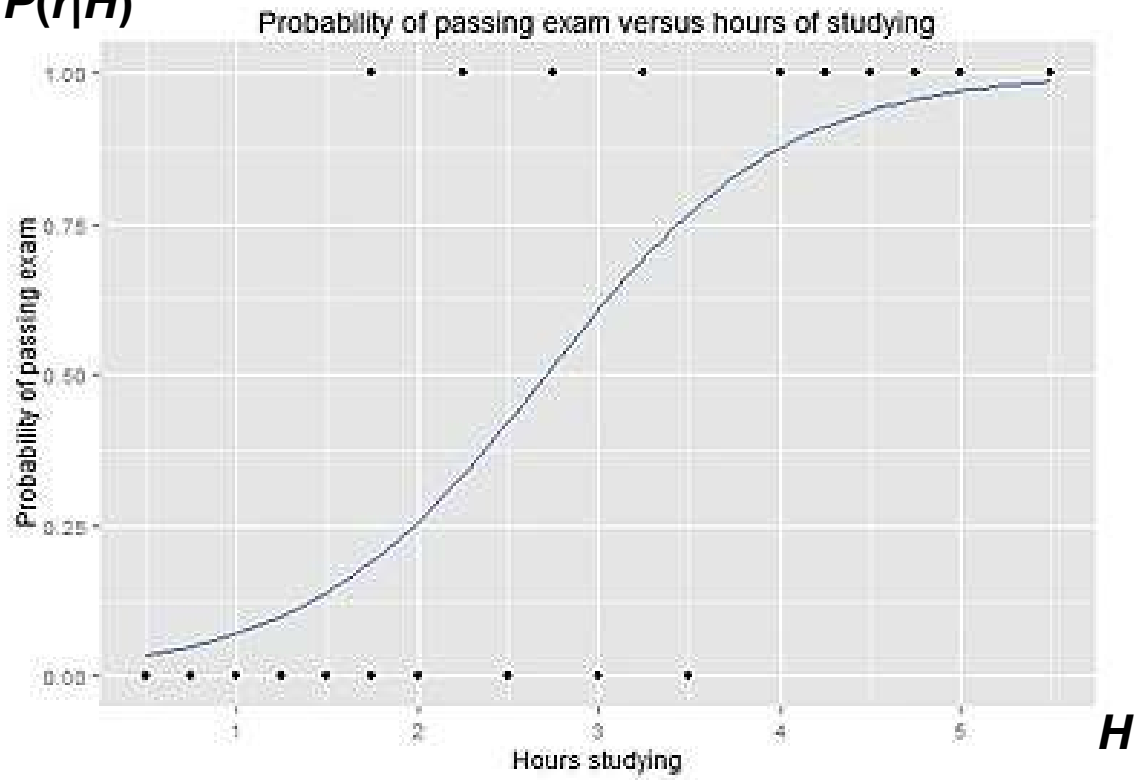
Logistic Regression

- Logistic regression is a type of regression analysis that models the probability of a certain class or event indicated by a dependent variable.
- There can be one or more independent variables ("predictors").
- For the binary-case (binomial, dichotomous) there are two categories such as positive/negative, healthy/sick.
- The multi-class (multinomial, polychotomous) version is also called “softmax”. This is different than the ordinal case (with dependent variables that are ordered).



What is the probability that a student passing with *H*=3 hours?

If the answer is 75% chance (3 out of 4 times), then the odds ratio is 3 to 1.

If the odds ratio is 2, then the probability of passing is 2/3.

Hours	0.50	0.75	1.00	1.25	1.50	1.75	1.75	2.00	2.25	2.50
Pass	0	0	0	0	0	0	1	0	1	0

2.75	3.00	3.25	3.50	4.00	4.25	4.50	4.75	5.00	5.50
1	0	1	0	1	1	1	1	1	1

Why not "just" regression?

