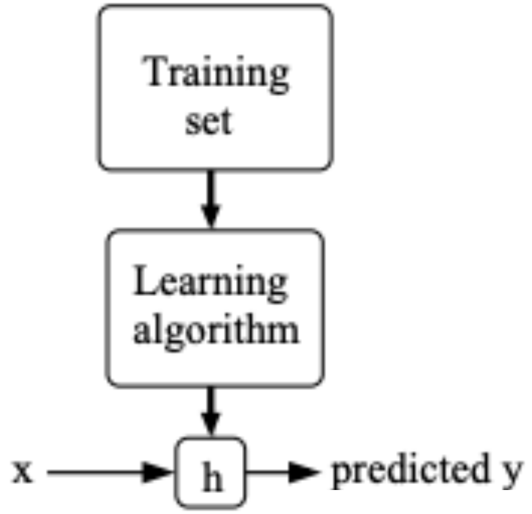
COMP [56]630– Machine Learning

Lecture 7 – Logistic Regression

The basics

- Input: a set of inputs X = {x_1, x_2, … x_n}, also called ***features***
- Output: a set of expected outputs or ***targets*** Y = {y_1, y_2, … y_n}
- Goal: to learn a function $h : X \rightarrow Y$ such that the function $h(x_i)$ is a good predictor of the corresponding value y_i
    - h(x) is called the ***hypothesis***
- If the target is continuous the problem setting is called ***regression.***
- If the target is discrete or categorical, the problem is called ***classification.***



Example

- Suppose we have a dataset giving the living areas and prices of 47 houses from Stillwater, OK

Living area (ft^2)	# bedrooms	Price (1000$s)	House Type
1643	4	256	Condo
1356	3	202	Apartment
1678	3	287	House
…	…	…	
3000	4	400	House

Features (X)

Targets (Y)

Logistic Regression

- Goal: formulate a hypothesis function h(x) which will model the 3-d input feature (size, # bedrooms, price) and produce the expected target value (type of home i.e. condo, apartment, house, etc.).
- Let us consider a 2-class problem i.e. a binary classifier that says whether the given home is a house or not a house.
  - Represent as 0 and 1
  - 0 → negative class
  - 1 → positive class

Discrete outputs!

- Given $x^{(i)}$, the corresponding $y^{(i)}$ is also called the label for the training example.

Logistic Regression

- We could approach the classification problem ignoring the fact that y is discrete-valued, and use our old linear regression algorithm to try to predict y given some features x

Logistic Regression

- We could approach the classification problem ignoring the fact that y is discrete-valued, and use our old linear regression algorithm to try to predict y given some features x
  - Does not work all the time.
  - Does not make sense to have predictions greater than 1 or less then 0.
    - Since the categories are 1 and 0 and $y \in \{0, 1\}$.

Logistic Regression

- We could approach the classification problem ignoring the fact that y is discrete-valued, and use our old linear regression algorithm to try to predict y given some features x
  - Does not work all the time.
  - Does not make sense to have predictions greater than 1 or less then 0.
    - Since the categories are 1 and 0 and $y \in \{0, 1\}$.
- New hypothesis function:

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

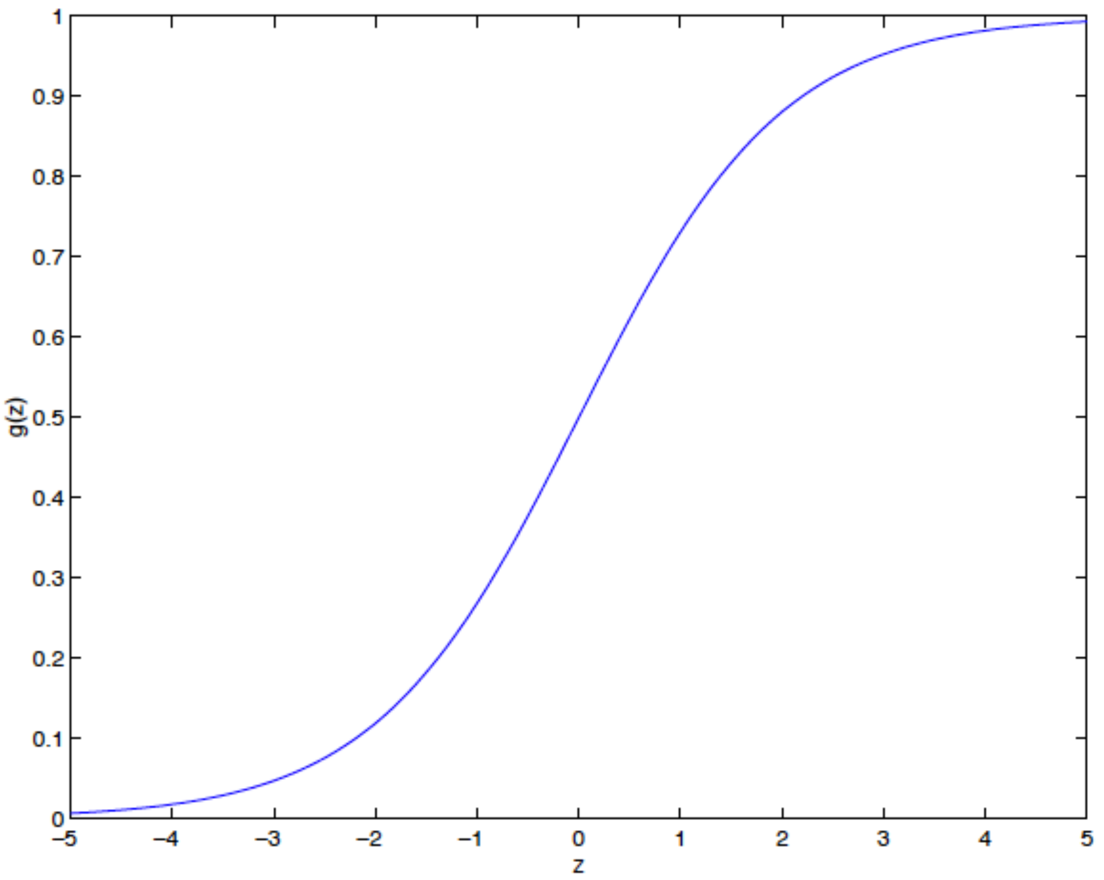
Logistic Regression

- We could approach the classification problem ignoring the fact that y is discrete-valued, and use our old linear regression algorithm to try to predict y given some features x
  - Does not work all the time.
  - Does not make sense to have predictions greater than 1 or less then 0.
    - Since the categories are 1 and 0 and y ∈ {0, 1}.
- New hypothesis function:

