Linear Regression

Input : Vector of feature measured by real number

Output : single scalar real number

Example : Predicting weight based on height

Input : height of one individual

Output : weight of the same individual

Data set

height	weight
180	86
162	55
183	86.5
174	70
160	62
163	54
180	60
165	72
175	93
170	89
170	60
169	82
168	59
175	75
169	56
171	89
155	45
158	60
175	60
165	72

Goal : Predict weight given height

Hypothesis : A specific predictor

$\mathcal{X}$: Domain of our hypothesis function (input) $\mathbb{R}$ (more input e.g. 2 : $\mathbb{R}^2$)

$\mathcal{Y}$: Range of our hypothesis function (output) $\mathbb{R}$

each hypothesis is a function

$h : \mathbb{R} \to \mathbb{R}$

What is the form of h?

$h(x) = \theta_0 + \theta_1 x_1$

($h(x)$: predicted weight; θ_1: parameter of h (weights); x_1: height)

(x, y) - one training example

$(x^{(i)}, y^{(i)})$ - i^{th} training example

$x^{(1)} = 180$

$x^{(2)} = 162$

$y^{(1)} = 86$

$\theta^T = [\theta_0\ \theta_1 \cdots \theta_n]$ $x = \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix}$

What if we have n inputs ? n+1 parameters

$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$

$x = \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix} \to \begin{bmatrix} 1 \\ x_1 \\ \cdots \\ x_n \end{bmatrix}$ $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \cdots \\ \theta_n \end{bmatrix}$

$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \cdots \\ \theta_n \end{bmatrix}$ $x = \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix}$

$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x$ (dot product)

multivariate linear regression

$= \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$ ($x_0 = 1$)

$h(x) = \theta_0 + \theta_1 x_1$ - Linear equation

$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$ $x = \begin{bmatrix} 1 \\ x \end{bmatrix}$

Hypothesis : Output (y) is a linear function of the inputs (x)

How to find the best possible θ ?

Cost function : A function of θ that tell us how BAD (GOOD) θ is

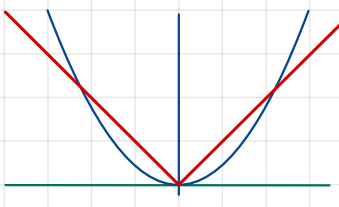
squared error function

Regression problem normally use the Least Square Cost function (J)

$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$ (m: # of training)

- $h_\theta(x^{(i)})$: our hypothesis function with specific value for θ
- $x^{(i)}$: input feature of i_th example
- $y^{(i)}$: observed target of i_th example
- $(\ldots)^2$: convert negative error to positive cost

$h(x^{(i)}) = \theta_0 + \theta_1 x_1^{(i)}$ minimize $J(\theta_0, \theta_1)$ over θ_0, θ_1



$(h_\theta(x^{(i)}) - y^{(i)})^2$

$|h_\theta(x^{(i)}) - y^{(i)}|$

$h_\theta(x^{(i)}) - y^{(i)}$

both satisfy the requirement of being minimum when prediction is perfect and larger when error is larger

So far, we have

1. Form of the hypothesis function
2. Cost function
3. Form of dataset

Still need :

1. An algorithm for minimizing the cost function

To find the minimum of J :

$$\theta^* = \arg\min_\theta J(\theta) \qquad \nabla_\theta J(\theta)$$

$J(\theta)$ is minimized by $\frac{\partial J}{\partial \theta_0} = 0$, $\frac{\partial J}{\partial \theta_1} = 0$

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$\frac{\partial u^2}{\partial x} = 2u \cdot u'$

$\nabla_\theta J(\theta_0)$

$$\frac{\partial J}{\partial \theta_0} = \frac{1}{2}\sum_{i=1}^{m} 2(h_\theta(x^{(i)}) - y^{(i)}) \cdot \frac{\partial(\theta_0 + \theta_1 x_1)}{\partial \theta_0}$$

$\nabla_\theta J(\theta_1)$

$$\frac{\partial J}{\partial \theta_1} = \frac{1}{2}\sum_{i=1}^{m} 2(h_\theta(x^{(i)}) - y^{(i)}) \cdot \frac{\partial(\theta_0 + \theta_1 x_1)}{\partial \theta_1}$$

$$0 = \sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot 1 \qquad x_0^{(i)}$$

$$0 = \sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_1^{(i)}$$

substitute $\hat{y}^{(i)}$ for $h_\theta(x^{(i)})$

$$\sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)}) = 0$$

$$\sum_{i=1}^{m}(\hat{y}^{(i)} x_1^{(i)} - y^{(i)} x_1^{(i)}) = 0$$

$$\sum_{i=1}^{m}\hat{y}^{(i)} - \sum_{i=1}^{m} y^{(i)} = 0$$

$$\sum_{i=1}^{m}\hat{y}^{(i)} x_1^{(i)} - \sum_{i=1}^{m} y^{(i)} x_1^{(i)} = 0$$

so $\sum_{i=1}^{m}\hat{y}^{(i)} = \sum_{i=1}^{m} y^{(i)}$, $\sum_{i=1}^{m}\hat{y}^{(i)} x_1^{(i)} = \sum_{i=1}^{m} y^{(i)} x_1^{(i)}$

