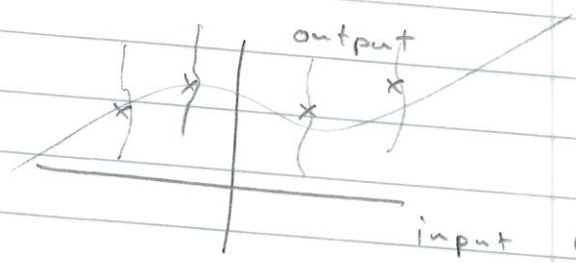
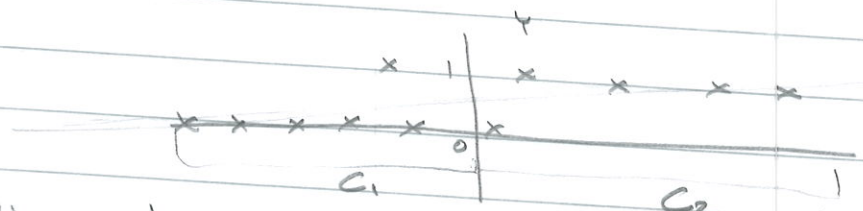
Classification ; In particular logistic reg

Classification and regression are close cousins. In probabilistic regression we model the conditional distribution of output given input (features).




In 2-class classification we do the same except the output is discrete rather than continuous, eg. $y_i = 1 \Rightarrow x_i \in C_1$. The goal is to learn a conditional distribution of the class label given the input.

Imagine, in 1-D input for now, that we have data like




it's clear that a linear regressor fit to such data will do something with a decision rule like $x_i \in C_1$ if $y_i < .5$ but such a rule a) doesn't allow a sharp decision boundary and b) doesn't have a coherent probabilistic interpretation: i.e, what does output = .7 mean?

What we might like is 1) a "noise" distribution that penalizes misclassification appropriately and 2) a controllable function that stretches input space.

The Bernoulli distribution is just the ticket for 1). For any given $y_i$ we can write

$$p(y_i|\dots) = p_i^{y_i}(1-p_i)^{1-y_i}$$

which if we know $p(y_i=1) = p_i$ then "getting it right" has likelihood $p_i$ and wrong $1-p_i$. If $p_i = 0.9$ and we observe $y_i = 1$ then we're better than if we observe $y_i = 0$. The only question is, what's $p_i$?

We only know that $p_i$ must be between 0 and 1; the choice is arbitrary past that.

For now, assume $x_i \in \mathbb{R}$. Let's consider

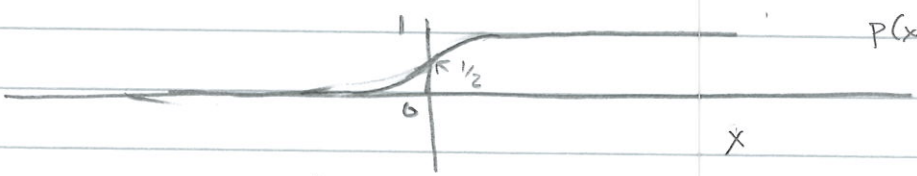
$$p_i = \frac{1}{1+\exp(-x_i\beta)}$$

the logistic sigmoid function. More generally

$$\sigma(x) = \frac{1}{1+\exp(-x)}$$

For large values of $x$, $\lim_{x\to\infty}\sigma(x) = 1$, and for small $\lim_{x\to-\infty}\sigma(x) = 0$.

