Linear Models for Classification

Kai-Lung Hua (花凱龍)

Data Representation

Our input data is a vector $x \in \mathbb{R}^D$ and the whole dataset is a matrix $X \in \mathbb{R}^{N \times D}$

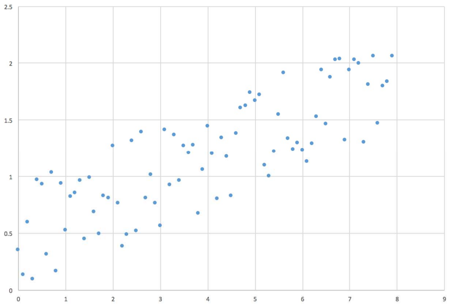
Different from linear regression, our target variable y for a classification task is a discrete scalar value.

Let us start with the simplest case where y can only take on two values (binary classification).

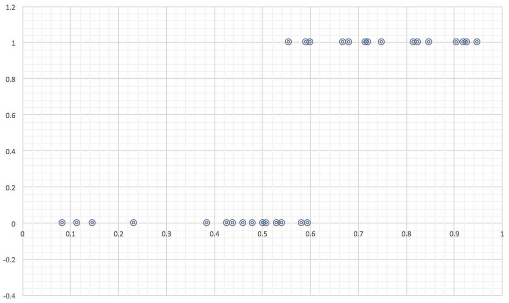
$$y \in \{0,1\}$$

What would this look like?

Regression ($y \in \mathbb{R}$)



Classification ($y \in \{0,1\}$)

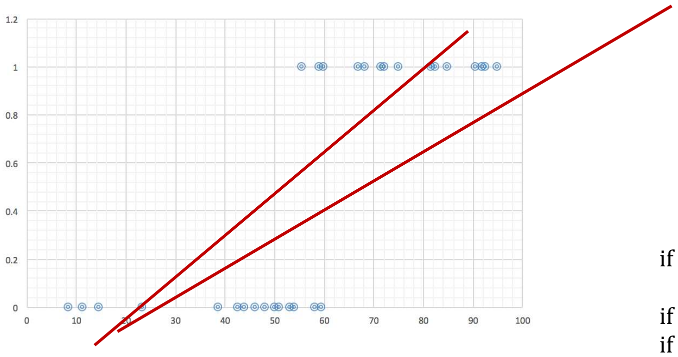


Mathematical Modeling

Our Hypothesis

What we know so far..

Assuming we use linear regression for a classification task, what will be the problem?



● ●

What if we had points further back

$y = \theta^T x + b$

if $x = 50$, $\hat{y}$ is around 0.28

if $x = 85$, $\hat{y}$ is around 0.71
if $x = 15$, $\hat{y}$ is around -0.1

Where do we cut off?

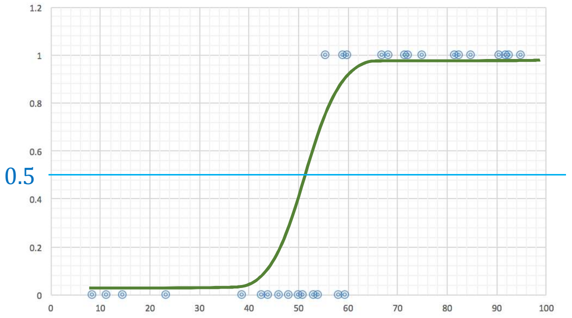
Think of the assumptions of Linear Regression, are they still valid?

- y can take on any real value ✘

- $y \sim N(\theta^T x, \sigma^2)$ ✘
  - Assumes mean of y is a linear combination of x ✘
  - y has constant conditional variance ✘

If many of the assumptions are violated, then you can't expect the model to have a good fit on your data

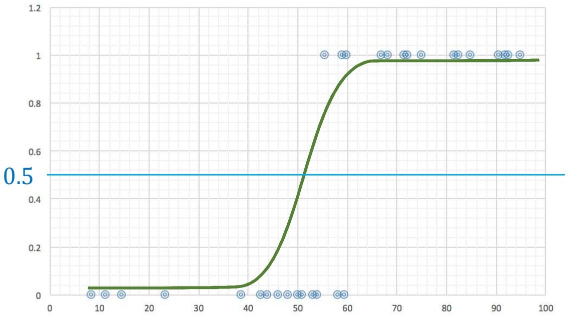
A more appropriate hypothesis function



Sigmoid / logistic function $\sigma(x) = \left(\frac{1}{1+e^{-x}}\right)$

*Note: Do not confuse the sigmoid function $\sigma(x)$, with the variance σ^2. They denote different things but unfortunately they have similar notations.

Some properties of the sigmoid (logistic) function



Sigmoid function $\sigma(x) = \left(\frac{1}{1+e^{-x}}\right)$

Domain

$$-\infty < x < \infty$$

Range

$$0 < \sigma(x) < 1$$

As $x \to \infty, \sigma(x) \to 1,$

$x \to -\infty, \sigma(x) \to 0$

*We will discuss where this function came from later

Logistic Regression

For logistic regression, we assume that the input x and the output y are related as follows:

$$y = h(x;\theta) = g(\theta^T x + b)$$
$$g(z) = \sigma(z) = \frac{1}{1 + e^{-z}}$$

where $h(x;\theta)$ is our hypothesis function, $\theta \in \mathbb{R}^D$ are the parameters (weights), $\sigma(z)$ is the sigmoid / logistic function.

*Note: In the assignment #4 we will be explicitly separating out the bias term to avoid regularizing it.

Logistic Regression

We assume that the output of our hypothesis function is a probability

$$P(y = 1|x) = h(x;\theta)$$
$$P(y = 0|x) = 1 - h(x;\theta)$$

We can interpret this output as the model's confidence in its prediction. This is useful specially in a task where mistakes are risky.

Linear regression

Logistic regression

Let $\text{score} = \theta^T x$

$$y = \text{score}$$

Returns an unbounded continuous value

$$y = \sigma(\text{score}) = \frac{1}{1 + e^{-scor}}$$

Returns a probability $P(y = 1|x)$

The sigmoid / logistic function σ converts raw scores to probabilities!

*The technical term for this is the response function. (more on this later)

Mathematical Modeling

The Objective

Recall: Bernoulli Distribution

A discrete random variable X that can only take on two values 1 ("success") and 0 ("fail") will follow a Bernoulli distribution, denoted as $X \sim \text{Bernoulli}(p)$, where p a parameter that represents the probability of success.

$$P(X=1)=p$$
$$P(X=0)=1-p$$

$$P(X=x)=\begin{cases} p & \text{if } x=1 \\ 1-p & \text{if } x=0 \end{cases}$$

$$P(X=x)=p^{x}(1-p)^{1-x}$$

if $x=1$ $\quad p^{1}(1-p)^{1-1}=p$

if $x=0$ $\quad p^{0}(1-p)^{1-0}=1-p$

Note:

$\mathbb{E}[X]=p$

$\text{Var}[X]=p(1-p)$

Objective function via Maximum Likelihood Estimation (MLE)

$$P(y_i=1|x_i)=h(x_i;\theta)$$
$$P(y_i=0|x_i)=1-h(x_i;\theta)$$
$$P(y_i|x_i)=\left(h(x_i;\theta)\right)^{y_i}\left(1-h(x_i;\theta)\right)^{1-y_i}$$

$$\text{likelihood } \mathcal{L}(\theta)=\prod_{i=1}^{N}P(y_i|x_i)=\prod_{i=1}^{N}\left(h(x_i;\theta)\right)^{y_i}\left(1-h(x_i;\theta)\right)^{1-y_i}$$

$$\underset{\theta}{\operatorname{argmax}} \log \mathcal{L}(\theta) = \boxed{\frac{1}{N}\sum_{i}^{N}\left[y_i\log(h(x_i;\theta))+(1-y_i)\log(1-h(x_i;\theta))\ \right]}$$

So the loss does not increase with the batch size

↑ Binary cross entropy

What is MLE doing?

y
1
1
1
1
0
0
0
0

$$\mathcal{L}(\theta) = \prod_{i=1}^{8} P(y_i|x;\theta)$$

$$\begin{aligned}\mathcal{L}(\theta) &= \textstyle\prod_{i=1}^{8} p^{y_i}(1-p)^{1-y_i} \\ &= (0.5)^4(1-0.5)^4 \\ &= 0.00390625\end{aligned}$$

$$\begin{aligned}\mathcal{L}(\theta) &= \textstyle\prod_{i=1}^{8} p^{y_i}(1-p)^{1-y_i} \\ &= (0.8)^4(1-0.8)^4 \\ &= 0.00065536\end{aligned}$$

What is MLE doing?

y
1
1
1
1
0
1
0
1

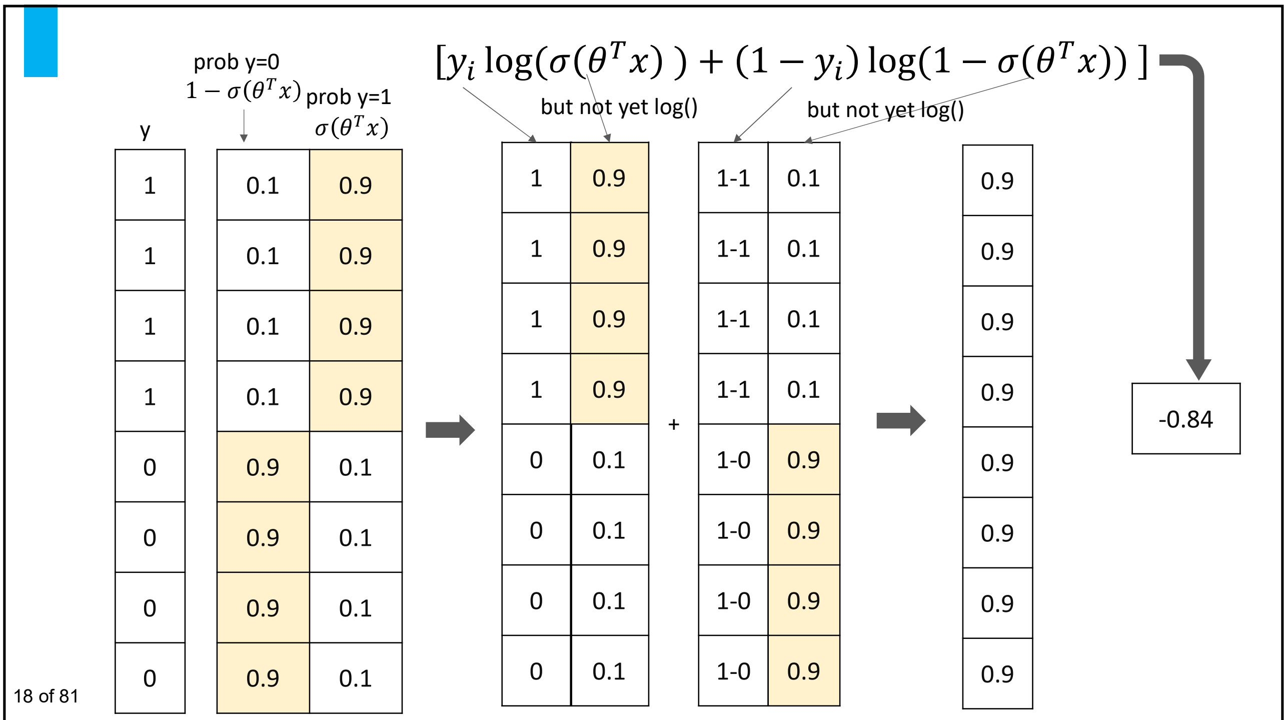
$$\mathcal{L}(\theta) = \prod_{i=1}^{8} P(y_i|x;\theta)$$

$$\begin{aligned}\mathcal{L}(\theta) &= \textstyle\prod_{i=1}^{8} p^{y_i}(1-p)^{1-y_i} \\ &= (0.5)^6(1-0.5)^2 \\ &= 0.00390625\end{aligned}$$

$$\begin{aligned}\mathcal{L}(\theta) &= \textstyle\prod_{i=1}^{8} p^{y_i}(1-p)^{1-y_i} \\ &= (0.8)^6(1-0.8)^2 \\ &= 0.01048576\end{aligned}$$