$$\hat{y}^{(i)} = \theta_0 x_0^{(i)} + \theta_1 x_1^{(i)}$$

$$\nabla_J(\theta) = \frac{\partial J(\theta)}{\partial \theta} = \sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 x_J^{(i)}$$

$$\sum_{i=1}^{m}\hat{y}^{(i)} = \sum_{i=1}^{m} y^{(i)} \; ; \; \hat{y}^{(i)} = \theta_0 x_0^{(i)} + \theta_1 x_1^{(i)}$$

$$\sum_{i=1}^{m}\theta_0 x_0^{(i)} x_0^{(i)} + \sum_{i=1}^{m}\theta_1 x_1^{(i)} x_0^{(i)} = \sum_{i=1}^{m} y^{(i)} x_0^{(i)}$$

$$\theta_0 \sum_{i=1}^{m} x_0^{(i)} x_0^{(i)} + \theta_1 \sum_{i=1}^{m} x_1^{(i)} x_0^{(i)} = \sum_{i=1}^{m} y^{(i)} x_0^{(i)}$$

$$\sum_{i=1}^{m}\hat{y}^{(i)} x_1^{(i)} = \sum_{i=1}^{m} y^{(i)} x_1^{(i)}$$

$$\sum_{i=1}^{m}\theta_0 x_0^{(i)} x_1^{(i)} + \sum_{i=1}^{m}\theta_1 x_1^{(i)} x_1^{(i)} = \sum_{i=1}^{m} y^{(i)} x_1^{(i)}$$

$$\theta_0 \sum_{i=1}^{m} x_0^{(i)} x_1^{(i)} + \theta_1 \sum_{i=1}^{m} x_1^{(i)} x_1^{(i)} = \sum_{i=1}^{m} y^{(i)} x_1^{(i)}$$

$X^T y$

$$X = \begin{bmatrix} x_0^{(1)} & x_1^{(1)} \\ x_0^{(2)} & x_1^{(2)} \\ x_0^{(3)} & x_1^{(3)} \\ \dots & \dots \\ x_0^{(m)} & x_1^{(m)} \end{bmatrix} \qquad y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ \dots \\ y^{(m)} \end{bmatrix}$$

$$X^T X = \begin{bmatrix} \sum_{i=1}^{m} x_0^{(i)} x_0^{(i)} & \sum_{i=1}^{m} x_0^{(i)} x_1^{(i)} \\ \sum_{i=1}^{m} x_1^{(i)} x_0^{(i)} & \sum_{i=1}^{m} x_1^{(i)} x_1^{(i)} \end{bmatrix} \qquad X^T y = \begin{bmatrix} \sum_{i=1}^{m} x_0^{(i)} y^{(i)} \\ \sum_{i=1}^{m} x_1^{(i)} y^{(i)} \end{bmatrix}$$

$$X^T X \cdot \theta = X^T y$$

$$\begin{bmatrix} \sum_{i=1}^{m} x_0^{(i)} x_0^{(i)} & \sum_{i=1}^{m} x_0^{(i)} x_1^{(i)} \\ \sum_{i=1}^{m} x_1^{(i)} x_0^{(i)} & \sum_{i=1}^{m} x_1^{(i)} x_1^{(i)} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{m} x_0^{(i)} y^{(i)} \\ \sum_{i=1}^{m} x_1^{(i)} y^{(i)} \end{bmatrix}$$

$$X^T X \cdot \theta = X^T y \rightarrow \theta = (X^T X)^{-1} X^T y \quad \text{Normal equation}$$

if $X^T X$ has a matrix inverse!

$$X = \begin{bmatrix} 1 & x_1^{(1)} \\ 1 & x_1^{(2)} \\ 1 & x_1^{(3)} \\ \dots & \dots \\ 1 & x_1^{(m)} \end{bmatrix} \qquad \begin{matrix} x_0 & x_1 \end{matrix}$$

m examples, $n = 1$

Exercise : Form matrices X, y for our weight-height dataset and find optimal values of θ_0, θ_1 Before tomorrow !

Last time → sample machine learning problem (predict weight given height)

the 4 things you need for a machine learning

1. Dataset & its structure (X, y)
2. Form of the hypothesis function $h : X \to Y$ (parameterized by θ)
3. Cost function
4. An algorithm for minimizing the cost function

4 methods: Analytical approach
Batch Gradient Descent
Stochastic Gradient Descent
Mini-Batch Gradient Descent

use one training example at a time to update θ → faster than batch but noiser

Why squared error? $(h_\theta(x^{(i)}) - y^{(i)})^2$?