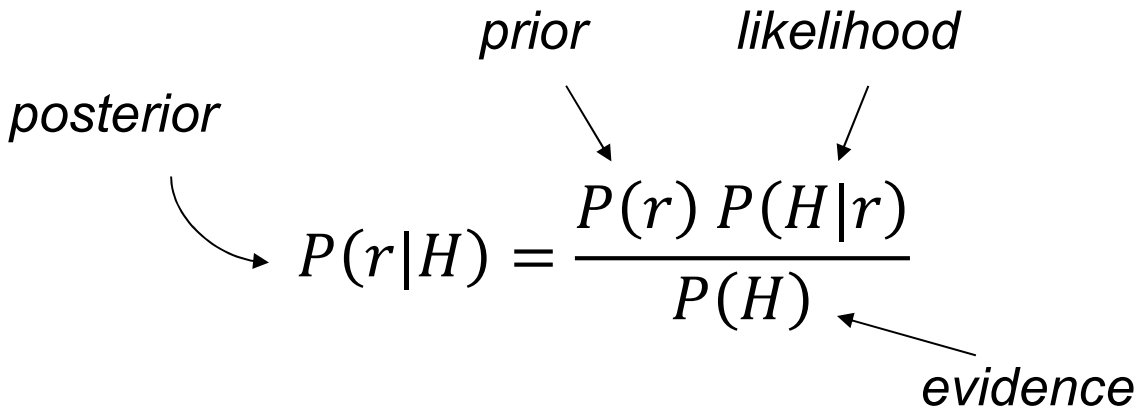
In ordinary regression (OLS) :

- $y = w_0 + w \times H$ and $r = y + \delta$
- Minimize MSE (sum of squares of δ values - residuals).

But when $r \in \{0, 1\}$:

- The error is hard to define because the predicted values, *y*, can be greater than 1 or less than 0.
- δ is not normally distributed because *r* takes on only two values.

Bayes' Rule



r is the class-label (pass/fail)

$P(r = 0) + P(r = 1) = 1$

$P(H) = p(H \mid r = 1)P(r = 1) + p(H \mid r = 0)P(r = 0)$

$P(r = 0 \mid H) + P(r = 1 \mid H) = 1$

For extending Bayes' rule to multivariate case to use more than one independent variables we will assume conditional independence (more on this later)

Generative vs Discriminative

- To estimate the posterior probability using Bayes' rule for classification, we typically assume the likelihood $P(X|C=c)$ for each class c to follow a distribution (Gaussian, Bernoulli, or Multinomial) that generates the observations (independent variables). Naïve Bayes, Discriminant Analysis, HMM, Gaussian Mixture Models are examples of generative models. Variational Autoencoders are recently popular, they can be used for generating fake observations as well as classification.
- Famous quote from Vladimir Vapnik in machine learning: "When solving a problem of interest, do not solve a more general problem as an intermediate step".
- In discriminative models, as we directly model the posterior probability we rely on fewer assumptions about the underlying data distribution. Logistic Regression is a discriminative classifier as well as SVMs, Decision Trees, Neural Networks, KNN and so on.

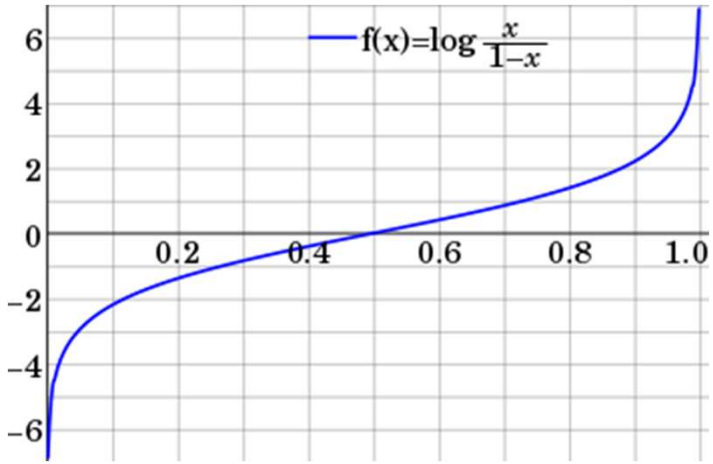
Logistic Regression

- Uses regression to model the odds ratio or equivalently its logarithm.
- Class-labels are not directly used as the dependent variable. The log-odds (the logarithm of the odds ratio) is modeled as a linear combination of the independent variables.