Log Likelihood Example

17 of 81



$[y_i \log(\sigma(\theta^T x)\ ) + (1 - y_i) \log(1 - \sigma(\theta^T x))\ ]$

y	prob y=0 $1 - \sigma(\theta^T x)$	prob y=1 $\sigma(\theta^T x)$
1	0.1	0.9
1	0.1	0.9
1	0.1	0.9
1	0.1	0.9
0	0.9	0.1
0	0.9	0.1
0	0.9	0.1
0	0.9	0.1

but not yet log()

1	0.9
1	0.9
1	0.9
1	0.9
0	0.1
0	0.1
0	0.1
0	0.1

+

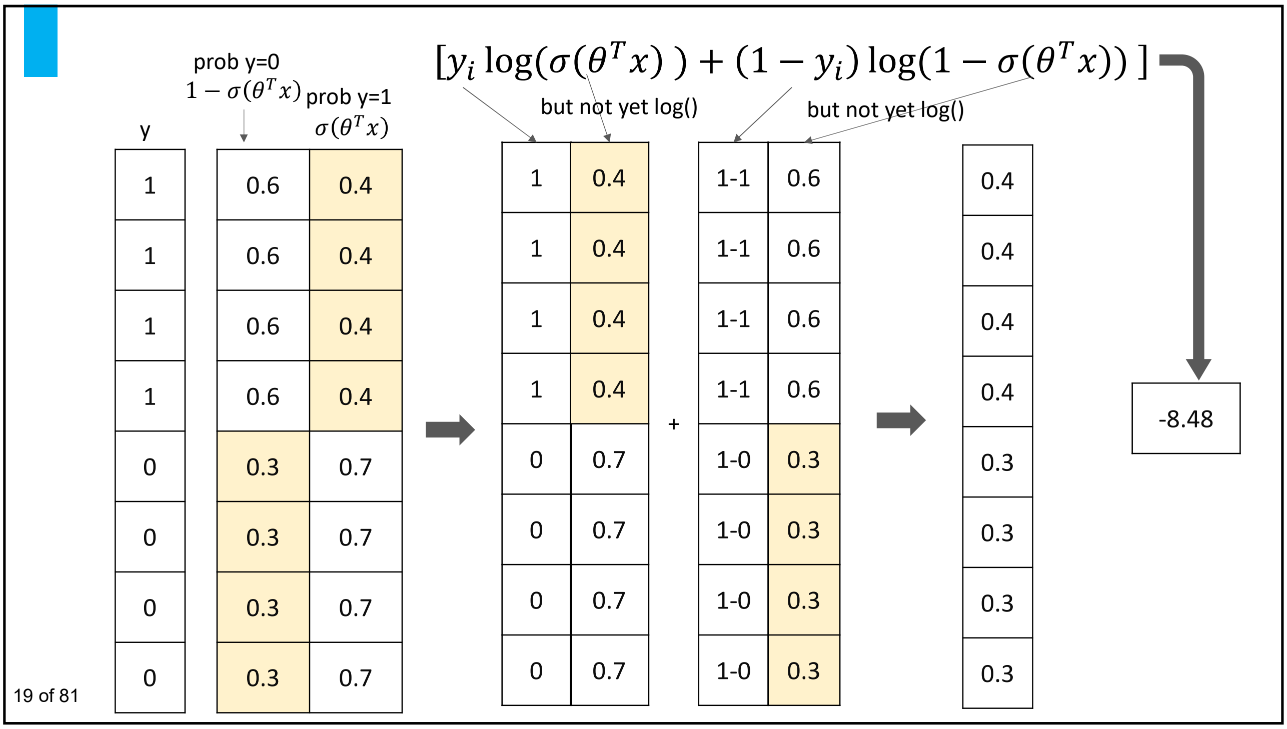
but not yet log()

1-1	0.1
1-1	0.1
1-1	0.1
1-1	0.1
1-0	0.9
1-0	0.9
1-0	0.9
1-0	0.9

0.9
0.9
0.9
0.9
0.9
0.9
0.9
0.9

-0.84

18 of 81



$$[y_i \log(\sigma(\theta^T x)\ ) + (1 - y_i) \log(1 - \sigma(\theta^T x))\ ]$$

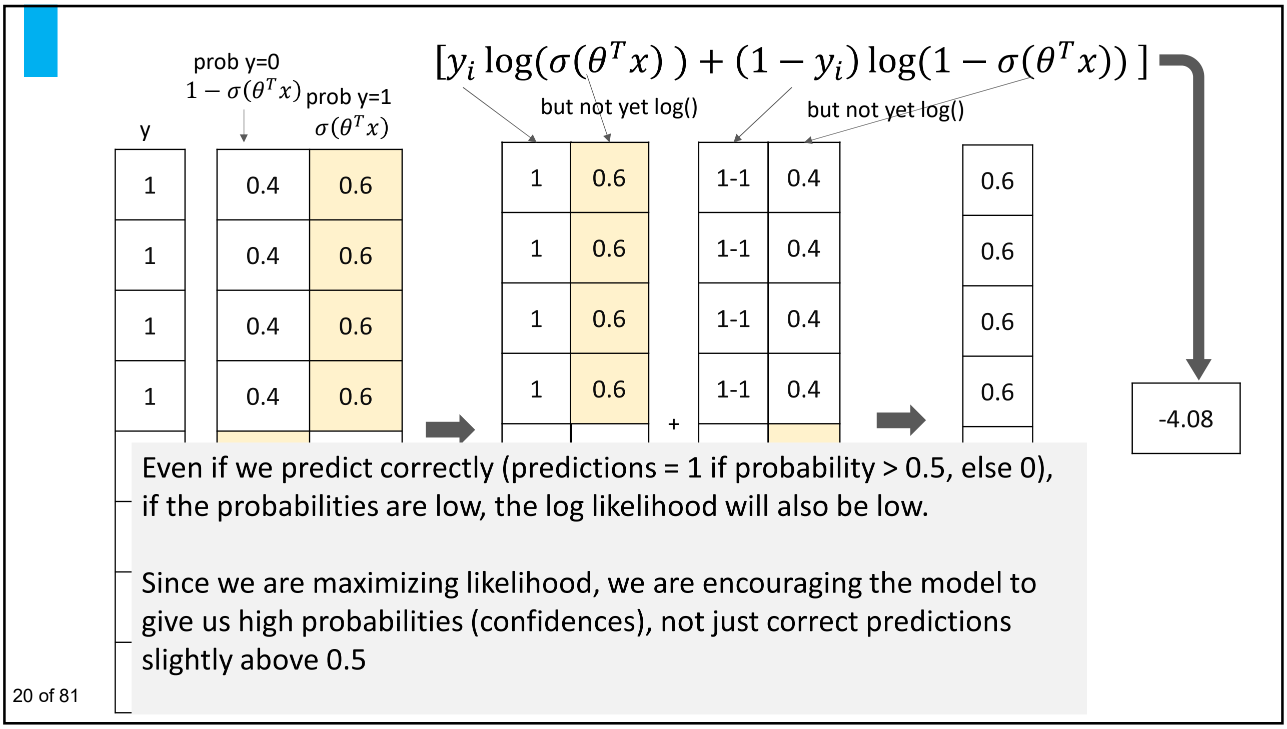
y	prob y=0 $1-\sigma(\theta^T x)$	prob y=1 $\sigma(\theta^T x)$
1	0.6	0.4
1	0.6	0.4
1	0.6	0.4
1	0.6	0.4
0	0.3	0.7
0	0.3	0.7
0	0.3	0.7
0	0.3	0.7

	but not yet log()			but not yet log()
1	0.4		1-1	0.6
1	0.4		1-1	0.6
1	0.4		1-1	0.6
1	0.4	+	1-1	0.6
0	0.7		1-0	0.3
0	0.7		1-0	0.3
0	0.7		1-0	0.3
0	0.7		1-0	0.3

0.4
0.4
0.4
0.4
0.3
0.3
0.3
0.3

-8.48

19 of 81



$$[y_i \log(\sigma(\theta^T x)\ ) + (1 - y_i) \log(1 - \sigma(\theta^T x))\ ]$$

y	prob y=0 $1-\sigma(\theta^T x)$	prob y=1 $\sigma(\theta^T x)$
1	0.4	0.6
1	0.4	0.6
1	0.4	0.6
1	0.4	0.6

	but not yet log()			but not yet log()
1	0.6		1-1	0.4
1	0.6		1-1	0.4
1	0.6		1-1	0.4
1	0.6	+	1-1	0.4

0.6
0.6
0.6
0.6

-4.08

Even if we predict correctly (predictions = 1 if probability > 0.5, else 0), if the probabilities are low, the log likelihood will also be low.

Since we are maximizing likelihood, we are encouraging the model to give us high probabilities (confidences), not just correct predictions slightly above 0.5

20 of 81

Optimization

Learning the optimal parameters for our model

Gradient Descent

$$\underset{\theta}{\operatorname{argmax}}\,\mathcal{L}(\theta) = \underset{\theta}{\operatorname{argmin}} -\log\mathcal{L}(\theta) = \; -\sum_{i}^{N}[y_i \log(\sigma(\theta^T x)\,) + (1-y_i)\log(1-\sigma(\theta^T x))\,]$$

↑ Negative log likelihood

$$L(\theta) = -\sum_{i}^{N}[y_i \log(\sigma(\theta^T x)\,) + (1-y_i)\log(1-\sigma(\theta^T x))\,]$$

Loss Function

$$\theta \coloneqq \theta - \alpha\frac{\partial L}{\partial\theta}$$

Weight update (via gradient descent)

Newton-Raphson Method

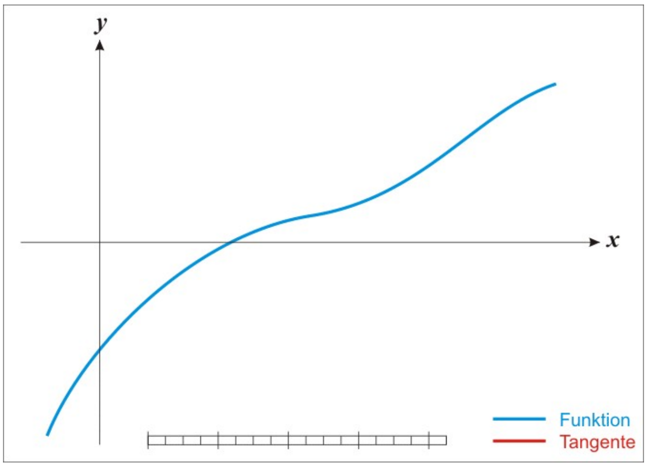
A very fast numerical method to solve for roots (zeroes) of a real-valued equation.

Solve for x such that $f(x) = 0$.

Given an initial guess x_0. We iteratively update our guess using the following equation until convergence.