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

  - where $g(z) = \frac{1}{1 + e^{-z}}$ ← **Logistic Function** $\theta^T x = \theta_0 + \sum_{j=1}^{n} \theta_j x_j$

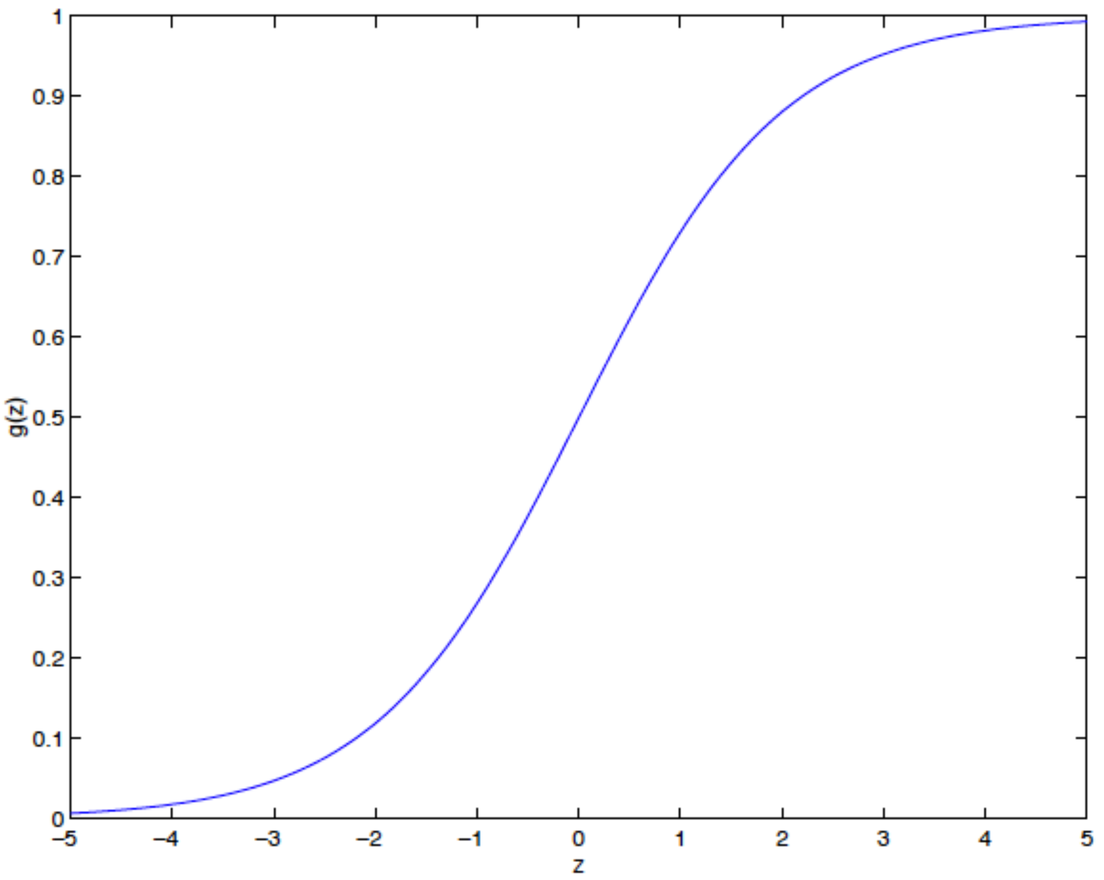
Logistic Function



- g(z) tends towards 1 as z → ∞
- g(z) tends towards 0 as z → −∞.

- What does this tell us?

Logistic Function



- g(z) tends towards 1 as z → ∞
- g(z) tends towards 0 as z → −∞.

- What does this tell us?
- **g(z), and hence also h(x), is always bounded between 0 and 1.**

How do we get θ?

- Gradient Descent!
- What do we need for gradient descent?
  - An objective function J(θ)
  - A Learning rate α
  - An initial “guess” for θ called θ_j
- Then, update θ_j until convergence as follows

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

Defining the Objective Function J(θ)

- Let us say that
$$\begin{aligned} P(y=1 \mid x;\theta) &= h_\theta(x) \\ P(y=0 \mid x;\theta) &= 1-h_\theta(x) \end{aligned}$$

- Or, more concisely:

$$p(y \mid x;\theta) = (h_\theta(x))^y \, (1-h_\theta(x))^{1-y}$$

Defining the Objective Function J(θ)

- Given: $$p(y \mid x; \theta) = (h_\theta(x))^y \, (1 - h_\theta(x))^{1-y}$$

- We want to estimate θ that will capture the dependency between y and x. When we wish to explicitly view this as a function of θ, we will instead call it the ***likelihood function*** that maximizes p(y|X; θ) and is given by

$$L(\theta) \; = \; \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)}; \theta)$$

Defining the Objective Function J(θ)

- Given: $$p(y \mid x; \theta) = (h_\theta(x))^y \, (1 - h_\theta(x))^{1-y}$$

- We want to estimate θ that will capture the dependency between y and x. When we wish to explicitly view this as a function of θ, we will instead call it the ***likelihood function*** that maximizes p(y|X; θ) and is given by

$$\begin{aligned} L(\theta) &= \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)}; \theta) \\ &= \prod_{i=1}^{m} \left(h_\theta(x^{(i)})\right)^{y^{(i)}} \left(1 - h_\theta(x^{(i)})\right)^{1-y^{(i)}} \end{aligned}$$

Defining the Objective Function J(θ)

- Now, given this probabilistic model how can we get θ?