For the example problem: $\log(p/(1-p)) = w_0 + w \times H$

- p is the probability of class-1, $P(r=1 \mid H)$
  - Range is [0, 1]
- $p/(1-p)$ is the "odds ratio"
  - Range is $[0, \infty)$
- $\log(p/(1-p))$: log odds ratio, or "logit"
  - Range is $(-\infty, +\infty)$

Logit Function



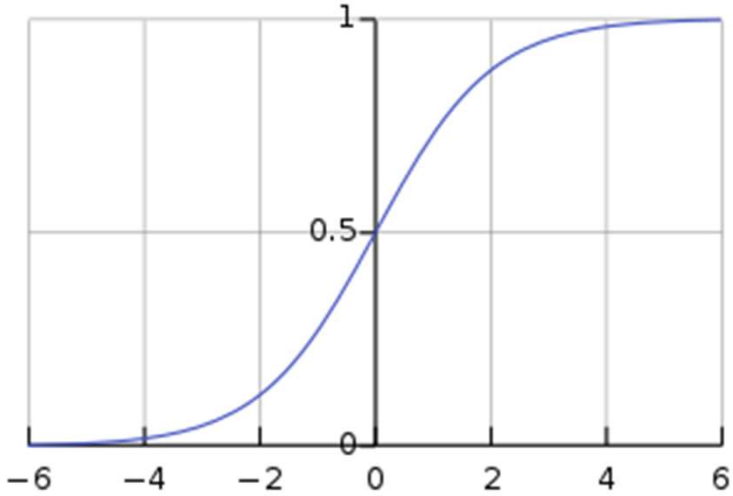
a= logit(p) = log(p/(1–p))

By simple algebraic manipulation,

- p = exp(a) / (1 + exp(a))
- p = 1 / (1 + exp(–a))

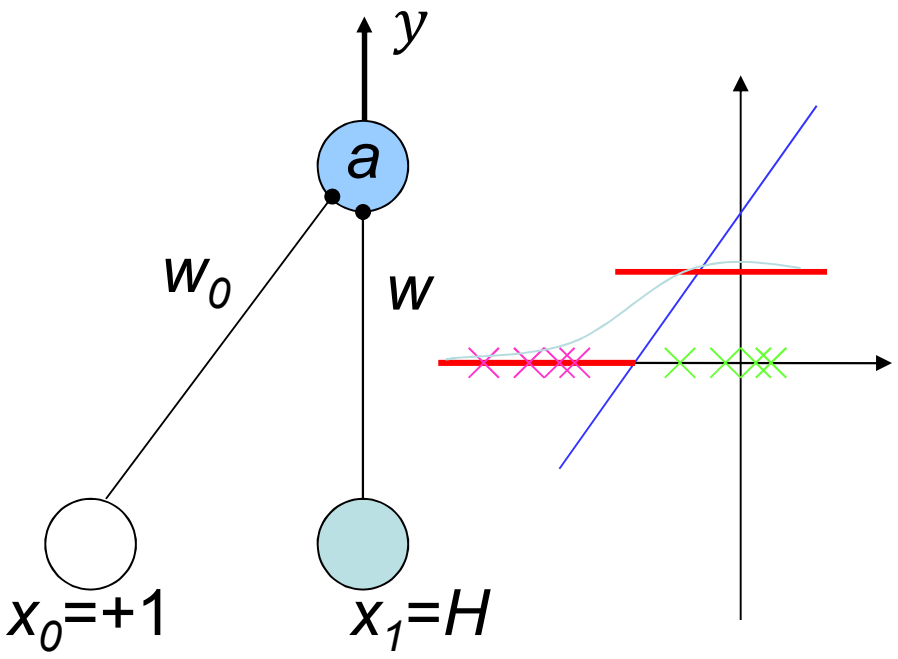
- The inverse of logit is logistic sigmoid, “logsig”

p = logsig(a)