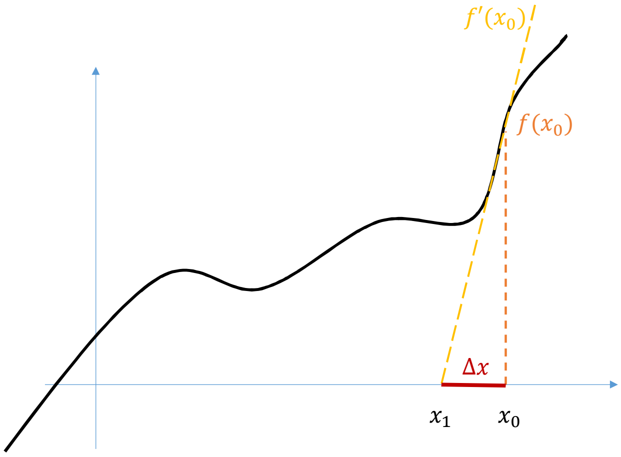
$$x_{t+1} = x_t - \frac{f(x_t)}{f'(x_t)}$$

Newton-Raphson Method



Source: Wikipedia

Derivation



$$\Delta x = x_0 - x_1$$

$$f'(x_0) = \frac{f(x_0)}{\Delta x}$$

$$\Delta x = \frac{f(x_0)}{f'(x_0)}$$

$$x_0 - x_1 = \frac{f(x_0)}{f'(x_0)}$$

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

$$x_{t+1} = x_t - \frac{f(x_t)}{f'(x_t)}$$

Newton-Raphson Method

But if it's a root-finding algorithm, how do we use it for optimization?

Recall: to get the minima / maxima of a function we get its derivative and set it to 0, ($f'(x) = 0$).

So we are finding the roots of the first derivative.

Newton-Raphson Method

So in our case, to use it for optimization, we just set

$$f(\theta) = \frac{\partial L}{\partial \theta} = 0$$

Following the update formula we get

$$\theta_{t+1} = \theta_t - \frac{f(\theta)}{f'(\theta)} = \theta_t - \frac{\frac{\partial L}{\partial \theta}}{\frac{\partial^2 L}{\partial \theta^2}}$$

But θ is a vector.

A very short crash course on matrix calculus

Recall: Gradients

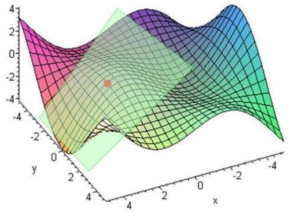
- Mathematical definition of a derivative

$$\frac{\partial f(x)}{\partial x} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

- You can think of gradients as derivatives in higher dimensions. Let $\boldsymbol{x}$ be a vector:

$$\nabla_{\boldsymbol{x}} f(x) = \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x_1}} \\ \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x_2}} \\ \vdots \\ \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x_n}} \end{bmatrix}$$

- The gradient defines (hyper) plane approximating the function infinitesimally



$$\Delta z = \frac{\partial f}{\partial x} \cdot \Delta x + \frac{\partial f}{\partial y} \cdot \Delta y$$

29 of 81

Gradients

- Gradients are a natural extension of partial derivatives to functions of multiple variables.
- If $f: \mathbb{R}^{m \times n} \to \mathbb{R}$
- The gradient of f with respect to $A \in \mathbb{R}^{m \times n}$ is the matrix of partial derivatives defined as:

$$\nabla_A f(A) \in \mathbb{R}^{m \times n} = \begin{bmatrix} \frac{\partial f(A)}{\partial A_{11}} & \frac{\partial f(A)}{\partial A_{12}} & \cdots & \frac{\partial f(A)}{\partial A_{1n}} \\ \frac{\partial f(A)}{\partial A_{21}} & \frac{\partial f(A)}{\partial A_{22}} & \cdots & \frac{\partial f(A)}{\partial A_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f(A)}{\partial A_{m1}} & \frac{\partial f(A)}{\partial A_{m2}} & \cdots & \frac{\partial f(A)}{\partial A_{mn}} \end{bmatrix}$$

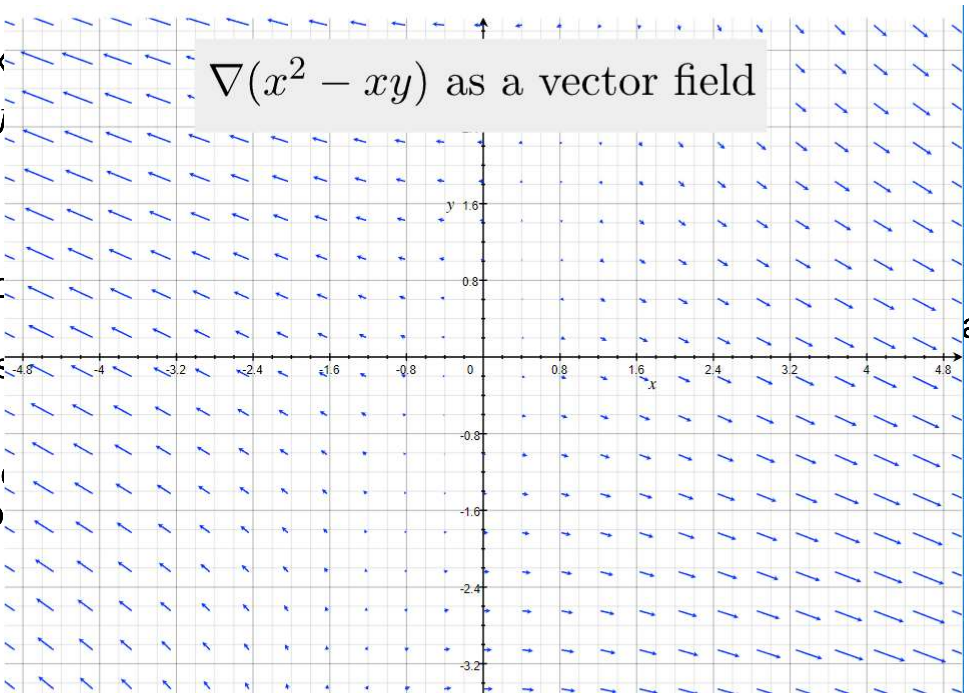
i.e., an $m \times n$ matrix with

$$(\nabla_A f(A))_{ij} = \frac{\partial f(A)}{\partial A_{ij}}.$$

30 of 81

- The size of $\nabla_A f(A)$ always have the same size/dimensions of A
- so if A is just a vector $x \in \mathbb{R}^n$ then $\nabla_x f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix}$
- The gradient is only defined if the function is real-valued and returns a scalar value.
- $\nabla_x\big(f(x) + g(x)\big) = \nabla_x f(x) + \nabla_x g(x)$
- For $c \in \mathbb{R}$, $\nabla_x\big(cf(x)\big) = c\nabla_x f(x)$
- The vector ∇f tells you which direction has the most rapid / steepest increase in the value of f.
- The gradient vectors are perpendicular to the contour lines of f.

- Ex
- If j
  -
- Nc so in are jus
- W sp


$\nabla(x^2 - xy)$ as a vector field

Hessian

- We can think of gradient as the first derivative for functions of vectors and the Hessian is the second derivative.
- $f: \mathbb{R}^n \rightarrow \mathbb{R}$

$$\nabla_x^2 f(x) \in \mathbb{R}^{n\times n} = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{bmatrix}.$$

$$(\nabla_x^2 f(x))_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}.$$

- Often also denoted as $\boldsymbol{H_f}$.
- Similar to gradients it only makes sense for scalar-valued functions.

Example

- $f(x,y) = x^3 - 2xy - y^6$
- $\frac{\partial f}{\partial x} = 3x^2 - 2y$
- $\frac{\partial f}{\partial y} = -2x - 6y^5$

- $\frac{\partial^2 f}{\partial x^2} = 6x$
- $\frac{\partial^2 f}{\partial x \partial y} = -2$
- $\frac{\partial^2 f}{\partial y \partial x} = -2$
- $\frac{\partial^2 f}{\partial y^2} = -30y^4$

$$\nabla f = \begin{bmatrix} 3x^2 - 2y \\ -2x - 6y^5 \end{bmatrix}$$

$$\nabla^2 f = H_f = \begin{bmatrix} 6x & -2 \\ -2 & -30y^4 \end{bmatrix}$$

Going back to Newton-Raphson Method

Going Back to Newton-Raphson Method

Single dimensional setting

$$\theta_{t+1} = \theta_t - \frac{\frac{\partial L}{\partial \theta}}{\frac{\partial^2 L}{\partial \theta^2}}$$

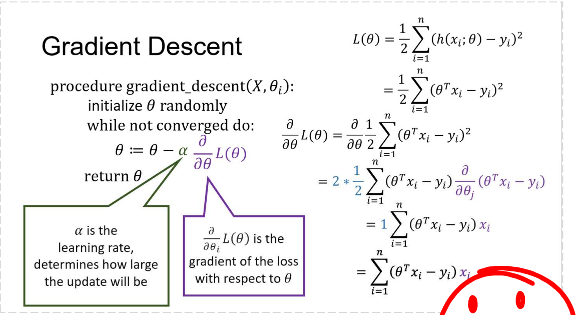
Generalization to Multidimensional setting, where H is the hessian of L and $\nabla_\theta L$ is the gradient of L.

$$\theta_{t+1} = \theta_t - H^{-1} \nabla_\theta L$$

Going Back to Newton-Raphson Method

$$\theta_{t+1} = \theta_t - H^{-1}\nabla_\theta L$$

Doesn’t this look familiar?



Gradient Descent

```
procedure gradient_descent($X, \theta_i$):
    initialize $\theta$ randomly
    while not converged do:
        $\theta := \theta - \alpha \frac{\partial}{\partial\theta} L(\theta)$
    return $\theta$
```

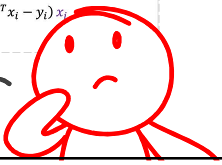
α is the learning rate, determines how large the update will be

$\frac{\partial}{\partial\theta_i} L(\theta)$ is the gradient of the loss with respect to θ

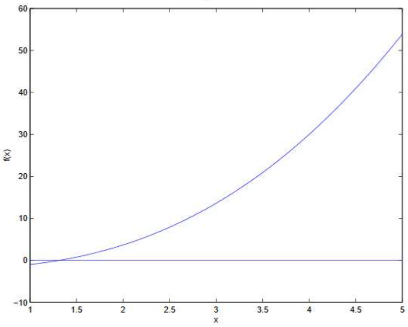
$$L(\theta) = \frac{1}{2}\sum_{i=1}^{n}(h(x_i;\theta) - y_i)^2$$
$$= \frac{1}{2}\sum_{i=1}^{n}(\theta^T x_i - y_i)^2$$
$$\frac{\partial}{\partial\theta} L(\theta) = \frac{\partial}{\partial\theta}\frac{1}{2}\sum_{i=1}^{n}(\theta^T x_i - y_i)^2$$
$$= 2 * \frac{1}{2}\sum_{i=1}^{n}(\theta^T x_i - y_i)\frac{\partial}{\partial\theta_j}(\theta^T x_i - y_i)$$
$$= 1\sum_{i=1}^{n}(\theta^T x_i - y_i)\, x_i$$
$$= \sum_{i=1}^{n}(\theta^T x_i - y_i)\, x_i$$

Instead of a fixed learning rate α, we have a dynamic H^{-1} that computes for the most suitable step size!

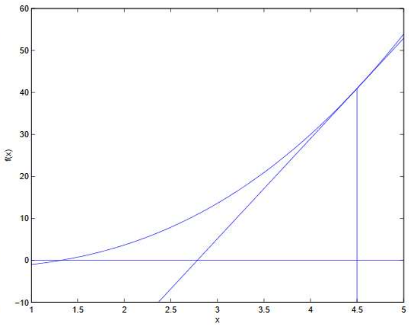
like gradient descent?