Defining the Objective Function J(θ)

- Now, given this probabilistic model how can we get θ?
- The principal of ***maximum likelihood*** says that we should should choose θ that makes the data as high probability as possible. i.e., we should choose θ to maximize L(θ).

Defining the Objective Function $J(\theta)$

- Now, given this probabilistic model how can we get θ?
- The principal of ***maximum likelihood*** says that we should should choose θ that makes the data as high probability as possible. i.e., we should choose θ to maximize $L(\theta)$.
- Instead of maximizing $L(\theta)$, we can also maximize any strictly increasing function of $L(\theta)$. So we will maximize the log likelihood $\ell(\theta)$:

$$\ell(\theta) = \log L(\theta)$$

Defining the Objective Function J(θ)

- Now, given this probabilistic model how can we get θ?
- The principal of ***maximum likelihood*** says that we should should choose θ that makes the data as high probability as possible. i.e., we should choose θ to maximize L(θ).
- Instead of maximizing L(θ), we can also maximize any strictly increasing function of L(θ). So we will maximize the log likelihood ℓ(θ):

$$\ell(\theta) = \log L(\theta)$$

$$= \sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)}))$$

How do we get θ?

- Gradient Descent!
- Then, update θ_j until convergence as follows

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

- So what is $\frac{\partial}{\partial \theta_j} J(\theta)$?

$$l(\theta) = \log(L(\theta)) \qquad\qquad h(x) = g(\theta^T x)$$

$$= \sum \left[y \cdot \log(h(x)) + (1-y) \cdot \log(1-h(x)) \right]$$

$$= \sum \left[y \cdot \log(g(\theta^T x)) + (1-y) \cdot \log(1-g(\theta^T x)) \right]$$

$$\frac{d\,l(\theta)}{d\theta} = \sum \frac{d}{d\theta} \left[y \cdot \log(g(\theta^T x)) + (1-y) \cdot \log(1-g(\theta^T x)) \right]$$

For simplicity, ignore the summation.

$$\therefore \frac{d\,l(\theta)}{d\theta} = y \cdot \frac{d}{d\theta} \left(\log(g(\theta^T x)) \right) + (1-y) \cdot \frac{d}{d\theta} \left(\log(1-g(\theta^T x)) \right)$$

we know that $\frac{d}{dz}(\log(z)) = \frac{1}{z}$

$$\therefore \frac{d\,l(\theta)}{d\theta} = \frac{y}{g(\theta^T x)} \cdot \frac{d}{d\theta}\left(g(\theta^T x)\right) + \left(\frac{(1-y)}{1-g(\theta^T x)}\right) \cdot \frac{d}{d\theta}\left(1-g(\theta^T x)\right)$$

$$= \left[\frac{y}{g(\theta^T x)} + \frac{(-1)\cdot(1-y)}{1-g(\theta^T x)}\right] \cdot \frac{d}{d\theta}\left(g(\theta^T x)\right)$$

$$= \frac{y(1-g(\theta^T x)) - (1-y)\cdot g(\theta^T x)}{g(\theta^T x)\cdot(1-g(\theta^T x))} \cdot \frac{d}{d\theta}\left(g(\theta^T x)\right)$$

$$= \frac{y - y \cdot g(\theta^T x) - g(\theta^T x) + y \cdot g(\theta^T x)}{g(\theta^T x)(1 - g(\theta^T x))} \cdot \frac{d}{d\theta}(g(\theta^T x))$$

$$= \frac{y - g(\theta^T x)}{g(\theta^T x) \cdot (1 - g(\theta^T x))} \cdot \frac{d}{d\theta}(g(\theta^T x))$$

we know that $g(z) = \frac{1}{1+e^{-z}}$

$$\frac{d\,g(z)}{dz} = \frac{d}{dz}\left(\frac{1}{1+e^{-z}}\right) = \frac{d}{dz}(1+e^{-z})^{-1}$$

we know that $\frac{d}{dz}(z^n) = n(z^{n-1})$

$$\therefore \frac{d\,g(z)}{dz} = \frac{-1}{(1+e^{-z})^2} \cdot \frac{d}{dz}(1+e^{-z})$$

we know that $\frac{d}{dz} e^z = e^z$

$$\therefore \frac{d}{dz}(g(z)) = \frac{-1}{(1+e^{-z})^2} \cdot (-e^{-z}) = \frac{e^{-z}}{(1+e^{-z})^2}$$

$$= \frac{1+e^{-z}-1}{(1+e^{-z})^2} = \frac{1+e^{-z}-1}{1+e^{-z}} \cdot \left(\frac{1}{1+e^{-z}}\right)$$

$$= \left(1 - \frac{1}{1+e^{-z}}\right) \cdot \left(\frac{1}{1+e^{-z}}\right)$$

$$\therefore \frac{d}{dz}(g(z)) = g(z) \cdot (1-g(z))$$

$$\therefore \frac{d\,l(\theta)}{d\theta} = \frac{y - g(\theta^T x)}{g(\theta^T x)\cdot(1-g(\theta^T x))} \cdot \left[g(\theta^T x) \cdot (1 - g(\theta^T x)) \right] \cdot \frac{d}{d\theta}(\theta^T x)$$

$$\therefore \frac{d\,l(\theta)}{d\theta} = \left[y - g(\theta^T x) \right] \cdot x$$

How do we get θ?

- Gradient Descent!
- Then, update θ_j until convergence as follows

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

- Hence our new update rule for *logistic* regression is

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)}) \right) x_j^{(i)}$$

How do we get θ?