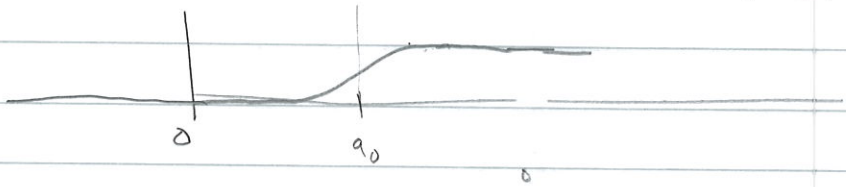
This function looks like



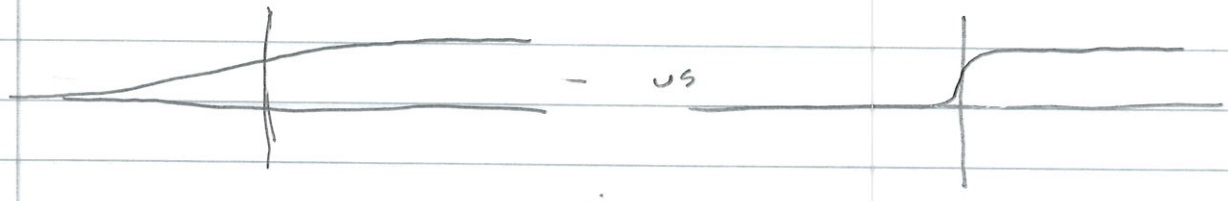

This is a non-linear function that stretches

One way shift this function by adding an offset

$$\sigma(x+a_0) = \frac{1}{1+\exp(-x-a_0)}$$




or make the slope more or less steep by adding a weight in front of x




via

$$\sigma(a_1 x + a_0) = \frac{1}{1+\exp(-a_1 x - a_0)}$$

The logistic sigmoid function has some nice properties which can be verified algebraically, for instance

$$1-\sigma(x) = \sigma(-x)$$

and

$$\frac{d\sigma}{dx} = \sigma(1-\sigma) .$$

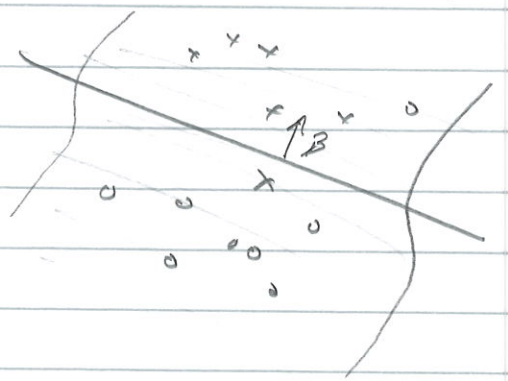
With these choices, (likelihood and nonlinear transform, we get logistic regression classification which imposes the following classification likelihood

$$p(\vec{y}=\{y_1 \cdots y_N\} \mid X=\{\vec{x}_1, \cdots \vec{x}_N\}; \vec{\beta}) = \prod_{n=1}^{N} \sigma(\vec{x}_n^T \vec{\beta})^{y_i} \left(1-\sigma(\vec{x}_n^T \vec{\beta})\right)^{1-y_i}$$

where $\vec{x}_i \in \mathbb{R}^{D+1}$, $\vec{\beta} \in \mathbb{R}^{D+1}$, $y_i = \{0,1\}$ and $\vec{x}_i = \begin{bmatrix} 1 \\ \vec{x}_i \end{bmatrix}$

like usual

Given a dataset and adding a 1 to the end of $\vec{x}$ as usual we learn




linear decision boundaries in that gracefully tolerate misclassifications

We can train logistic regression by maximum likelihood methods as per usual. You may verify yourself that

$$\frac{\partial}{\partial \vec{\beta}} \mathcal{L}(Y, X; \vec{\beta}) = \frac{\partial}{\partial \vec{\beta}} \log \prod_i \sigma(\vec{x}_i^T \vec{\beta})^{y_i} \sigma(-\vec{x}_i \vec{\beta})^{1-y_i}$$

$$= \sum_i y_i \left(1 - \sigma(\vec{x}_i^T \vec{\beta})\right) x_i^T - \sum_i (1-y_i)\left(1-\sigma(\vec{x}_i^T \vec{\beta})\right) x_i^T$$

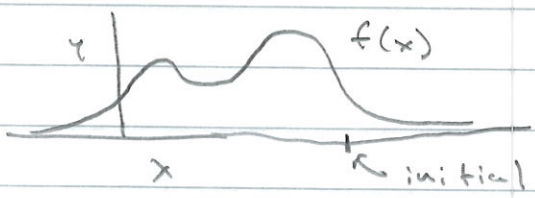
which, effectively, pushes and pulls on $\vec{\beta}$ depending on the sign induced by the value of $y_i$.