Example Newton-Raphson Method

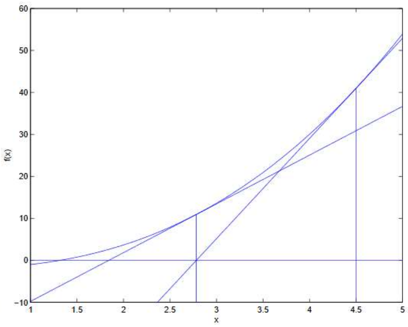


function(x) and line y = 0
We’re trying to find: x
Such that f(x) = 0



Step 0

1. Initialize x = 4.5
2. Fit tangent line at x= 4.5
3. Solve for where line evaluates at 0 (which is 2.8)
4. Update x = 2.8



Step 1

1. Fit tangent line at x= 2.8
2. Solve for where line evaluates at 0 (which is 1.8)
3. Update x= 1.8

Disadvantages of Newton's Method

- No convergence guarantees if the initial guess x_0 is far from the solution.
- No convergence guarantees, if tangent line becomes parallel or almost parallel to the x-axis.
- Hessian H must exist and must be invertible
- Both the Hessian H and the inverse operation are expensive to compute.

Advantages of Newton's Method

- Very simple to implement
- If it does converge, it converges very fast (quadratic)

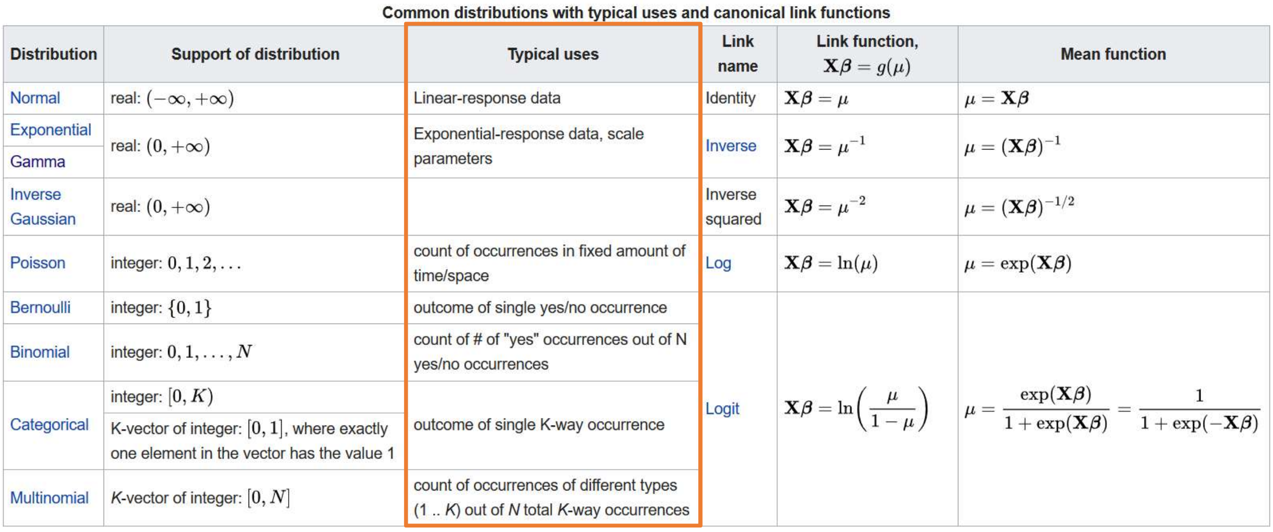
A deeper look into logistic regression

Generalized Linear Models

Linear regression and Logistic regression are actually a part of a bigger family of Linear Models called Generalized Linear Models.

We can extend the ideas of Linear Regression and Logistic Regression beyond the Gaussian and Bernoulli setting to a more general exponential family.

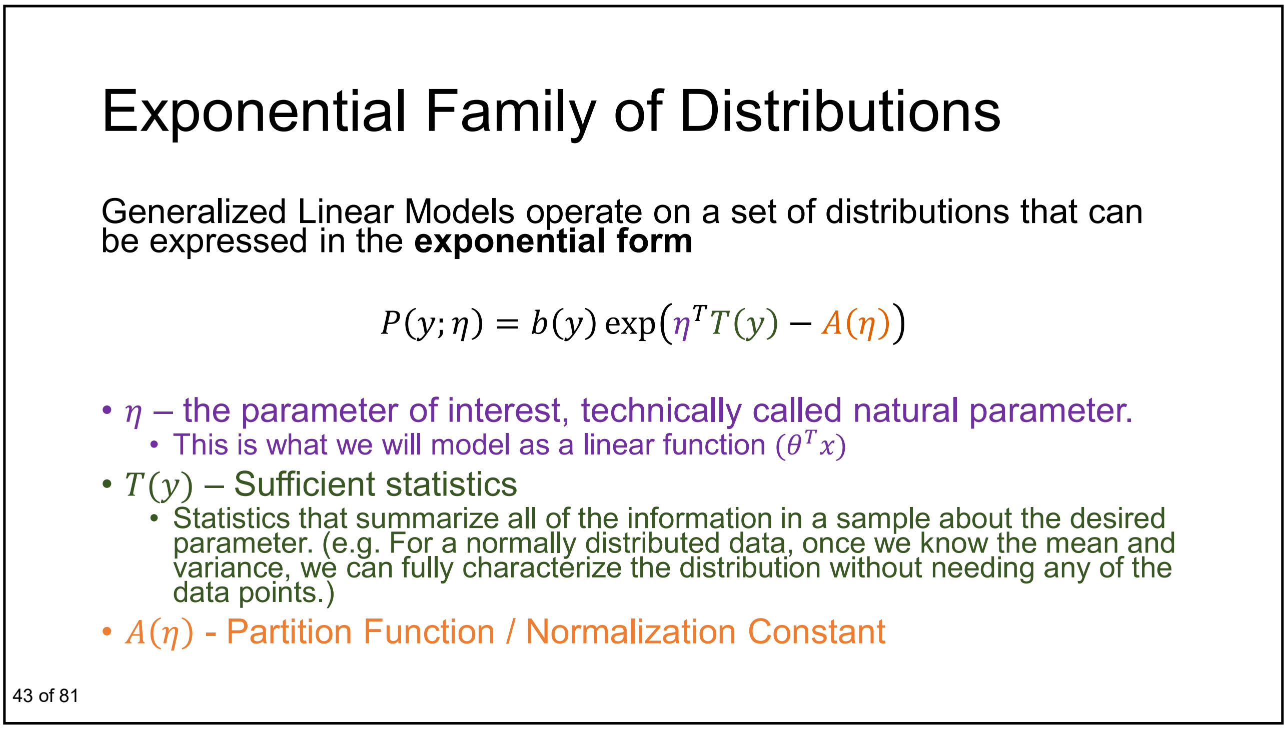
E.g. Poisson distribution for count data



Common distributions with typical uses and canonical link functions

Distribution	Support of distribution	Typical uses	Link name	Link function, $\mathbf{X}\boldsymbol{\beta} = g(\mu)$	Mean function
Normal	real: $(-\infty, +\infty)$	Linear-response data	Identity	$\mathbf{X}\boldsymbol{\beta} = \mu$	$\mu = \mathbf{X}\boldsymbol{\beta}$
Exponential Gamma	real: $(0, +\infty)$	Exponential-response data, scale parameters	Inverse	$\mathbf{X}\boldsymbol{\beta} = \mu^{-1}$	$\mu = (\mathbf{X}\boldsymbol{\beta})^{-1}$
Inverse Gaussian	real: $(0, +\infty)$		Inverse squared	$\mathbf{X}\boldsymbol{\beta} = \mu^{-2}$	$\mu = (\mathbf{X}\boldsymbol{\beta})^{-1/2}$
Poisson	integer: $0, 1, 2, \ldots$	count of occurrences in fixed amount of time/space	Log	$\mathbf{X}\boldsymbol{\beta} = \ln(\mu)$	$\mu = \exp(\mathbf{X}\boldsymbol{\beta})$
Bernoulli	integer: $\{0, 1\}$	outcome of single yes/no occurrence	Logit	$\mathbf{X}\boldsymbol{\beta} = \ln\left(\frac{\mu}{1-\mu}\right)$	$\mu = \frac{\exp(\mathbf{X}\boldsymbol{\beta})}{1+\exp(\mathbf{X}\boldsymbol{\beta})} = \frac{1}{1+\exp(-\mathbf{X}\boldsymbol{\beta})}$
Binomial	integer: $0, 1, \ldots, N$	count of # of "yes" occurrences out of N yes/no occurrences			
Categorical	integer: $[0, K)$ *K*-vector of integer: $[0, 1]$, where exactly one element in the vector has the value 1	outcome of single K-way occurrence			
Multinomial	*K*-vector of integer: $[0, N]$	count of occurrences of different types (1 .. *K*) out of *N* total *K*-way occurrences			

Source: Wikipedia



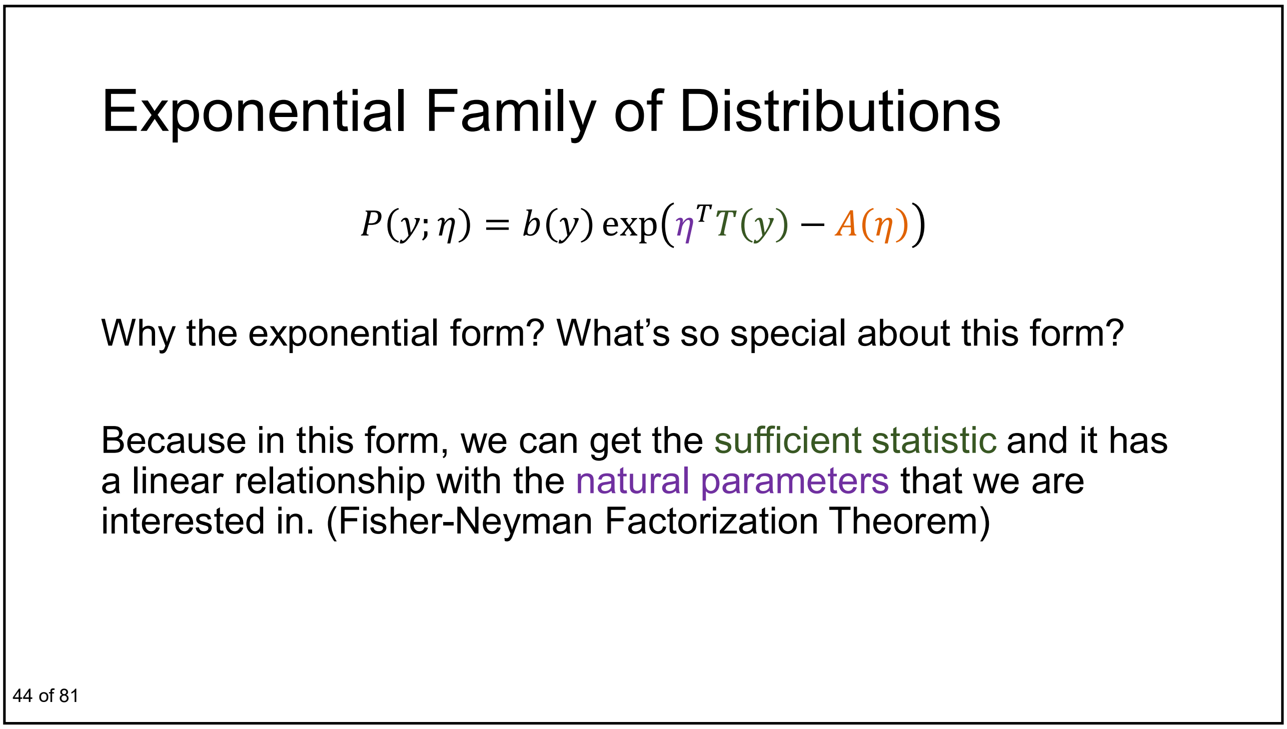
Exponential Family of Distributions

Generalized Linear Models operate on a set of distributions that can be expressed in the **exponential form**

$$P(y;\eta) = b(y)\exp\left(\eta^T T(y) - A(\eta)\right)$$

- η – the parameter of interest, technically called natural parameter.
  - This is what we will model as a linear function $(\theta^T x)$
- $T(y)$ – Sufficient statistics
  - Statistics that summarize all of the information in a sample about the desired parameter. (e.g. For a normally distributed data, once we know the mean and variance, we can fully characterize the distribution without needing any of the data points.)
- $A(\eta)$ - Partition Function / Normalization Constant

43 of 81



Exponential Family of Distributions

$$P(y;\eta) = b(y)\exp\left(\eta^T T(y) - A(\eta)\right)$$

Why the exponential form? What's so special about this form?