- we only have to minimize
- smooth & nice
- analytical optimum $\theta^* = (x^T x)^{-1} x^T y$
- can be derived from probabilistic point of view

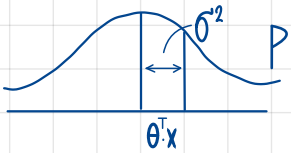
New Cost function (Objective function): Likelihood

Generative model for the provenance of our dataset (original)

1. sample x i.i.d (independent & identically distributed) from some unknown distribution over X

$$x \sim P(x)$$

2. sample y from a gaussian distribution with mean of $\theta^T x$, variance of σ^2

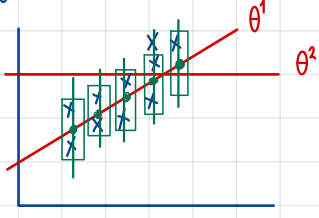


$$P(Y = y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(y-\theta^T x)^2}{2\sigma^2}}$$

$$y \sim N(\theta^T x, \sigma^2)$$ normal distribution (or gaussian)

standardization of a normal distribution

Given a dataset we will measure the goodness or quality of a particular value of θ in terms of how consistent the data are with a gaussian distribution with mean $\theta^T x$ and variance σ^2



Linear Regression

$x : \mathbb{R}$, $y : \mathbb{R}$ after augmenting x with dummy variable 1 (add)

$h(x) : x \to \theta^T x$

$$J(\theta; X, y) = \frac{1}{2} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, J(\theta) = (x^T x)^{-1} x^T y$$ (normal equation)

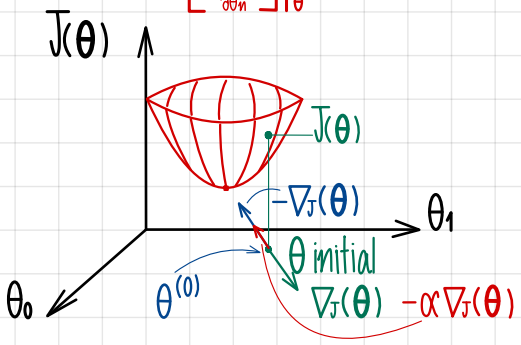
Alternative optimization algorithm:

Gradient Descent

1. pick an initial θ
2. calculate gradient

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

$$\nabla_J(\theta) = \begin{bmatrix} \frac{\partial J}{\partial \theta_0} \\ \frac{\partial J}{\partial \theta_1} \\ \vdots \\ \frac{\partial J}{\partial \theta_n} \end{bmatrix}_\theta$$



$$\theta^{(1)} \leftarrow \theta^{(0)} - \alpha \nabla_J(\theta^{(0)})$$ (learning rate)

3. $\theta^{(l+1)} \leftarrow \theta^{(l)} - \alpha \nabla_J(\theta^{(l)})$

$$\theta^{(l+1)} \leftarrow \theta^{(l)} - \alpha \sum_{i=1}^{m} (h_{\theta^{(l)}}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

4. Repeat from #2 untill convence

Likelihood of θ :

$$L(\theta; X, y) = P(X, y; \theta)$$

outcomes of a random experiment

parameters — contextual data / constants used in the function — parameters of the distribution

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, L(\theta; X, y)$$

$$= \underset{\theta}{\operatorname{argmin}} - L(\theta; X, y)$$

Maximum Likelihood Approach

Choose θ to maximized $\ell(\theta)$

New Objective function : Likelihood of θ function of θ parameter $\quad \ell(\theta) = \log L(\theta)$

$$\theta^* = \arg\max_\theta L(\theta; \mathbf{x}, \mathbf{y})$$

$$L(\theta; X, y) = P(X, y; \theta) \qquad X \; y$$

$$= P(x^{(1)}, x^{(2)}, \dots, x^{(2)}, y^{(1)}, y^{(2)}, \dots, y^{(2)}; \theta)$$

$P(A, B) \overset{\text{sometimes}}{=} P(A) \cdot P(B)$ if A, B mutually independent

Example : two flips of a fair coin

$P(H, H) = P(H)\,P(H) = 0.5 \times 0.5 = 0.25$

since $(x^{(i)}, y^{(i)})$, $(x^{(j)}, y^{(j)})$, $i \neq j$ are sampled i.i.d. from the same distribution, we have

$$P(x^{(i)}, x^{(j)}, y^{(i)}, y^{(j)}) = P(x^{(i)}, y^{(i)}) \cdot P(x^{(j)}, y^{(j)})$$

$$P(X, y; \theta) = \prod_{i=1}^{m} P(x^{(i)}, y^{(i)}; \theta)$$ ← change form

Are $x^{(i)}$ and $y^{(i)}$ independent? Nope

$$P(x^{(i)}, y^{(i)}) \neq P(x^{(i)}) \cdot P(y^{(i)})$$

factorization of joint probability

$$P(A|B) = \frac{P(A,B)}{P(A)} \rightarrow P(A, B) = P(A, B)\,P(A)$$

$$P(x^{(i)}, y^{(i)}) = P(y^{(i)}|x^{(i)}) \cdot P(x^{(i)})$$

$$L(\theta; X, y) = \left[\prod_{i=1}^{m} P(y^{(i)}|x^{(i)}; \theta)\right]\left[\prod_{i=1}^{m} P(x^{(i)}; \theta)\right]$$

by assumption of our generative model for linear regression

$y^{(i)} \sim N(\theta^T x^{(i)}, \sigma^2)$

$$P(x^{(i)}|y^{(i)}; \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}}$$

$$L(\theta; X, y) = \left[\prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}}\right]\left[\prod_{i=1}^{m} P(x^{(i)}; \theta)\right]$$