- Gradient Descent!
- Then, update θ_j until convergence as follows

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

- Hence our new update rule for *logistic* regression is

Looks familiar?

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$$

How do we get θ?

- Gradient Descent!
- Then, update θ_j until convergence as follows

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

- Hence our new update rule for *logistic* regression is

Looks familiar?

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)}) \right) x_j^{(i)}$$

Identical to Linear Regression Update rule!

How do we get θ?

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$$

- Although the update rule is similar it is not the same algorithm!
- We have a non-linear hypothesis function!
- Is this coincidence, or is there a deeper reason behind this?

How do we get θ?

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$$

- Although the update rule is similar it is not the same algorithm!
- We have a non-linear hypothesis function!
- Is this coincidence, or is there a deeper reason behind this?

- **Both are part of a family of models called Generalized Linear Models!**

Generalized Linear Models

- Not to be confused with the term *general linear model* (GLM).
  - It usually refers to conventional linear regression models for a continuous response variable given continuous predictors
- The form is $y_i \sim N(x_i^T\beta, \sigma^2)$, where x_i contains known covariates and β contains the coefficients to be estimated. These models are fit by least squares and weighted least squares
- The term ***generalized linear model*** (GLIM or GLM) refers to a larger class of models popularized by McCullagh and Nelder (1982, 2nd edition 1989).
  - In these models, the response variable y_i is assumed to follow an exponential family distribution with mean μ_i, which is assumed to be some (often nonlinear) function of $x_i^T\beta$.

Generalized Linear Models

- There are three components to any GLM:
- **Random Component** – refers to the probability distribution of the response variable (Y); e.g. normal distribution for *Y* in the linear regression, or binomial distribution for *Y* in the binary logistic regression. Also called a noise model or error model. How is random error added to the prediction that comes out of the link function?
- **Systematic Component** - specifies the explanatory variables ($X_1, X_2, \ldots X_k$) in the model, more specifically their linear combination in creating the so called *linear predictor*; e.g., $\beta_0+\beta_1x_1+\beta_2x_2$ as we have seen in a linear regression, or as we will see in a logistic regression in this lesson.
- ***Link Function*, η or g(μ)** - specifies the link between random and systematic components. It says how the expected value of the response relates to the linear predictor of explanatory variables; e.g., $\eta=g(E(Y_i))=E(Y_i)$ for linear regression, or $\eta=\text{logit}(\pi)$ for logistic regression.

Common Assumptions

- The data $Y_1, Y_2, \ldots, Y_n$ are independently distributed, i.e., cases are independent.
- The dependent variable Y_i does NOT need to be normally distributed, but it typically assumes a distribution from an exponential family (e.g. binomial, Poisson, multinomial, normal,...)
- GLM does NOT assume a linear relationship between the dependent variable and the independent variables, but it does assume linear relationship between the transformed response in terms of the link function and the explanatory variables; e.g., for binary logistic regression $\text{logit}(\pi)=\beta_0+\beta X$.
- Independent (explanatory) variables can be even the power terms or some other nonlinear transformations of the original independent variables.
- The homogeneity of variance does NOT need to be satisfied. In fact, it is not even possible in many cases given the model structure, and *overdispersion* (when the observed variance is larger than what the model assumes) maybe present.
- Errors need to be independent but NOT normally distributed.

Newton’s Method for Estimating Theta

Newton’s Method

- an iterative equation solver
  - Typically used to find the roots of a polynomial function
- Performs iterative updates as follows

$$\theta := \theta - \frac{f(\theta)}{f'(\theta)}$$

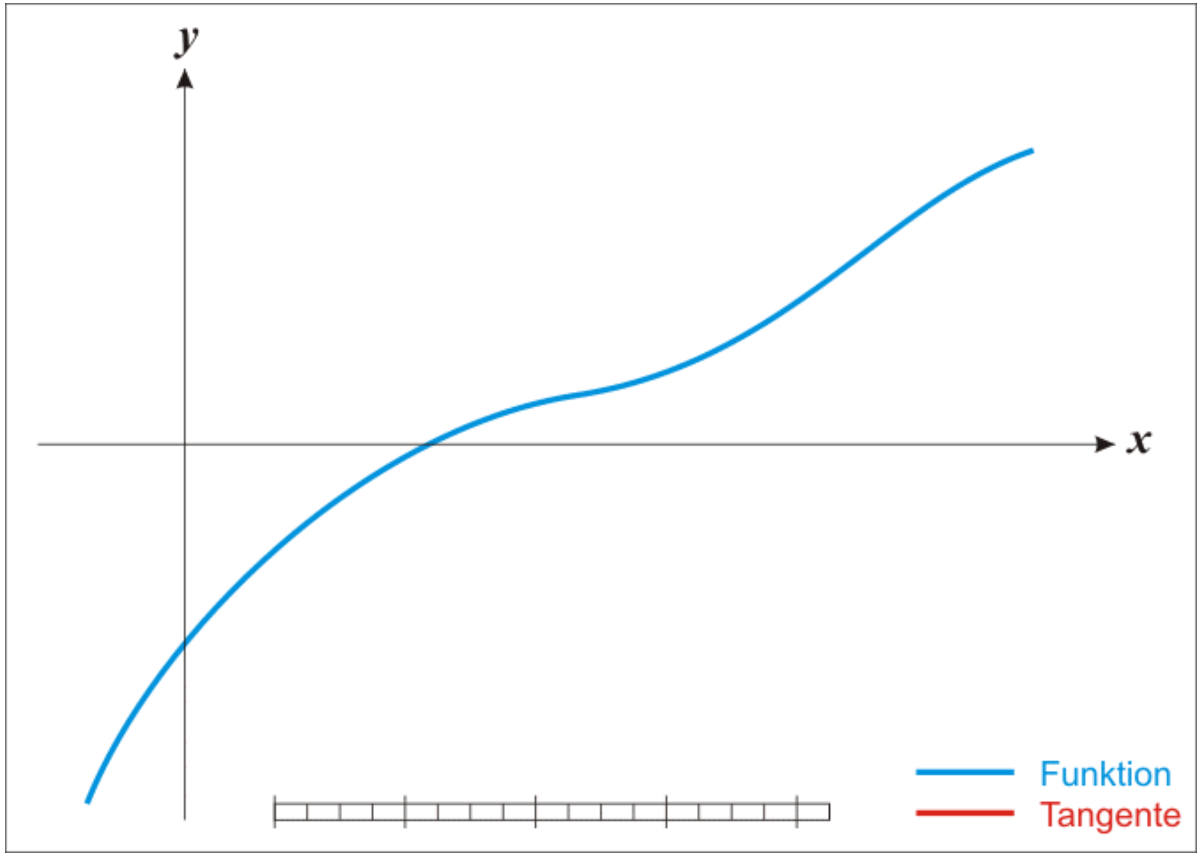
- Where $f'(\theta)$ is the derivative.