Because in this form, we can get the sufficient statistic and it has a linear relationship with the natural parameters that we are interested in. (Fisher-Neyman Factorization Theorem)

44 of 81

Exponential Family of Distributions

Some Members of the Exponential Family of Distributions:

- Gaussian Distribution
- Bernoulli / Binomial
- Multinomial
- Gamma
- Poisson
- Beta
- Dirichlet

Bernoulli Distribution

$$P(y;\eta) = b(y)\exp\left(\eta^T T(y) - A(\eta)\right)$$

Let $p = P(y = 1|x)$

$$\begin{aligned}
P(y;p) &= p^y(1-p)^{1-y} \\
&= \exp\left(\log(p^y(1-p)^{1-y})\right) \\
&= \exp(y\log(p) \ + (1-y)\log(1-p)) \\
&= \exp(y\log(p) + \log(1-p) - y\log(1-p)) \\
&= \exp(y\log\left(\tfrac{p}{1-p}\right) + \log(1-p)) \\
&= \exp(\log\left(\tfrac{p}{1-p}\right) y + \log(1-p))
\end{aligned}$$

$$\exp(\log\left(\frac{p}{1-p}\right)y + \log(1-p))$$

Bernoulli Distribution

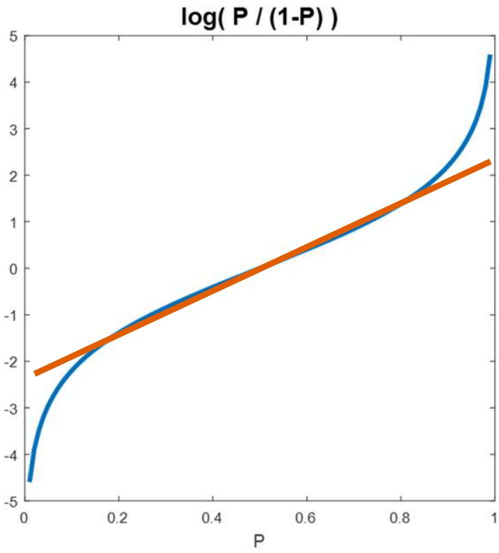
Natural parameter

$$\eta = \log\left(\frac{p}{1-p}\right)$$

What does this look like?

Can be approximated well by a linear function*

$$\eta = \log\left(\frac{p}{1-p}\right) = \theta^T x$$



*Except for the tail ends where assumptions are violated. That’s where extreme value models come in.

Bernoulli Distribution

$$\eta = \log\left(\frac{p}{1-p}\right) = \theta^T x$$
$$\frac{p}{1-p} = e^{\theta^T x}$$
$$p = (e^{\theta^T x})(1-p)$$
$$p = e^{\theta^T x} - pe^{\theta^T x}$$
$$p + pe^{\theta^T x} = e^{\theta^T x}$$
$$p\left(1 + e^{\theta^T x}\right) = e^{\theta^T x}$$
$$p = P(y=1|x) = \boxed{\frac{e^{\theta^T X}}{1+e^{\theta^T X}} = \frac{1}{1+e^{-\theta^T x}} = \sigma(\theta^T x)}$$

↑ Sigmoid / Logistic Function

So this is where the sigmoid function comes from.

Generalized Linear Model Framework

In general, consider a classification or regression problem where we would like to predict the value of some random variable y as a function of x, $P(y|x)$.

Given x and θ, assume that the distribution of y follows some exponential family distribution with parameter η.

$$P(y|x;\theta) \sim \text{ExponentialFamily}(\eta)$$

Goal is to predict $\mathbb{E}[T(y)|x]$. Since y is probabilistic / random, we predict the mean of the possible values / distribution of y.

Generalized Linear Model Framework

By default y would be a sufficient statistic since it is our actual data ($T(y) = y$), which implies that we would like our prediction outputted by our learned hypothesis to be

$$h(x;\theta) = \mathbb{E}[T(y)|x] = \mathbb{E}[y|x]$$

For example in Logistic Regression

$$h(x;\theta) = \mathbb{E}[y|x] = 0 * P(y=0|x;\theta) + 1 * P(y=1|x;\theta)$$

$$h(x;\theta) = P(y=1|x;\theta)$$

Generalized Linear Model Framework

Assume that the natural parameter η and the inputs x are related linearly

$$\eta = \theta^T x$$

This is a design choice that makes it easier to derive GLMs that can accommodate different types of distributions.

For example in the Bernoulli distribution (refer to Slide 46):

$$\eta = \log\left(\frac{p}{1-p}\right)$$

GLM Framework

$$h(x;\theta) = \mathbb{E}[y|x] = 0 * P(y=0|x;\theta) + 1 * P(y=1|x;\theta)$$
$$h(x;\theta) = p = P(y=1|x;\theta)$$

In the GLM framework, the function g is the (canonical) response function that maps your natural parameters to what we are interested in predicting (expected value of the sufficient statistic).

$$g(\eta) = \mathbb{E}[T(y)|x]$$

In the case of the Bernoulli Distribution

$$\mathbb{E}[T(y)|x] = \mathbb{E}[y|x] = p = P(y=1|x)$$
$$g = \text{sigmoid function} = \sigma$$

$$p = P(y=1|x) = \frac{e^{\,\theta^T X}}{1 + e^{\,\theta^T X}} = \frac{1}{1+e^{-\theta^T x}} = \sigma(\theta^T x)$$

$$\eta = \log\left(\frac{p}{1-p}\right) = \theta^T x$$

Sigmoid function maps $\eta = \theta^T x$ to the probability p

GLM Framework

For completeness, the inverse g^{-1} that maps it back to the natural parameter is called the (canonical) link function.

In the case of Bernoulli Distribution
$$\eta = g(\theta^T x)$$
$$g^{-1}(\eta) = \theta^T x = \mu$$

$$\begin{tikzcd}
\eta \arrow[r, bend left] & \mathbb{E}[T(y)|x] = \mu \arrow[l, bend left]
\end{tikzcd}$$

$$g^{-1}(p) = \log\left(\frac{p}{1-p}\right) = \theta^T x$$

$$\log\left(\frac{p}{1-p}\right) \Rightarrow \eta$$

*In the case of linear regression, we assumed that the underlying distribution is Gaussian, you can show through the same derivations that the response function is the identity function.

How do we accommodate multiple classes?

Recall: Multinomial Distribution

Multinomial is the generalization of binomial / Bernoulli distribution to more than two outcomes.

Suppose we have n independent trials / items that can fall into any one of k classes independently with probabilities $p_1, p_2, \ldots, p_k$.

Let $n = x_1 + x_2 + x_3 + \cdots + x_k$, where each x_i represents the number of items that belong to class k.

$$P(X) = \binom{n}{x_1, x_2, \ldots, x_k} p_1^{x_1} p_2^{x_2} \ldots p_k^{x_k}$$

Bernoulli is a special case of the multinomial where $n = 1$ and $k = 2$.

Multinomial Distribution

For our purposes, we can simplify the math to $n = 1$, since we are treating each example independently.

$$P(y; p_1 \ldots p_k) = \begin{cases} p_1, & y = 1 \\ p_2, & y = 2 \\ p_3, & y = 3 \\ & \ldots \end{cases} \qquad P(\mathrm{y}; p_1 \ldots p_k) = \prod_{i=1}^{k} p_i^{\mathbf{1}\{y=i\}}$$

$$p_1 + p_2 + \ \ldots + p_k = 1$$

Multinomial logistic regression

Some changes:

$$T(y)_i = \mathbf{1}(y = i) \qquad T(y) = \begin{bmatrix} \mathbf{1}(y=1) \\ \mathbf{1}(y=2) \\ \mathbf{1}(y=3) \\ \vdots \\ \mathbf{1}(y=k-1) \end{bmatrix}$$

So if we have 5 classes/labels:

$$T(1) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, T(2) = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, T(3) = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, T(4) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, T(5) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

In practice, we output k classes

In theory, we only need k-1

$$P(y;p) = b(x)\exp(\eta^T T(y) - A(\eta))$$

$$\begin{aligned}
P(y;p) &= p_1^{1\{y=1\}} p_2^{1\{y=2\}} \ldots p_k^{1\{y=k\}} \\
&= p_1^{1\{y=1\}} p_2^{1\{y=2\}} \ldots p_k^{1-\sum_i^{k-1} 1\{y=i\}} \\
&= p_1^{(T(y))_1} p_2^{(T(y))_2} \ldots p_k^{1-\sum_i^{k-1}(T(y))_i} \\
&= \exp\left(\log\left(p_1^{(T(y))_1} p_2^{(T(y))_2} \ldots p_k^{1-\sum_i^{k-1}(T(y))_i}\right)\right)
\end{aligned}$$

$$= \exp\left(\left(T(y)\right)_1 \log(p_1) + \left(T(y)\right)_2 \log(p_2) + \cdots + (1 - \textstyle\sum_i^{k-1}\left(T(y)\right)_i) \log(p_k)\right)$$

$$= \exp\left(\begin{matrix} \left(T(y)\right)_1 \log(p_1) + \left(T(y)\right)_2 \log(p_2) + \cdots + \\ (\log(p_k) - \left(T(y)\right)_1 \log(p_k) - \left(T(y)\right)_2 \log(p_k) - \cdots - \left(T(y)\right)_{k-1} \log(p_k)) \end{matrix}\right)$$

$$= \exp\left(\left(T(y)\right)_1 \log\left(\frac{p_1}{p_k}\right) + \left(T(y)\right)_2 \log\left(\frac{p_2}{p_k}\right) + \cdots + \left(T(y)\right)_{k-1} \log\left(\frac{p_{k-1}}{p_k}\right) + \log(p_k)\right)$$

$$p(y:\eta) = \mathrm{b(x)}\exp(\eta^T T(y) - A(\eta))$$

$$= \exp\left((T(y))_1 \log\left(\frac{p_1}{p_k}\right) + (T(y))_2 \log\left(\frac{p_2}{p_k}\right) + \cdots + (T(y))_{k-1} \log\left(\frac{p_{k-1}}{p_k}\right) + \log(p_k)\right)$$

$$\eta = \begin{bmatrix} \log\left(\frac{p_1}{p_k}\right) \\ \log\left(\frac{p_2}{p_k}\right) \\ \vdots \\ \log\left(\frac{p_{k-1}}{p_k}\right) \end{bmatrix}$$

$$\eta_i = \log\left(\frac{p_i}{p_k}\right)$$

$$i = \{1, .., k-1\}$$

$$A(\eta) = -\log(p_k)$$

59 of 81

$$\eta_i = \log\left(\frac{p_i}{p_k}\right)$$

$$i = \{1, .., k-1\}$$

$$e^{\eta_i} = \frac{p_i}{p_k}$$

$$e^{\eta_i} p_k = p_i$$

$$e^{\eta_i} \frac{1}{\sum_{j=1}^{k} e^{\eta_j}} = p_i$$

$$\frac{e^{\eta_i}}{\sum_{j=1}^{k} e^{\eta_j}} = p_i$$

$$\sum_{i=1}^{k} p_i = 1$$

$$p_k \sum_{i=1}^{k} e^{\eta_i} = \sum_{i=1}^{k} p_i = 1$$

$$p_k \sum_{i=1}^{k} e^{\eta_i} = 1$$

$$p_k = \frac{1}{\sum_{i=1}^{k} e^{\eta_i}}$$

$$g(z) = \mathrm{softmax}(z) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

