range from –inf to inf to 0 to 1.



The latter is like a probability, and thus we can maximize log likelihood like we did in linear regression (which we assumed was Gaussian)

$$exp\left(-\frac{\left(y_\mu - b_0 \cdot 1 + b_1 x_{1\mu} + \cdots + b_p x_{p\mu}\right)^2}{2\sigma^2}\right)$$

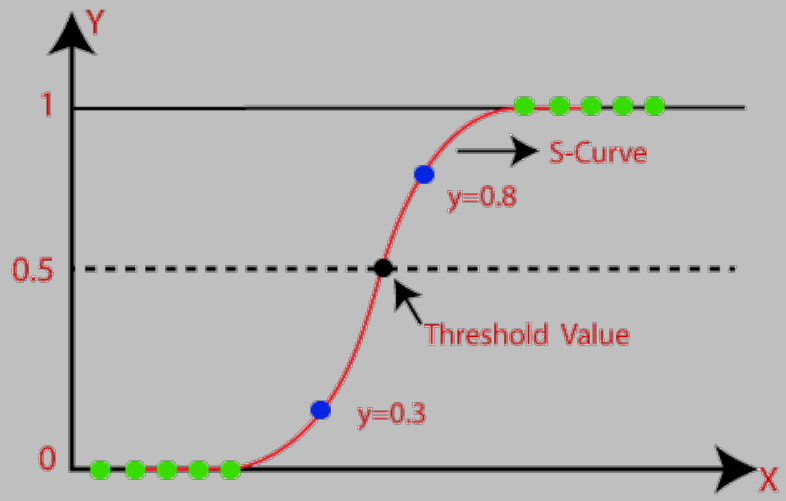
Binomial Logistic Regression

Binomial Logistic Regression

The logistic function (or s-curve) that satisfies this is the sigmoid function

$$y_\mu = \frac{1}{1 + exp(-\sum w_j x_{j\mu})}$$

why is this? We want to maximize the log likelihood of this new probability so use the log(odds) ratio



The log odds is defined to be linear in the parameters

$$\log(odds) = log\frac{P}{1-P} = \sum w_j x_{j\mu}$$

which we see when we solve for P

$$\frac{P}{1-P} = exp\left(\sum w_j x_{j\mu}\right) \quad \longrightarrow \quad P = exp\left(\sum w_j x_{j\mu}\right) - Pexp\left(\sum w_j x_{j\mu}\right)$$

$$\longrightarrow \quad P = \frac{exp(\sum w_j x_{j\mu})}{1 + exp(\sum w_j x_{j\mu})} \quad \longrightarrow \quad P = \frac{1}{1 + exp(-\sum w_j x_{j\mu})}$$

Binomial Logistic Regression

Because logistic regression predicts probabilities, rather than just classes, we can fit it using maximum likelihood, i.e. for each pattern or training point y_μ

(1) Define two Bernoulli probability distribution functions

$$P(y_\mu = 1|\sum w_j x_{j\mu}) = \frac{1}{1+exp(-\sum w_j x_{j\mu})} = S\left(\sum w_j x_{j\mu}\right)$$

$$P(y_\mu = 0|\sum w_j x_{j\mu}) = 1 - S\left(\sum w_j x_{j\mu}\right)$$

(2) multiplies them together to define a likelihood function over N examples for $y_\mu \in [0,1]$

$$\mathrm{L}(\{w\}|\mathrm{y},\mathrm{x}) = \prod_\mu^N \left[S\left(\sum w_j x_{j\mu}\right)\right]^{y_\mu} \left[1 - S\left(\sum w_j x_{j\mu}\right)\right]^{1-y_\mu}$$

I.e., if output variable y_μ of an example belongs to second class the likelihood function is

$$\left[S\left(\sum w_j x_{j\mu}\right)\right]^1 \left[1 - S\left(\sum w_j x_{j\mu}\right)\right]^0 = \frac{1}{1+exp(-\sum w_j x_{j\mu})}$$

The log-likelihood turns products into sums:

$$\log \mathrm{L}(\{w\}|\mathrm{y},\mathrm{x}) = \sum_{\mu}^{N} y_\mu \log S\left(\sum w_j x_{j\mu}\right) + (1-y_\mu)\log\left[1 - S\left(\sum w_j x_{j\mu}\right)\right]$$

To find the extremum we differentiate log L with respect to $\{w\}$, set the derivatives equal to zero, and solve.

This yields a transcendental equation, and there is no closed-form solution, so we must solve numerically using mathematical optimization (Newton-Raphson)

But with an appropriate choice of activation function

$$a(\vec{w}^T \cdot \vec{x}) = tanh(\beta(\vec{w}^T \cdot \vec{x}))$$

We can calculate it with our little perceptron to do logistic regression with the same back-propagation scheme we used for linear regression! You will do this in HW 4

Simple Perceptron is Logistic Regression too!