Newton’s Method

- Intuitively, we can think of it as approximating the function f via a linear function that is tangent to f at the current guess θ, solving for where that linear function equals to zero, and letting the next guess for θ be where that linear function is zero.

Newton's Method

- Intuitively, we can think of it as approximating the function f via a linear function that is tangent to f at the current guess θ, solving for where that linear function equals to zero, and letting the next guess for θ be where that linear function is zero.

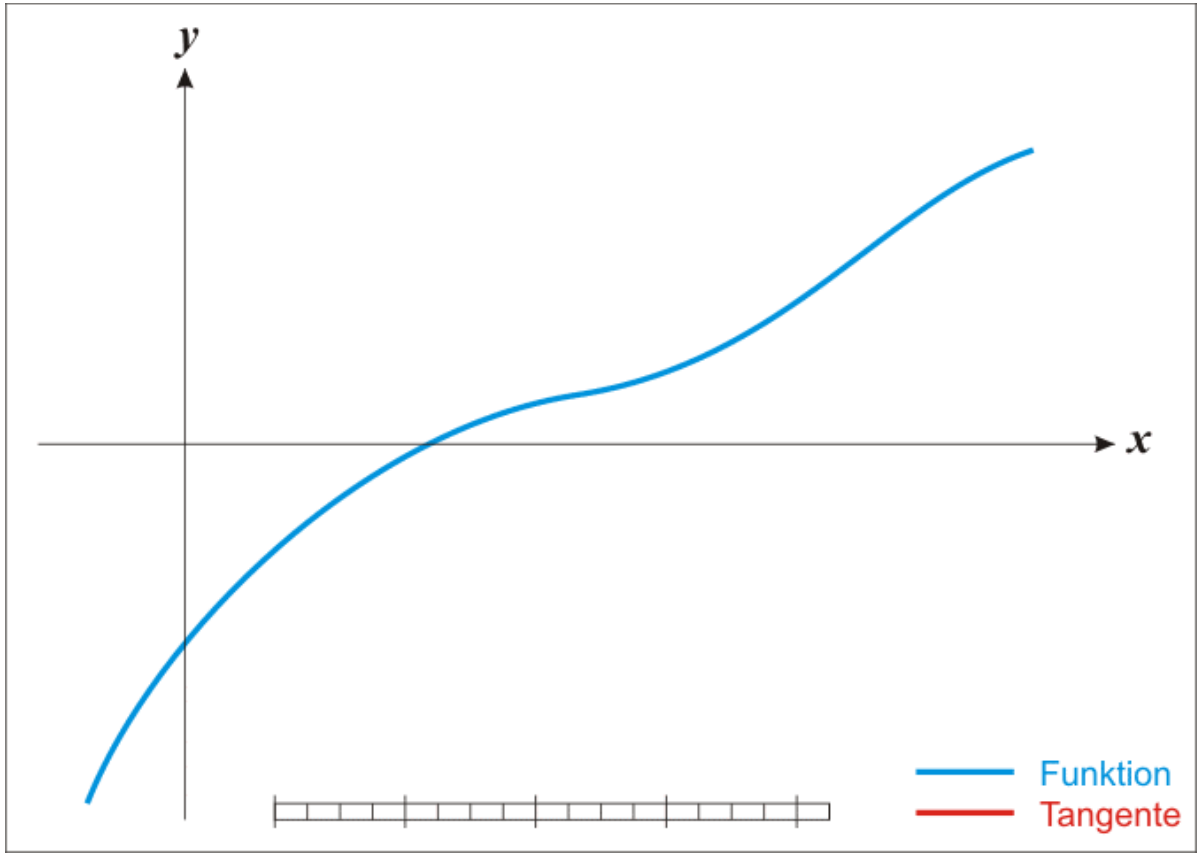


Newton's Method

- Intuitively, we can think of it as approximating the function f via a linear function that is tangent to f at the current guess θ, solving for where that linear function equals to zero, and letting the next guess for θ be where that linear function is zero.

Until $x_n - x_{n+1} \approx 0$:

- $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$



Newton's Method for LR

- In our case, we consider the log likelihood as the function for which we want to find the ideal θ
- But we want to maximize the log likelihood, whereas Newton's method gives us the point at which it is 0.
- **How do we handle this?**