Softmax function ↓

$$\eta_i = \theta_i^T x$$

$$p_i = \boxed{\frac{e^{\theta_i^T x}}{\sum_{j=1}^{k} e^{\theta_j^T x}}}$$

60 of 81

$$h_\theta(x) = \begin{bmatrix} p(y=1|x;\theta) \\ p(y=2|x;\theta) \\ \vdots \\ p(y=k|x;\theta) \end{bmatrix} = g(\theta^T x + b)$$

Some changes:
θ has shape (D, k)
b has shape $(k, 1)$

D – dimensions
k – number of classes

$$g(z) = \text{softmax}(z) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

In practice, we implement softmax a bit differently, which you will have to figure out in the assignment #5

Bernoulli

prediction/model output

$$h_\theta(x) = P(y = 1|x)$$

Probability that $y = 1$
If $p \geq 0.5$, $\hat{y} = 1$

Multinomial

prediction/model output

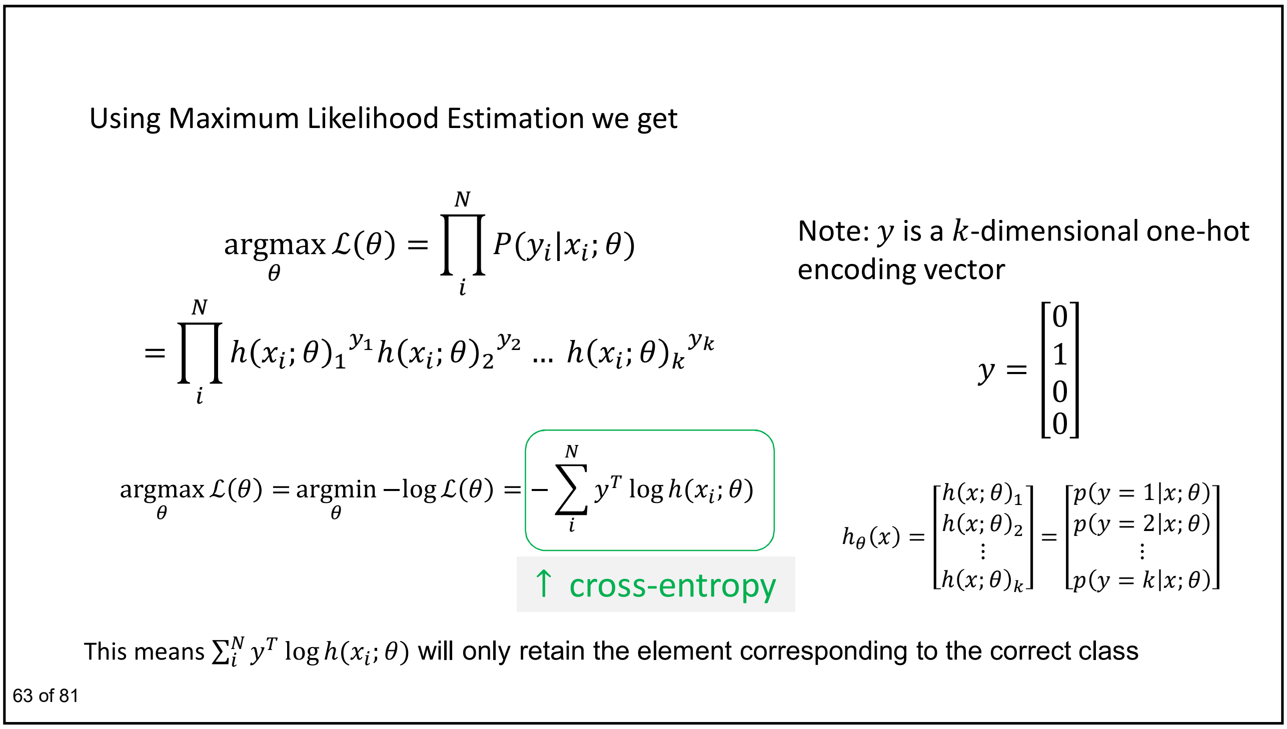
$$h_\theta(x) = \begin{bmatrix} P(y=1|x;\theta) \\ P(y=2|x;\theta) \\ \vdots \\ P(y=k|x;\theta) \end{bmatrix}$$

Probability that $y = 1$

Probability that $y = 2$

Probability that $y = k$

Technically, $p_k = 1 - (p_1 + \cdots + p_{k-1})$



Using Maximum Likelihood Estimation we get

$$\underset{\theta}{\operatorname{argmax}}\,\mathcal{L}(\theta) = \prod_{i}^{N} P(y_i|x_i;\theta)$$

$$= \prod_{i}^{N} h(x_i;\theta)_1{}^{y_1} h(x_i;\theta)_2{}^{y_2} \dots\ h(x_i;\theta)_k{}^{y_k}$$

$$\underset{\theta}{\operatorname{argmax}}\,\mathcal{L}(\theta) = \underset{\theta}{\operatorname{argmin}} -\log\mathcal{L}(\theta) = \boxed{-\sum_{i}^{N} y^T \log h(x_i;\theta)}$$

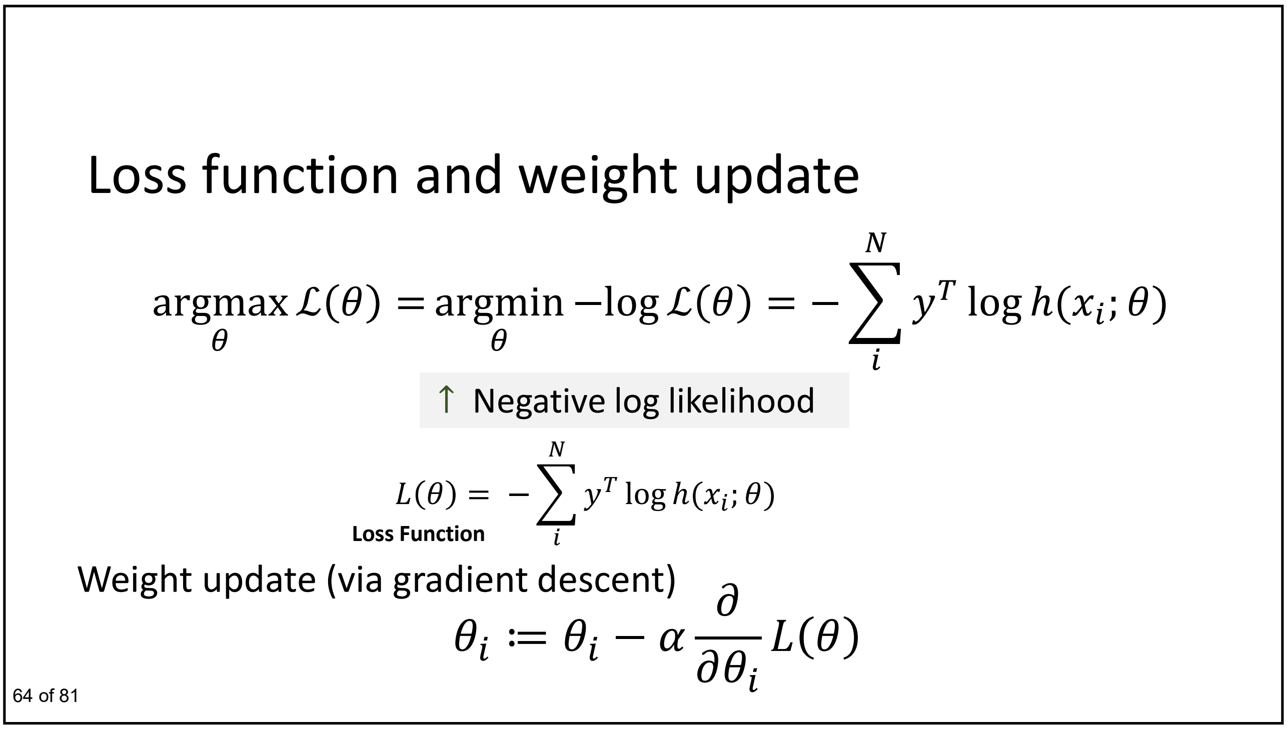
↑ cross-entropy

Note: y is a k-dimensional one-hot encoding vector

$$y = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$h_\theta(x) = \begin{bmatrix} h(x;\theta)_1 \\ h(x;\theta)_2 \\ \vdots \\ h(x;\theta)_k \end{bmatrix} = \begin{bmatrix} p(y=1|x;\theta) \\ p(y=2|x;\theta) \\ \vdots \\ p(y=k|x;\theta) \end{bmatrix}$$

This means $\sum_i^N y^T \log h(x_i;\theta)$ will only retain the element corresponding to the correct class



Loss function and weight update

$$\underset{\theta}{\operatorname{argmax}}\,\mathcal{L}(\theta) = \underset{\theta}{\operatorname{argmin}} -\log\mathcal{L}(\theta) = -\sum_{i}^{N} y^T \log h(x_i;\theta)$$