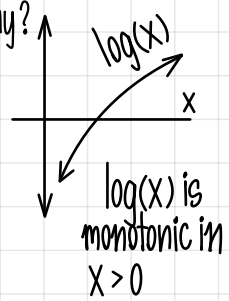
is maximized by θ maximizing

$$\prod_{i=1}^{m} e^{\frac{-(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}}$$

is maximized by θ maximizing

$$\log \prod_{i=1}^{m} e^{\frac{-(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}}$$

why?



ever negative? X

$e^0 = 1$

$e^{-100} > 0$

$\log(a \cdot b) = \log(a) + \log(b)$

$$\log \prod_{i=1}^{m} e^{\frac{-(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}} = \sum_{i=1}^{m} \log e^{\frac{-(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}} = \sum_{i=1}^{m} \frac{-(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}$$

$$\ell(\theta) = -\frac{1}{2\sigma^2} \sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)})^2$$

is maximized by θ maximizing

$$-\frac{1}{2} \sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)})^2$$

is maximized by θ minimizing

$$-\frac{1}{2} \sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)})^2 = \frac{1}{2} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

$h_\theta(x^{(i)}) = \theta^T x^{(i)}$

$$= J(\theta)$$

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)})^2$$

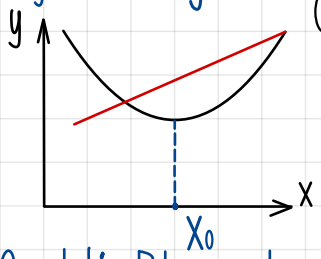
Maximizing the Likelihood of θ under the gaussian Generative model give us the least square cost function (J) for Linear Regression !!!

Squared error cost function

1. make sense !
2. give us an analytical solution to the optimization problem (convenient)
3. maximizes the Likelihood of θ (give us the θ most likely to have been used nature to generate with dataset $\mathbf{x}, y$.)

This approach is called the Generative approach to defining ML algorithms

Polynomial Regression



Quadatic (parabola)

linear

$y = (x - x_0)^2 + y_0 + \varepsilon$ (noise)

$\varepsilon \sim N(0, \sigma^2)$

$y = ax^2 + bx + c + \varepsilon$ (θ_2, θ_1, θ_0)

Quadatic Polynomial

in two features: (x_0, x_1)

$y = ax_0^2 + bx_1^2 + cx_0x_1 + dx_0 + ex_1 + f + \varepsilon$

$\varepsilon \sim N(0, \sigma^2)$

$h_\theta(x) = \theta^T x$

linear in parameter $\begin{bmatrix}\theta_0\\ \theta_1\\ \theta_2\end{bmatrix}$ $\begin{bmatrix}1\\ x\\ x^2\end{bmatrix}$ nonlinear in input features

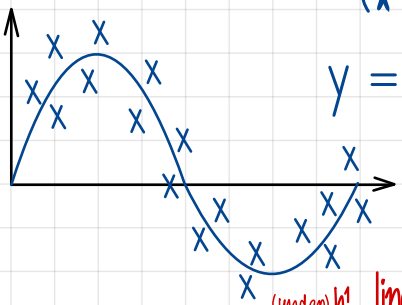
new features are nonlinear combinations original features

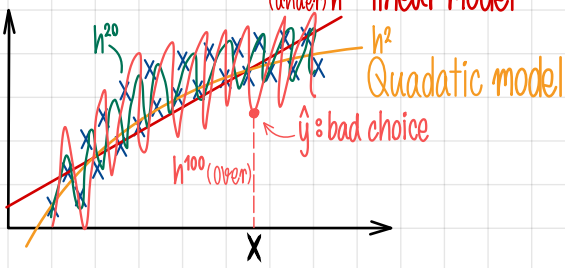
$$X = \begin{bmatrix} 1 & x_0^{(1)}x_0^{(1)} & x_1^{(1)}x_1^{(1)} & x_0^{(1)}x_1^{(1)} & x_0^{(1)} & x_1^{(1)} \\ \cdots & & & & & \\ 1 & x_0^{(m)}x_0^{(m)} & x_1^{(m)}x_1^{(m)} & x_0^{(m)}x_1^{(m)} & x_0^{(m)} & x_1^{(m)} \end{bmatrix}$$

$$Y = \begin{bmatrix} y^{(1)} \\ \cdots \\ y^{(m)} \end{bmatrix}$$

$$\theta^* = \arg\min_\theta J(\theta) = \arg\max_\theta \ell(\theta) = (X^T X)^{-1} X^T y$$