Newton's Method for LR

- In our case, we consider the log likelihood as the function for which we want to find the ideal θ
- But we want to maximize the log likelihood, whereas Newton's method gives us the point at which it is 0.
- **How do we handle this?**
- Remember the maxima of ℓ correspond to points where its first derivative $\ell'(\theta)$ is zero. So, by letting $f(\theta) = \ell'(\theta)$, we can use the same algorithm to maximize ℓ, and we obtain update rule:

$$\theta := \theta - \frac{\ell'(\theta)}{\ell''(\theta)}$$

Newton's Method for LR

- θ is vector-valued (multi-dimensional), so we need to generalize Newton's method to this setting.
- The generalization of Newton's method to this multidimensional setting is called the Newton-Raphson method is given by

$$\theta := \theta - H^{-1}\nabla_\theta \ell(\theta)$$

- Where $\nabla_\theta \ell(\theta)$ is the vector of partial derivatives of $\ell(\theta)$ w.r.t. θ and H is a nxn matrix called the **Hessian**

$$H_{ij} = \frac{\partial^2 \ell(\theta)}{\partial\theta_i \partial\theta_j}$$

Newton’s Method for LR

- θ is vector-valued (multi-dimensional), so we need to generalize Newton’s method to this setting.
- The generalization of Newton’s method to this multidimensional setting is called the Newton-Raphson method is given by

$$\theta := \theta - H^{-1}\nabla_\theta \ell(\theta)$$

- Where $\nabla_\theta \ell(\theta)$ is the vector of partial derivatives of $\ell(\theta)$ w.r.t. θ and H is a nxn matrix called the **Hessian**

$$H_{ij} = \frac{\partial^2 \ell(\theta)}{\partial \theta_i \partial \theta_j}$$

The Hessian is a square matrix of second-order partial derivatives of order n x n

Newton’s Method vs Gradient Descent

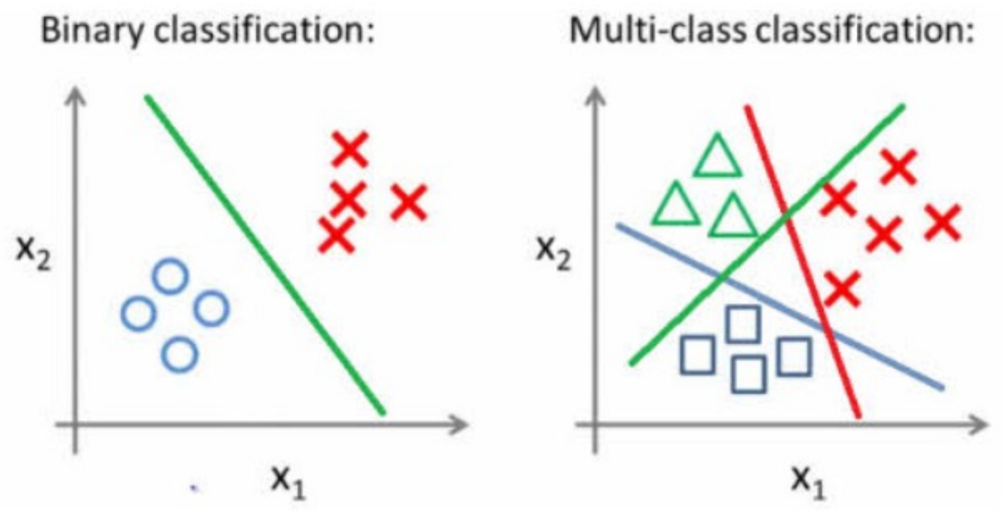
- Advantages:
  - Newton’s method typically enjoys faster convergence than (batch) gradient descent
  - It requires many fewer iterations to get very close to the minimum.
- Disadvantages:
  - One iteration of Newton’s can, however, be more expensive than one iteration of gradient descent, since it requires finding and inverting an n-by-n Hessian; but so long as n is not too large, it is usually much faster overall.
- When Newton’s method is applied to maximize the logistic regression log likelihood function $\ell(\theta)$, the resulting method is also called **Fisher scoring**.