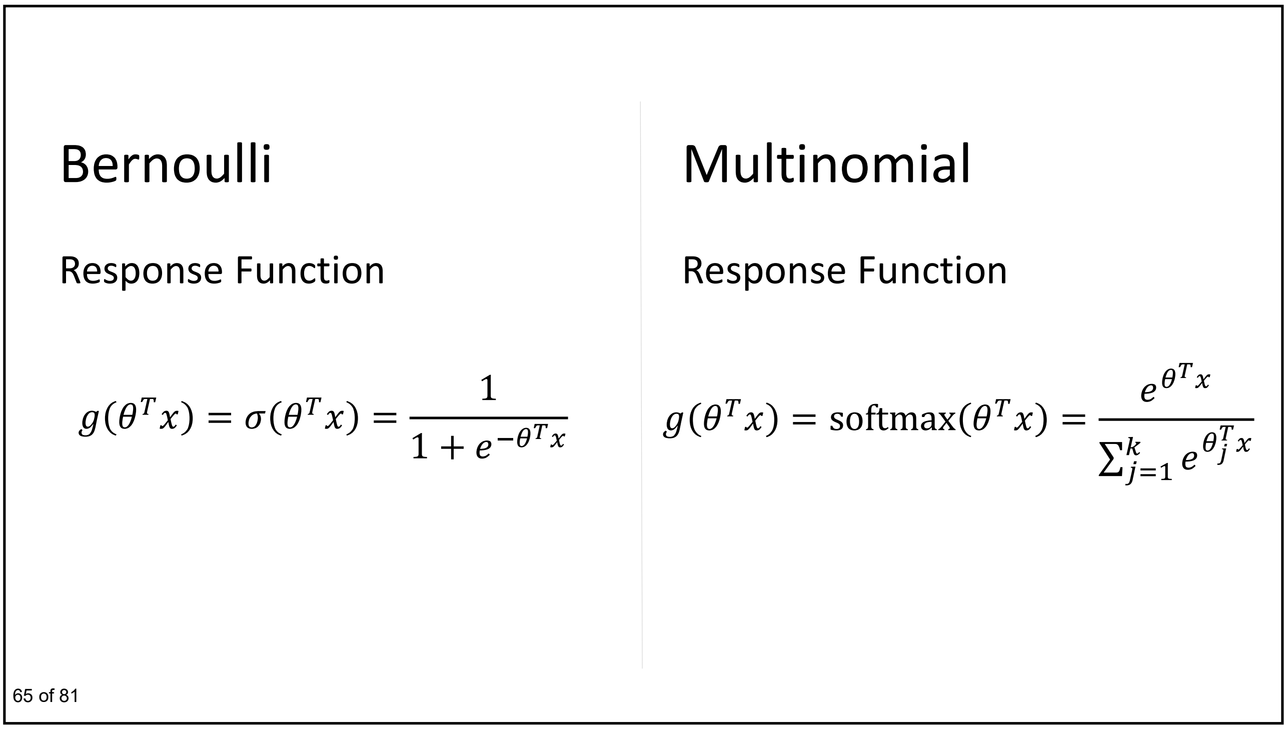
↑ Negative log likelihood

$$L(\theta) = -\sum_{i}^{N} y^T \log h(x_i;\theta)$$

Loss Function

Weight update (via gradient descent)

$$\theta_i := \theta_i - \alpha \frac{\partial}{\partial\theta_i} L(\theta)$$



Bernoulli

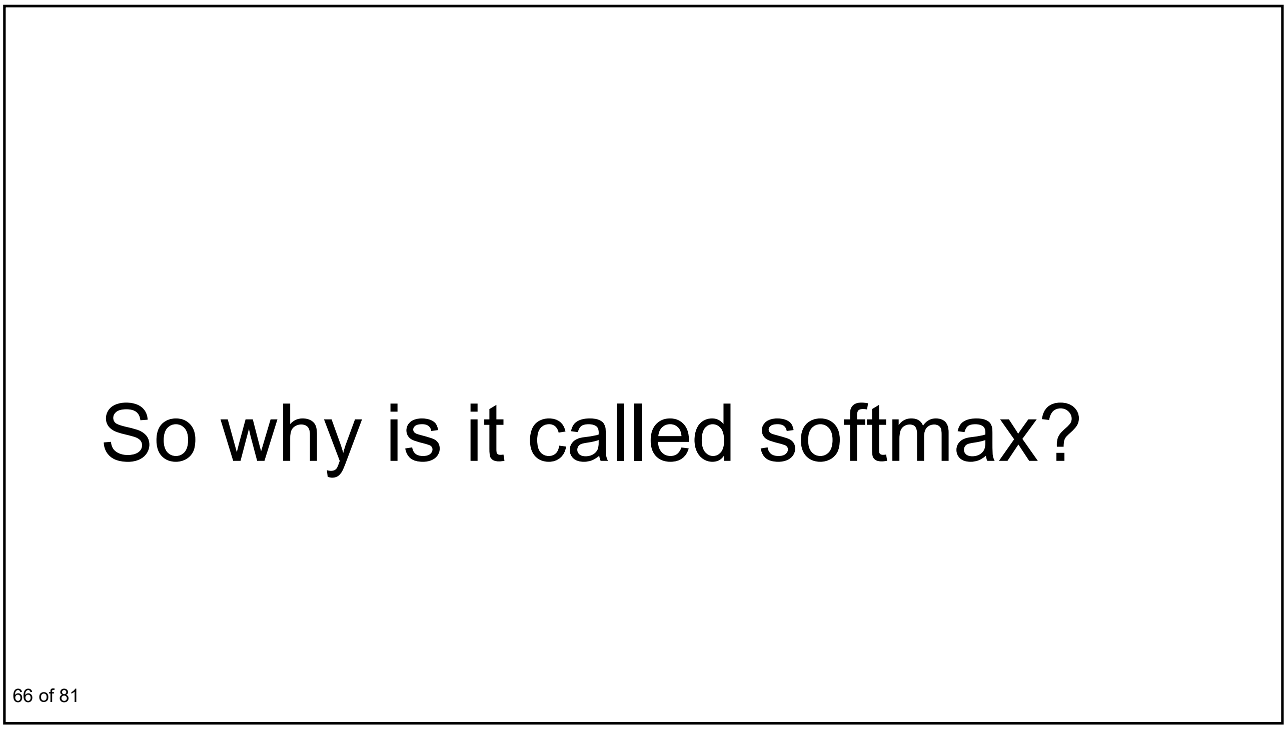
Response Function

$$g(\theta^T x) = \sigma(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

Multinomial

Response Function

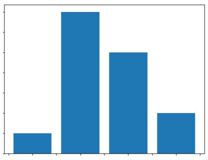
$$g(\theta^T x) = \text{softmax}(\theta^T x) = \frac{e^{\theta^T x}}{\sum_{j=1}^{k} e^{\theta_j^T x}}$$



So why is it called softmax?

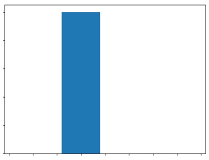
Why is it called "Softmax"?

Let $a = [1\ 7\ 5\ 2]$, its plot would look like this



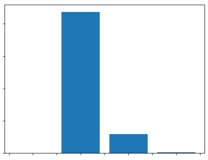
A hard max would give us $[0\ 1\ 0\ 0]$.

Our ground truth labels are like this.



A softmax would give us

[0.002 0.873 0.118 0.005]

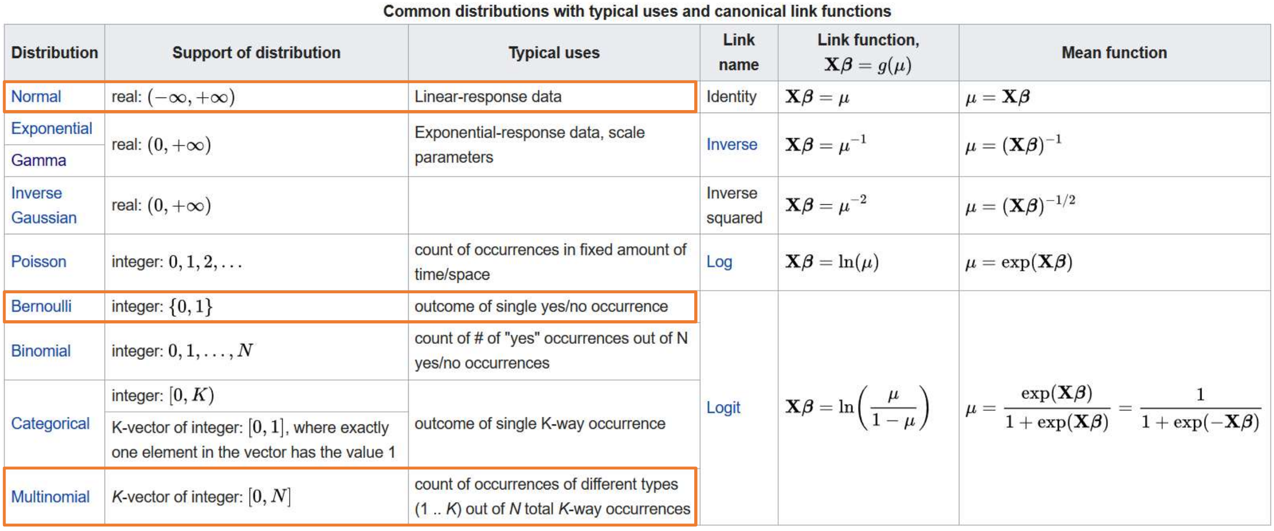


It's a "soft" approximation of the hard max.

The exponentiation makes it easier to push the probability of the correct class up.

Some Remarks

- Most probabilistic models have distributional assumptions
  - E.g. Linear regression – Gaussian, Logistic Regression - Bernoulli
- Before you apply these models, check first if your data satisfies the model's assumptions
  - For count data such as disease incidence, number of cars, etc.. Poisson distribution is more appropriate (or Negative Binomial if over-dispersed)
  - If your classes are ordinal, e.g. earthquake / typhoon alert levels, then an ordinary logistic regression (binomial or multinomial) will not do well as it assumes that the classes are independent.
    - If you are interested in this problem search for Ordinal Regression



Common distributions with typical uses and canonical link functions

Distribution	Support of distribution	Typical uses	Link name	Link function, $\mathbf{X}\boldsymbol{\beta}=g(\mu)$	Mean function
Normal	real: $(-\infty,+\infty)$	Linear-response data	Identity	$\mathbf{X}\boldsymbol{\beta}=\mu$	$\mu=\mathbf{X}\boldsymbol{\beta}$
Exponential Gamma	real: $(0,+\infty)$	Exponential-response data, scale parameters	Inverse	$\mathbf{X}\boldsymbol{\beta}=\mu^{-1}$	$\mu=(\mathbf{X}\boldsymbol{\beta})^{-1}$
Inverse Gaussian	real: $(0,+\infty)$		Inverse squared	$\mathbf{X}\boldsymbol{\beta}=\mu^{-2}$	$\mu=(\mathbf{X}\boldsymbol{\beta})^{-1/2}$
Poisson	integer: $0,1,2,\ldots$	count of occurrences in fixed amount of time/space	Log	$\mathbf{X}\boldsymbol{\beta}=\ln(\mu)$	$\mu=\exp(\mathbf{X}\boldsymbol{\beta})$
Bernoulli	integer: $\{0,1\}$	outcome of single yes/no occurrence	Logit	$\mathbf{X}\boldsymbol{\beta}=\ln\left(\frac{\mu}{1-\mu}\right)$	$\mu=\frac{\exp(\mathbf{X}\boldsymbol{\beta})}{1+\exp(\mathbf{X}\boldsymbol{\beta})}=\frac{1}{1+\exp(-\mathbf{X}\boldsymbol{\beta})}$
Binomial	integer: $0,1,\ldots,N$	count of # of "yes" occurrences out of N yes/no occurrences			
Categorical	integer: $[0,K)$	outcome of single K-way occurrence			
	K-vector of integer: $[0,1]$, where exactly one element in the vector has the value 1				
Multinomial	*K*-vector of integer: $[0,N]$	count of occurrences of different types (1 .. *K*) out of *N* total *K*-way occurrences			

Source: Wikipedia

A Little bit of Information Theory

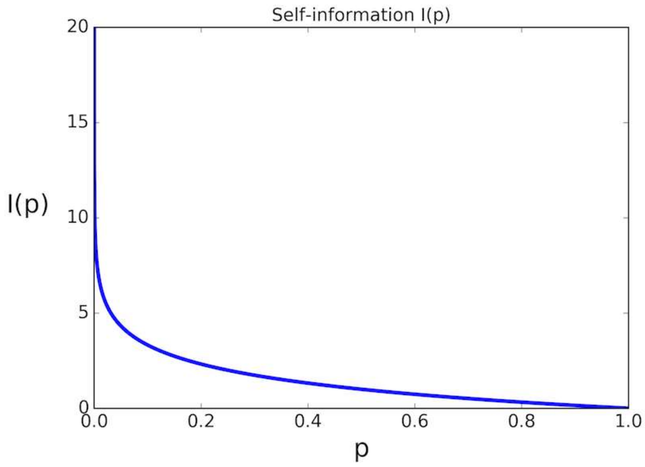
Information Content

Amount of information (sometimes called surprisal) contained in an event with some probability p_i.

$$I(p_i) = -\log p_i$$

When an unlikely outcome of an event is observed, we associate it with a high amount of information.