$y = \theta_0 + \theta_1 \cos x + \varepsilon$





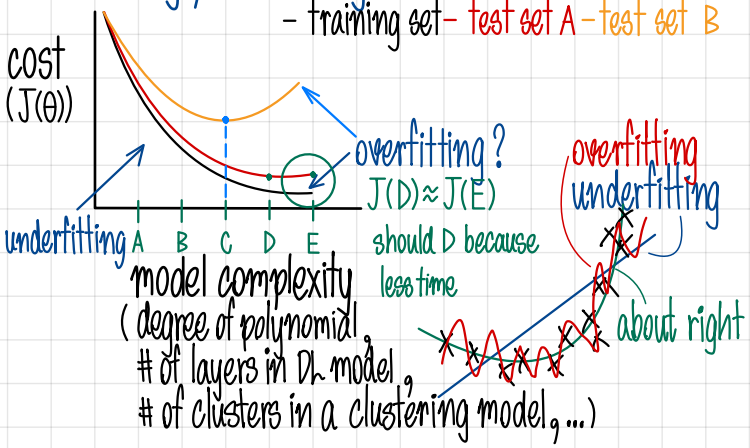
Possible nonlinear relationship between x, y

Q: when to stop adding parameters ???

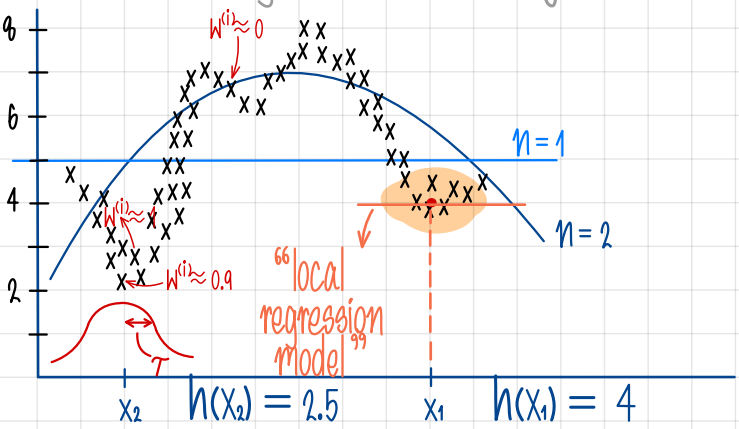
Overfitting: too many parameters

Underfitting: too few parameters

Underfitting, Overfitting



consider Locally Weight Linear Regression



$$\theta^* = \arg\min_\theta \sum_{i=1}^{m} w^{(i)} (h_\theta(x^{(i)}) - y^{(i)})^2$$

0 for the points I want to use

1 for the points I want to neglect

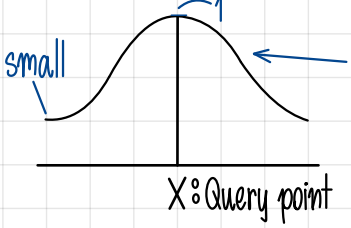
Idea: at this time, when asked what h(x) should be for a specific x, we do local linear regression, then answer with the prediction based on that local linear regression result.

How can we perform a linear regression using a subset of the training data

How can we set the weights $w^{(i)}$ to appropriate values?

"soft" weights are also possible: big for points $x^{(i)}$ close to x and small for far from x

Gaussian kernel for $w^{(i)}$:



$$w^{(i)} = e^{-\frac{(x^{(i)} - x)^T (x^{(i)} - x)}{2\tau^2}}$$

location x

τ "bandwidth" parameter tau

std. deviation if this was a probability distribution

If $x^{(i)}$ is on top of x (small), $w^{(i)} \approx 1$

If $x^{(i)} - x$ is large (negative of positive directional), $w^{(i)} \approx 0$

Classification (Binary Classification)

$y = \{0, 1\}$ 0 : negative, 1 : positive

$x = \mathbb{R}^n$ → not always easy !!! (fixed length vector)

Documents : term frequencies

List of credit card transaction → need to aggregate variable length input into a "summary"

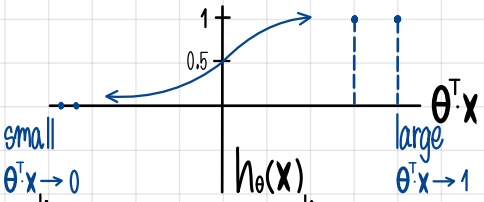
Logistic Regression

$X_1, X_2, \dots, X_n \rightarrow = \theta_0 X_0 + \theta_1 X_1 + \theta_2 X_2 + \dots + \theta_n X_n$ (g)

but! range must be limited to [0...1]

value can be threshold

($> 0.5 \rightarrow 1$, $\leq 0.5 \rightarrow 0$)



logistic sigmoid function

$\lim_{z \to \infty} g(z) = 1$ $\lim_{z \to -\infty} g(z) = 0$

$h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$ still have parameter $\theta_0, \theta_1, \theta_2, \dots, \theta_n$ how to learn them?