Multiclass predictions

Extending binary classification with Logistic Regression to multiple classes

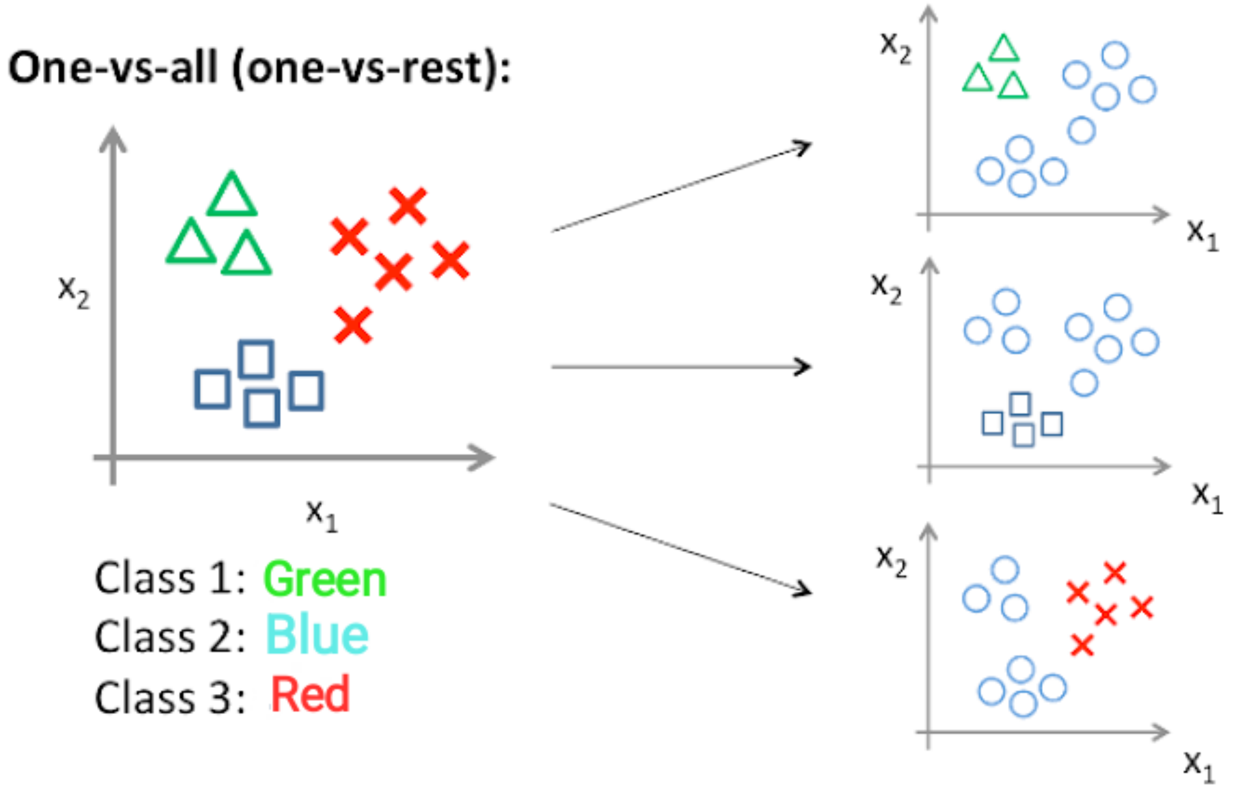
Binary vs Multi-class Classification

- More than two classes (binary) in our problem setting.
- Logistic regression returns a confidence score that scores the probability of whether an example is positive class or negative class.
- Binary classification requires only one classifier model.



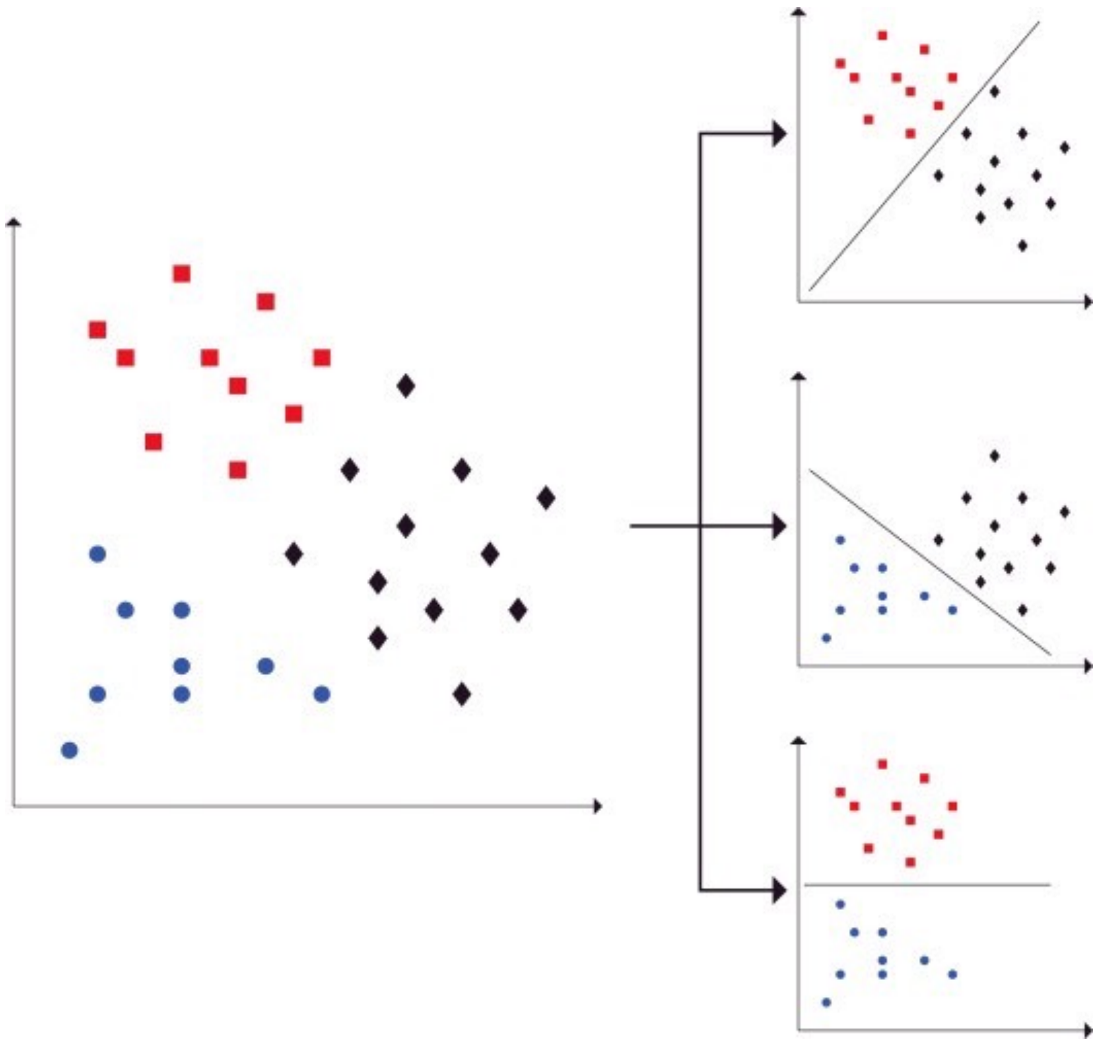
One-vs-all

- Create the N-binary classifier models, one for each class in the data.
- The number of class labels present in the dataset and the number of generated binary classifiers must be the same.
- Also called one-vs-rest classification



One-vs-one

- Create the **N* (N-1)/2** binary classifier models, one for each combination of classes in the data.
- Let us say there are three classes, A, B and C.
  - Create classifier for A-vs-B, A-vs-C, B-vs-C.
  - Each binary classifier predicts one class label.
  - Take the class with majority votes as final prediction



Perceptron Algorithm

The Perceptron

- Consider modifying the logistic regression method to “force” it to output values that are either 0 or 1 or exactly. To do so, it seems natural to change the definition of g to be the threshold function:

$$g(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

- If we then let h(x) = g(θT x) as before but using this modified definition of g, and if we use the update rule