Remember what a perceptron does …

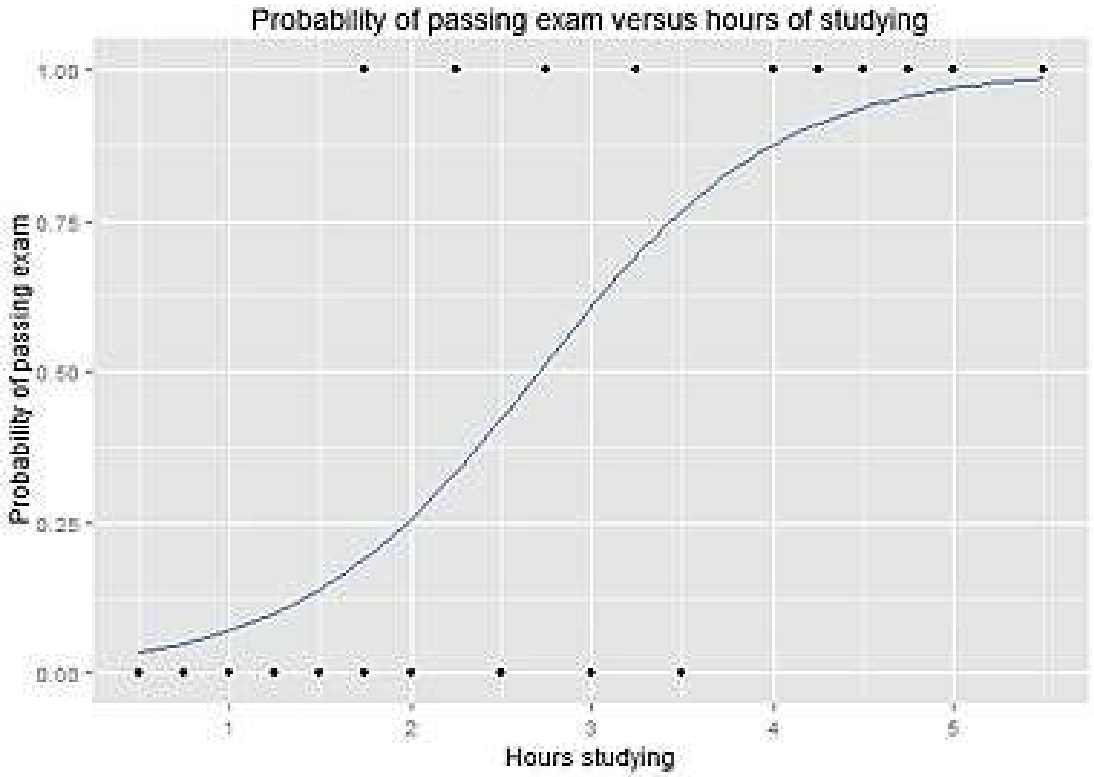
…for classification it returns 0 or 1* based on $[w^T x+w_0>0]$ (using a step function)



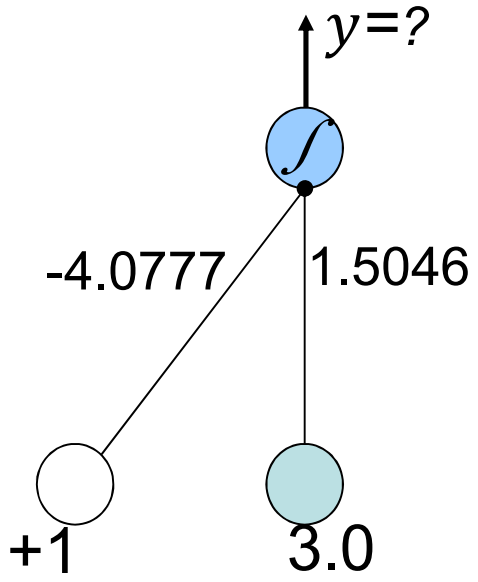
Do you remember we also said using sigmoid is a good idea? This discussion relates to the reason: $[\text{logsig}(w^T x+w_0)>0.5]$)

$$y = logsig(a) = \frac{1}{1+\exp(-w^T x - w_0)}$$

* using Iverson bracket notation



What is the probability that a student passing with H = 3 hours?



a = logit(p) = log(1/(1−p)) = 1.5046 × H − 4.0777

Probability of passing exam,

$$p = 1/(1+\exp(-a)) = \frac{1}{1+\exp(-(1.5046 \cdot \mathrm{H} - 4.0777))}$$

The Logistic Regression Model

The logit models $\log(p/(1-p)) = w_0 + w_1 x_1 \; [+ w_2 x_2 + \ldots]$

Here is p is the probability that the example belongs to class-1.
Again note that this is different than ordinary regression, in which $r \cong y(x) = w_0 + w_1 x_1 \; [+ w_2 x_2 + \ldots]$

Inference: Given the value of the feature(s), what is the probability $p = P(r=1 \mid x)$?