$g(0) = \frac{1}{1+e^{-0}} = \frac{1}{2}$

$g(1000) = \frac{1}{1+e^{-1000}} = 1$ $g(-1000) = \frac{1}{1+e^{+1000}} = 0$

can we use squared error?

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Yes, but negative log likelihood works better

$L(\theta) = P(y \mid X; \theta)$ → independence of training set items

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

so what is $P(y \mid x; \theta)$

Assume

$$P(y \mid x; \theta) = \begin{cases} h_\theta(x) & , y = 1 \\ 1 - h_\theta(x) & , y = 0 \end{cases}$$

Example $x^{(i)} = \begin{bmatrix} 1.2 \\ 3.1 \\ 0.5 \end{bmatrix}$ $y^{(i)} = 0$ ← target

$$h_\theta(x^{(i)}) = \frac{1}{1+e^{-[\theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \theta_3 x_3^{(i)}]}}$$

Suppose $h_\theta(x^{(i)}) = 0.9$ (bad pred) $P(y \mid x; \theta) = 1 - h_\theta(x)$

Prediction is it good or bad?

Bad! but the model says $P(y^{(i)} = 0 \mid x^{(i)}; \theta) = 0.1$

According to the model, the probability of the correct prediction is small

Maximum likelihood says "let's make the model's predicted probability for the correct label as high as possible"

we will try to maximize

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

which is maximized by θ maximizing

$$\log P(y \mid X; \theta) = \log \prod_{i=1}^{m} P(y^{(i)} \mid x^{(i)}; \theta)$$

$$\log L(\theta) = \sum_{i=1}^{m} \log P(y^{(i)} \mid x^{(i)}; \theta)$$

How to maximize this when we have a case statement??

In general, if we have

$$A = \begin{cases} B & \text{if } C = 0 \\ D & \text{if } C = 1 \end{cases} \quad \therefore A = B^{(1-C)} D^C$$

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

Therefore, θ maximizing $L(\theta)$ maximizes

$$\ell(\theta) = \log L(\theta) = \sum_{i=1}^{m} \log P(y^{(i)} \mid x^{(i)}; \theta)$$

$$= \sum_{i=1}^{m} [\log h_\theta(x^{(i)})^{y^{(i)}} + \log(1 - h_\theta(x^{(i)}))^{(1-y^{(i)})}]$$

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

Let $\hat{y}^{(i)} = h_\theta(x^{(i)})$ ← $\log x^y = y \log x$

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

which is maximized by θ minimizing

$$J(\theta) = -\sum_{i=1}^{m} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

Binary Cross Entropy Cost Function

$$\hat{y}^{(i)} = \frac{1}{1+e^{-\theta^T x^{(i)}}}$$

what is the derivative of sigmoid function?

$$\frac{d}{dz}\frac{f}{g}=\frac{gf'-fg'}{g^2} \qquad \frac{d}{dz}(1+e^{-z})=-e^{-z}$$

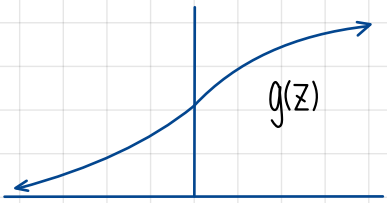
$$g(z)'=\frac{d}{dz}\frac{-1}{(1+e^{-z})}$$

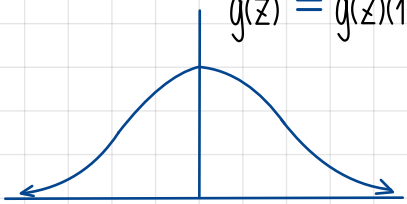
$$=\frac{(1+e^{-z})\cdot 0-1\cdot(-e^{-z})}{(1+e^{-z})^2}$$

$$=\frac{e^{-z}}{(1+e^{-z})^2} \qquad \frac{1+e^{-z}}{(1+e^{-z})}-\frac{1}{(1+e^{-z})}$$