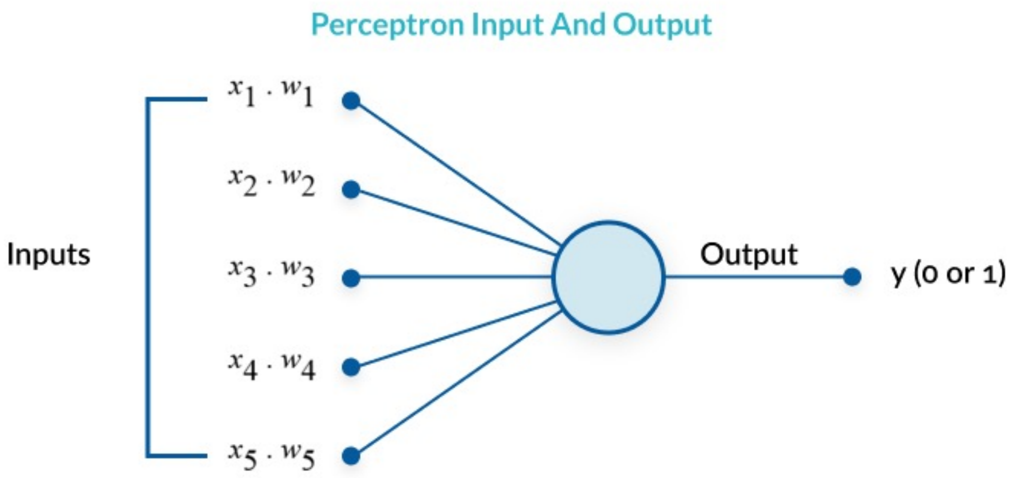
$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$$

Perceptron

- In the 1960s, this “perceptron” was argued to be a rough model for how individual neurons in the brain work.
- Note however that even though the perceptron may be cosmetically similar to the other algorithms we talked about, it is actually a very different type of algorithm than logistic regression.
  - The hypothesis in logistic regression provides a measure of uncertainty in the occurrence of a binary outcome based on a linear model.
  - The output from a step function can of course not be interpreted as any kind of probability.
  - Since a step function is not differentiable, it is not possible to train a perceptron using the same algorithms that are used for logistic regression.

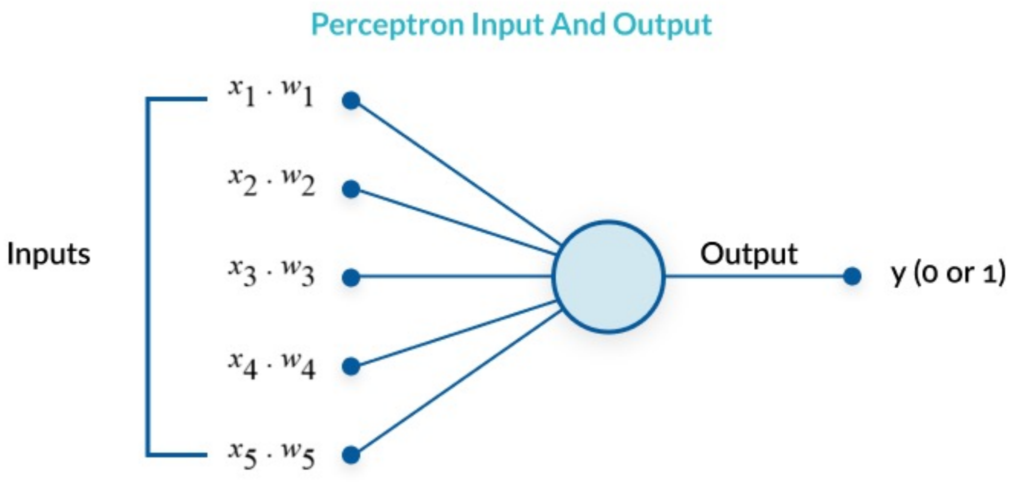
Multi-layer perceptron

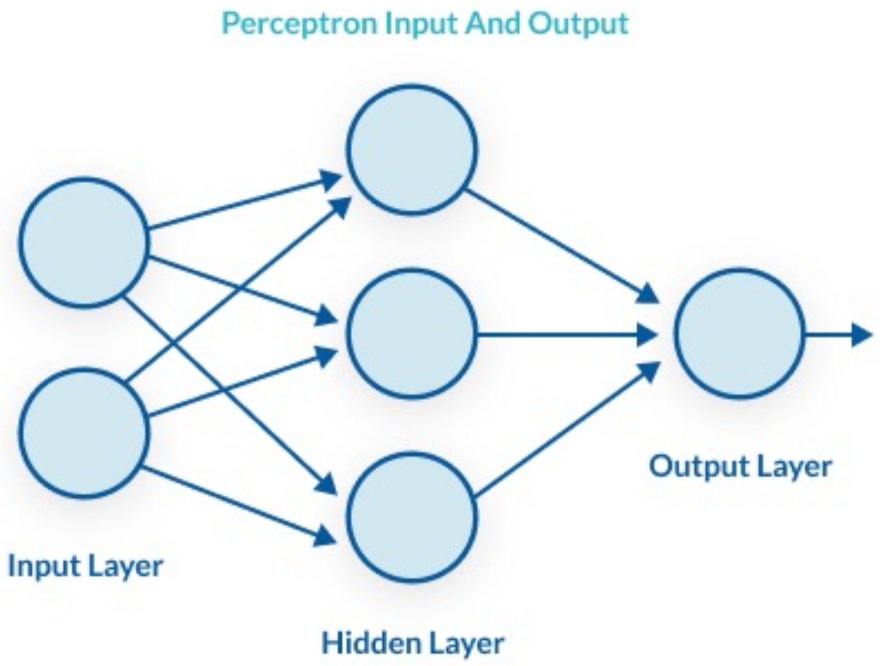
Perceptron vs Multilayer Perceptron (MLP)




THAT'S NOT ENOUGH
WE HAVE TO GO DEEPER

Perceptron vs Multilayer Perceptron (MLP)





Why do we need more layers?