Sensitivity analysis: severity of change of the prediction when a variable is modified. For example, how does increasing H by 1 change your confidence in your exam?

OR for $H=3$ is $p/(1-p) = \exp(w \times 3 + w_0)$

OR for $H=4$ is exp(w) times higher because it is $\exp(w \times 4 + w_0)$…
For $w \cong 1.5$, the new ratio is $2.72^{1.5} \cong 4.5$ times higher.

```python
import numpy as np
def sigmoid(x):
  return 1 / (1 + np.exp(-x))
w=1.5046
w0=-4.0777
def weighted_sum_a(H):
  return w*H + w0
p = sigmoid(weighted_sum_a(3))
odds_for_H3 = p/(1-p)
print('odds ratio for H = 3 is', odds_for_H3)
p = sigmoid(weighted_sum_a(4))
odds_for_H4 = p/(1-p)
print('odds ratio for H = 4 is', odds_for_H4)
print('The change in the ratio is', odds_for_H4/odds_for_H3, 'times')
print('This is equal to exp(w), which is also =', np.exp(w))
p = sigmoid(weighted_sum_a(5))
odds_for_H5 = p/(1-p)
print('odds ratio for H = 5 is', odds_for_H5)
print('By increasing H from 4 to 5, the change in the odds ratio is', \
      odds_for_H5/odds_for_H4, 'times, again!')
print('exp(w) is a measure of importance/sensitivity of H for pass/fail')
print('This is also clear from Slide 9, OR = p/(1-p) = exp(wH+w0)')
print('odds ratio for H = 3 is', np.exp(w*3+w0))
```

```
p = sigmoid(weighted_sum_a(4))
odds_for_H4 = p/(1-p)
print('odds ratio for H = 4 is', odds_for_H4)
print('The change in the ratio is', odds_for_H4/odds_for_H3, 'times')
print('This is equal to exp(w), which is also =', np.exp(w))
p = sigmoid(weighted_sum_a(5))
odds_for_H5 = p/(1-p)
print('odds ratio for H = 5 is', odds_for_H5)
print('By increasing H from 4 to 5, the change in the odds ratio is', \
      odds_for_H5/odds_for_H4, 'times, again!')
print('exp(w) is a measure of importance/sensitivity of H for pass/fail')
print('This is also clear from Slide 9, OR = p/(1-p) = exp(wH+w0)')
print('odds ratio for H = 3 is', np.exp(w*3+w0))


```

```
odds ratio for H = 3 is 1.5466634533616537
odds ratio for H = 4 is 6.963623801608575
The change in the ratio is 4.502352329120615 times
This is equal to exp(w), which is also = 4.50235232912062
odds ratio for H = 5 is 31.352687842292106
By increasing H from 4 to 5, the change in the odds ratio is 4.502352329120613 times, again!
exp(w) is a measure of importance/sensitivity of H for pass/fail
This is also clear from Slide 9, OR = p/(1-p) = exp(wH+w0)
odds ratio for H = 3 is 1.5466634533616537
```

Logistic Regression Optimization

We can now formulate the likelihood function and find the parameters (weights w and w_0) that maximize this likelihood.

Likelihood measures the goodness of fit of a statistical model to a sample of data for given values of the unknown parameters; for example, everyone you talk to after the exam complains about how bad it was, what are the chances that the mean of the distribution of the exam scores is 90? Or, some students complain but some others say it went great, what are the chances (likelihood) of having a standard deviation of only 2-3 points?

Typically, we maximize the log-likelihood.