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

$$g(z)'=g(z)(1-g(z))$$





$$\hat{y}^{(i)}=\frac{1}{1+e^{-\theta^T x^{(i)}}}$$

$$\frac{d}{dx}e^y=e^y\frac{dy}{dx}$$

$$\frac{d}{dx}\log y=\frac{1}{y}\frac{dy}{dx}$$

$$\nabla_J(\theta)=\frac{\partial}{\partial\theta_j}J(\theta)$$

$$=\frac{\partial}{\partial\theta_j}\left[-\sum_{i=1}^{m}y^{(i)}\log\hat{y}^{(i)}+(1-y^{(i)})\log(1-\hat{y}^{(i)})\right]$$

$$=\frac{\partial}{\partial\theta_j}\left[y\log\hat{y}+(1-y)\log(1-\hat{y})\right]$$

$$\frac{\partial}{\partial\theta_j}\frac{1}{1+e^{-\theta^T x^{(i)}}}=\frac{(1+e^{-\theta^T x})'1-1'(1+e^{-\theta^T x})}{(1+e^{-\theta^T x})^2}=\frac{e^{-\theta^T x}}{(1+e^{-\theta^T x})^2}(x_j)$$

$$=\frac{1+e^{-\theta^T x}-1}{(1+e^{-\theta^T x})(1+e^{-\theta^T x})}(x_j)=\frac{1}{(1+e^{-\theta^T x})}\left(\frac{(1+e^{-\theta^T x})}{(1+e^{-\theta^T x})}-\frac{1}{(1+e^{-\theta^T x})}\right)(x_j)$$

$$=\hat{y}(1-\hat{y})(x_j)$$

$$=y\frac{1}{\hat{y}}\hat{y}(1-\hat{y})(x_j)+(1-y)\frac{1}{1-\hat{y}}(-\hat{y}(1-\hat{y}))(x_j)$$

$$=y(1-\hat{y})(x_j)+(1-y)(-\hat{y})(x_j)$$

$$=(y-y\hat{y}-\hat{y}+y\hat{y})x_j=(y-\hat{y})x_j$$

$$\nabla_J(\theta)=(y-h_\theta(x))x_j$$

Stochastic Gradient Descent

$$\nabla\ell(\theta)=\begin{bmatrix}\frac{\partial J_\theta}{\partial\theta_0}\\ \frac{\partial J_\theta}{\partial\theta_1}\\ \vdots\\ \frac{\partial J_\theta}{\partial\theta_n}\end{bmatrix}=(y-h_\theta(x))\mathbf{x}$$

↘ Ascent

Batch Gradient Descent

$$\nabla\ell(\theta)=\sum_{i=1}^{m}(y^{(i)}-h_\theta(\mathbf{x}^{(i)}))\mathbf{x}^{(i)}$$

$$\nabla_-\ell(\theta)=\sum_{i=1}^{m}(h_\theta(\mathbf{x}^{(i)})-y^{(i)})\mathbf{x}^{(i)}$$

Gradient Descent Role (Batch)

$\frac{\partial\ell(\theta)}{\partial\theta_J}$

$$\theta^{i-1}\leftarrow\theta^i-\alpha\sum_{i=1}^{m}(h_\theta(\mathbf{x}^{(i)})-y^{(i)})\mathbf{x}^{(i)}$$

Gradient Descent Role (Stochastic)

$$\theta^{i-1}\leftarrow\theta^i-\alpha(h_\theta(\mathbf{x}^{(i)})-y^{(i)})\mathbf{x}^{(i)}$$

(update with one pattern)

Classification

1. Dataset $x=\mathbb{R}^n \quad y=\{0,1\}$
2. Hypothesis function $h_\theta(\mathbf{x})=\frac{1}{1+e^{-\theta^T x}}$
3. Cost function $J(\theta)=-\sum_{i=1}^{m}y^{(i)}\log\hat{y}^{(i)}+(1-y^{(i)})\log(1-\hat{y}^{(i)})$
4. Optimization algorithm

$\theta^{(0)}$: initialize arbitarily

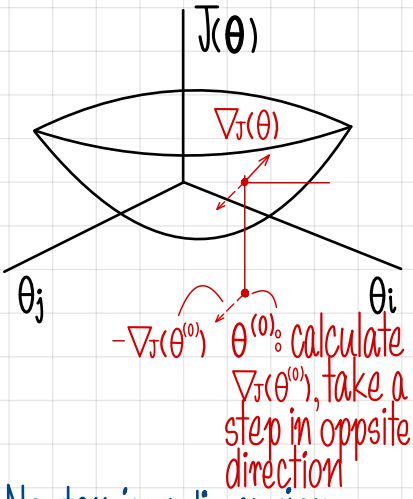
while not converged

$$\theta^{n-1}\leftarrow\theta^n-\alpha\nabla_J(\theta^{(j)})$$

J is convex → Gradient Descent always converges if α small enough

Newton's method

Gradient Descent is the first order method



Issues:
(1) learning rate difficult to find, especially when no normalization
(2) How many iteration

the second order method can use the shape of J( ) at current θ to decide what step

Calculate Gradient: $\theta^{(1)} \leftarrow \theta^{(0)} - \frac{J'(\theta^{(0)})}{J''(\theta^{(0)})}$

$\nabla J(\theta) = J'(\theta) = 2\theta - 6$

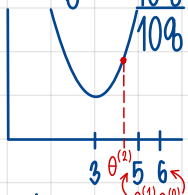
don't use yet: $2\theta - 6 = 0$, $\theta = 3$