You may also verify that if you set this equal to $\vec{0}$ and attempt to solve you will not arrive at an analytic soln.

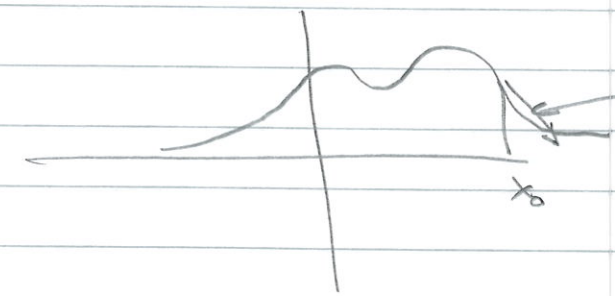
While there are fancier techniques for quickly ascending the likelihood it is worth definitively driving home gradient descent/ascent as a general optimization technique.

Gradient ascent is the simplest and in some ways the most important tool in the stats / M.L. arsenal.

In 1D let's say we have a function




$y = f(x)$ and a starting point $x_0$ and we would like to find the value $x_{max}$ that maximizes $f(x)$. If we are able to compute the derivative of $f$, $f' = \frac{df}{dx}$ then we can ascend $f$ by starting at $x_0$ and stepping in the direction opposite that of $f'(x_0)$




i.e. $\frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x}$ is negative

This suggests an iterative procedure for M.L. estimation that is powerfully general if not always efficient or perfect.

Function to maximize $f(x)$ given $f'$

$x = x_0$

until satisfied do

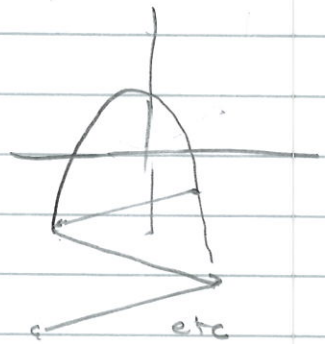
$x = x - \eta f'(x)$

return $x$

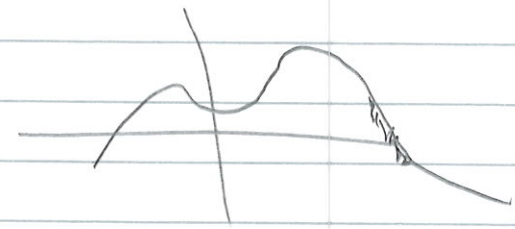
Some things to note:

1) If $f$ is multimodal this procedure may ascend to a single non-optimal mode.

2) $\eta$, the learning rate, is very important, too large and this procedure can diverge




too slow and it might take forever to converge



3) The exact argument holds in higher dim with gradients; more serious considerations apply -- see conjugate gradients, ADA grad, etc.

Note: when implementing gradient methods you should always check your gradient derivation & code using $f'(x) \approx \frac{f(x+\Delta x) - f(x)}{\Delta x}$ for, e.g. $\Delta x \approx 0.0001$

To conclude these 4(!?!) lectures let's consider our options for inference, estimation, and, in slightly more detail, classification.

We have considered ML estimation in which a point estimate of a parameter within a particular model family is found by maximizing a likelihood function. We have found that in $N$ $p$ setting ($N$ is "big $N$" and $p$ is little $p$ meaning lots of data a few parameters), and in the regression setting, we can estimate model parameters.

Further we can perform inference in two different ways; we can examine the sampling distribution of estimators <u>or</u> we can compute posterior dist's of parameters. The former is called <u>frequentist</u> inference, the latter <u>Bayesian</u>.

We also showed that regression and probabilistic approaches to classification are closely related (by example). It worth noting that there are widely divergent views on how to perform classification; they are, roughly

1) Model $p(x|C_k)$ and $p(C_k)$ $C_k$ = class
then assign class based on
$$p(C_k|x) = \frac{p(x|C_k)\,p(C_k)}{p(x)}$$

2) Model $p(C_k|x)$ directly ← log. reg.
3) Learn $f(x): X \to C$.