Likelihood

- Result of tossing a coin is $\in$ {Heads,Tails}
- Random variable $r \in \{1,0\}$

  Sample: $r = \{r^t\}$, for $t = 1..N$

For example, $r = [1, 1, 1, 0, 1, 0, 1, 1, 0, 0]^T$ for N=10

What is the likelihood that the coin is biased such that it comes up as Heads 90% of the time?

$0.9 \times 0.9 \times 0.9 \times 0.1 \times 0.9 \times 0.1 \times 0.9 \times 0.9 \times 0.1 \times 0.1 = 0.0000531441$

What is the likelihood that the coin is fair?

$0.5^{10} = 0.0009765625$

#{Heads}/#{Tosses} = $(1/N) \sum_t r^t = 0.6$ has the maximum likelihood. But why? How do we reach that formula? For this special case, we can solve for it but in general this needs an iterative optimization method such as gradient descent, Newton's method, etc.

Maximum Likelihood Estimation (MLE) in Machine Learning

- Assume a good model and optimize the parameters so that those parameter settings (weights etc.) have the maximum compatibility with the observed data.
- Find the optimal weights of a neural network (classification or regression)
- Find the clusters in a dataset
- Find distribution parameters

Maximum Likelihood Estimation (MLE) in Machine Learning

- Given the dataset X=$\{s^t\}$, where each example is in the form $s^t=(x^t,r^t)$, the likelihood of θ is:

$$l(\theta|X) = P(X|\theta) = \prod_t P(s^t|\theta)$$

- Log likelihood

$$L(\theta|X) = \log l(\theta|X) = \sum_t \log P(s^t|\theta)$$

- Maximum likelihood estimator (MLE)

$$\theta^* = \text{argmax}_\theta L(\theta|X)$$

Bernoulli Distribution

The Bernoulli distribution is the discrete probability distribution of a random variable which takes the value 1 with probability p and the value 0 with probability $q = 1 - p$

$$P(r^t = 1 \mid \theta = p) = p$$

$$P(r^t = 0 \mid \theta = p) = 1 - p = q$$

A better formulation for mathematical operations needed for the derivations is:

$$P(r^t \mid \theta = p) = p^{r^t} (1 - p)^{(1 - r^t)}$$

For the coin tossing dataset, we do not have any input variables, just the class-labels… So, p is fixed and the task is to find the best constant for p that maximizes the likelihood.

MLE for Bernoulli Distribution

Log-likelihood is

$$\mathcal{L}(\theta|\mathcal{X}) = \sum_t \log(p^{r^t}(1-p)^{(1-r^t)})$$

$$= \sum_t [r^t \log(p) + (1-r^t)\log(1-p)]$$

For this special (simple) case, we do not need an iterative optimization tool, we can take the derivative of the log-likelihood with respect to p and obtain the optimal solution to be $p^* = (1/N)\sum_t r^t$ as intuitively expected.

MLE for Logistic Regression

We can view classification in machine learning as repeating Bernoulli trials with the parameter p that depends on the input variables (for example, 3 hours studying, H=3, gives $p \cong 0.61$ probability: One student studies 3 hours and pass but another student may study the same amount of time but end up failing, it is probabilistic!)

So we still have: $P(r^t \mid \theta = p) = p^{r^t} (1 - p)^{(1 - r^t)}$

But p is not fixed… it is a function of the input variables (for example H): $p = 1 / (1 + \exp(-wH - w_0))$

Therefore, the task is to find the optimal w and w_0

Gradient descent is one way of doing that optimization that Perceptron uses. As you see in machine learning libraries Logistic Regression has other options (some are faster, some work mostly with big datasets) for the "solver".

References

Python program that I wrote for demonstration:
https://colab.research.google.com/drive/1pJA3iDK3Fbctdbu2mnVRK7OSchZoJuNq?usp=sharing

Ethem Alpaydin's book, Chapters 3 and 4
https://www.cmpe.boun.edu.tr/~ethem/i2ml3e/

https://msu.edu/~kenfrank/introduction%20to%20logistic%20regression.pptx

https://en.wikipedia.org/wiki/Logistic_regression#Model_fitting