Conversely, when a more likely outcome is observed, we associate it with a smaller amount of information.



Information Content

Consider, for example, an event that always occur such as the sun rising from the east.

Since this is fully predictable, we will never be surprised about the outcome, which means that we gain zero information from such an experiment.

In contrast, if we observe a rare event, such as an earthquake, then we get more information / more surprised from it.

Entropy

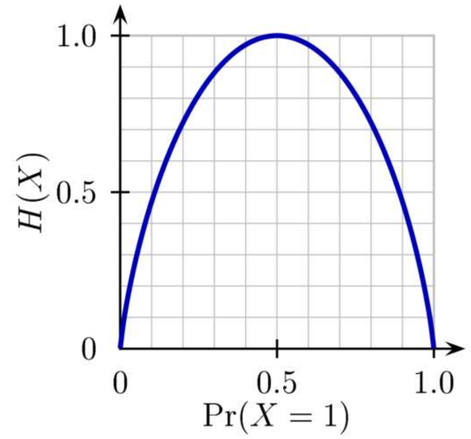
Given a probability distribution p, entropy is a measure of how uncertain the events are.

It is formulated as the average amount of information that you get from one sample drawn from a given probability distribution p.

$$H(p) = \mathbb{E}[I(p)] = \mathbb{E}[-\log p] = -\sum p_i \log p_i$$

So in general, it tells you how unpredictable a given probability distribution p is.

Entropy

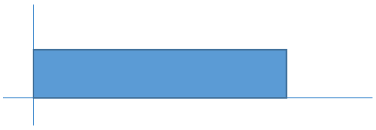


Entropy of a Bernoulli random variable X, as a function of X's probability of being 1

Entropy

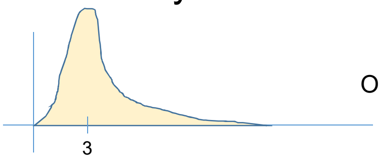
It also gives some information about the shape of the distribution.

If the distribution is flat (close to uniform) then entropy will be higher.



Everything is equally likely ⇒ unpredictable

If the distribution is peaked meaning some values are significantly more likely than others, then entropy will be lower



Outcome will likely be around 3 ⇒ predictable

Cross-Entropy

Let p and q be two probability distributions. The cross-entropy is defined as

$$H(p,q) = -\sum p_i \log q_i$$

It is very similar to entropy but instead of computing for the $\log p_i$, we compute for $\log q_i$.

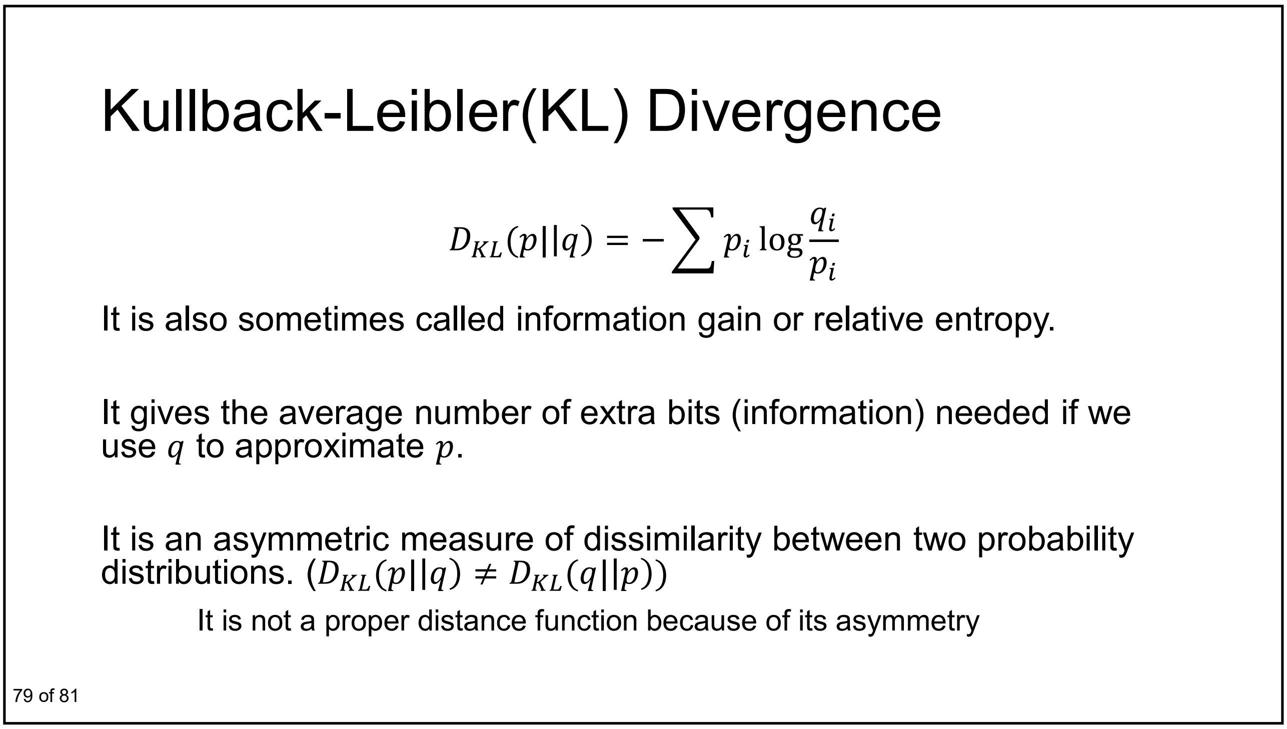
Cross-Entropy

We can think of p as the true probability distribution and q as the predicted probability distribution. We can rewrite cross-entropy as shown below, where D_{KL} denotes the Kullback-Leibler Divergence

$$H(p,q) = -\sum p_i \log q_i = H(p) + D_{KL}(p||q)$$

$$\begin{aligned} D_{KL}(p||q) &= -\sum p_i \log \frac{q_i}{p_i} = -\sum p_i (\log q_i - \log p_i) \\ &= -\sum p_i \log q_i + \sum p_i \log p_i \\ &= H(p,q) - H(p) \end{aligned}$$

So cross-entropy is getting the average amount of information if we use q (predicted distribution) instead of p (actual distribution).



Kullback-Leibler(KL) Divergence

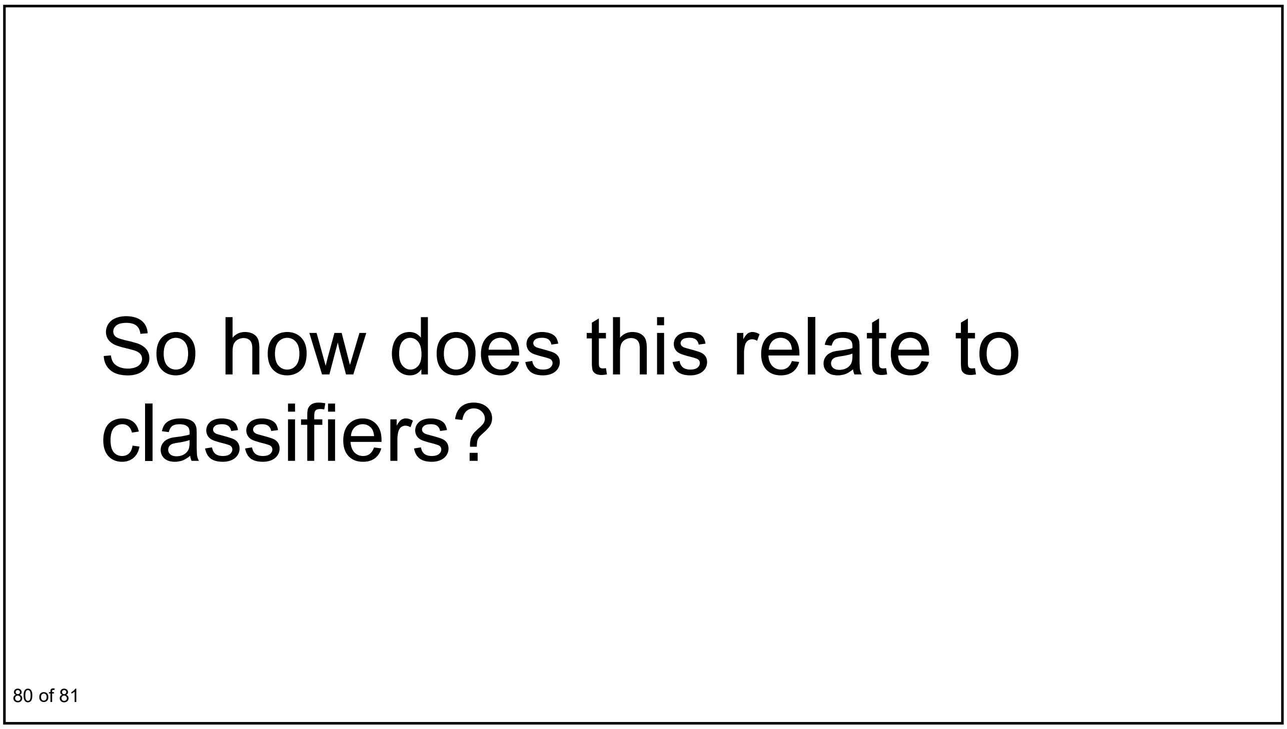
$$D_{KL}(p||q) = -\sum p_i \log \frac{q_i}{p_i}$$

It is also sometimes called information gain or relative entropy.

It gives the average number of extra bits (information) needed if we use q to approximate p.

It is an asymmetric measure of dissimilarity between two probability distributions. ($D_{KL}(p||q) \neq D_{KL}(q||p)$)

It is not a proper distance function because of its asymmetry



So how does this relate to classifiers?

So how does this relate to classifiers?

As you may have noticed, cross-entropy is commonly used in the objectives of classifiers.

Cross-entropy $H(p,q) = -\sum p_i \log q_i$

Logistic Regression

$$-\sum_{i}^{N}[y_i \log(h(x_i;\theta)) + (1-y_i)\log(1-h(x_i;\theta))\,]$$

Softmax Regression

$$-\sum_{i}^{N} y^T \log h(x_i;\theta)$$

So how does this relate to classifiers?

We can think of the labels as the true distribution p, and the predictions as the predicted distribution q.

	Cat	Dog	Fox	Cow	Bear	Fish
True Distribution (p)	0.00	0.00	0.00	0.00	1.00	0.00
Predicted Distribution (q)	0.02	0.30	0.25	0.25	0.05	0.13

$$H(p,q) = H(p) + D_{KL}(p||q) = 1.3010$$

	Cat	Dog	Fox	Cow	Bear	Fish
True Distribution (p)	0.00	0.00	0.00	0.00	1.00	0.00
Predicted Distribution (q)	0.02	0.00	0.05	0. 05	0.85	0.03

$$H(p,q) = H(p) + D_{KL}(p||q) = 0.0706$$

Looking at this form of the cross-entropy, we can think of $H(p)$ as the optimal encoding and $D_{KL}(p||q)$ is the additional “error” due to how mismatched q is from the true distribution p. If p and q are the same, then $D_{KL}(p||q) = 0$.

References / Slide Credits

- Andrew Ng – CS229
- Eric Xing – CMU 10-701

83 of 81

Homework 3 – Polynomial Regression

- We will be implementing Polynomial Regression
- Will be posted tonight
- Due by March 23 (Monday)

84 of 81