$J'(6) = 6$

$J''(\theta) = 2$

$= 6 - \frac{6}{2}$

$= 3$



$= 6 - \frac{10\frac{2}{3}}{16} = 5$

$\theta^{(2)} = 4\frac{1}{3}$

$\theta^{(10)} = 3.1$

$J'(4\frac{1}{3}) = 9.48$

$J''(4\frac{1}{3}) = 21.3$

$J'(\theta^{(10)}) = 0.004$

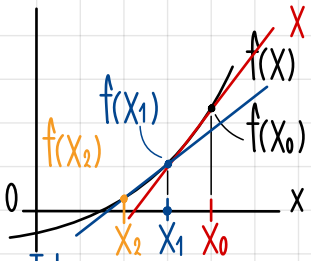
$J''(\theta^{(10)}) = 0.12$

$\theta^{(3)} = 4\frac{1}{3} - \frac{9.48}{21.3} = 3.99$

$\theta^{(11)} = 3.067$

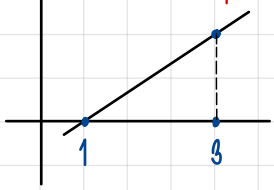
Newton in 1 dimension

zero-finding algorithm
given some complex function, we want to find where it crosses 0



How to calculate X_{i+1}

$X_{i+1} = X_i - \frac{f(X_i)}{f'(X_i)}$ ← current value; ← slope of f( ) at x_i

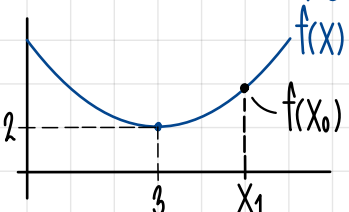


Idea:
(1) select X_0 arbitrarily
(2) let i = 0
(3) calculate $f(X_i)$
(4) linearized f(x) around X_i
(5) find X_{i+1} as zero of linearized f(x)
(6) let $i \leftarrow i+1$
(7) go to step (3)

random

$y = 1/2X - 1/2$
want x such that y is 0
$X = 3$ $y = 1$
adjust x By height divided by the slope
$X_0 = 3$ $f(X_0) = 1$
$X_1 = 3$ $f(X_1) = 1$

Newton's method find a zero.
In optimization, we don't want a zero of the Cost function. We want the minimum!



we can use Newton's method to find a zero of the Gradient of J

find zero of $\rightarrow \nabla J(\theta)$ this function

$J(\theta) = (\theta-3)^2+2 = \theta^2-6\theta+9+2 = \theta^2-6\theta+11$

start at $\theta^{(0)} = 6$

use Newton to find $\theta^{(1)}$

$J(\theta) = (\theta-3)^4+2$ (min is $\theta=3$)

$J'(\theta) = 4(\theta-3)^3$

$J''(\theta) = 12(\theta-3)^2$

$J'(\theta^{(0)}) = 108$

$J''(\theta^{(0)}) = 108$

So Newton's method can be used to find the minimum of cost function.
Quickly if the function is simple & smooth

In some cases, it may divercge.

Issue: method for 1-dimension function.
What about a (n+1)-dimension function.

vector version of the first derivative is the Gradient. find zero of $\nabla J(\theta)$

vector version of the second derivative is the Hession.

$$\nabla J(\theta) = \begin{bmatrix} \frac{\partial J}{\partial \theta_0} \\ \frac{\partial J}{\partial \theta_1} \\ \vdots \\ \frac{\partial J}{\partial \theta_n} \end{bmatrix} \quad H_J(\theta) = \begin{bmatrix} \frac{\partial^2 J}{\partial \theta_0 \theta_0} & \frac{\partial^2 J}{\partial \theta_0 \theta_1} & \cdots & \frac{\partial^2 J}{\partial \theta_0 \theta_n} \\ \frac{\partial^2 J}{\partial \theta_1 \theta_0} & \frac{\partial^2 J}{\partial \theta_1 \theta_1} & \cdots & \frac{\partial^2 J}{\partial \theta_1 \theta_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 J}{\partial \theta_n \theta_0} & \frac{\partial^2 J}{\partial \theta_n \theta_1} & \cdots & \frac{\partial^2 J}{\partial \theta_n \theta_n} \end{bmatrix}$$

$$\theta^{(i+1)} \leftarrow \theta^{(i)} - (H_J(\theta))^{-1} \nabla J(\theta)$$

what $\frac{\partial^2 J}{\partial \theta_j \theta_k}$; $\frac{\partial J}{\partial \theta_j} = (y - h_\theta(x)) x_j$ (y: $\hat{y}$)

$\frac{\partial}{\partial \theta_k}(y-\hat{y}) x_j = x_j \frac{\partial}{\partial \theta_k}(y-\hat{y})$

$= -x_j \frac{\partial}{\partial \theta_k}(\hat{y})$

$\hat{y} = \frac{1}{1+e^{-\theta^T x}}$

$= -x_j \hat{y}(1-\hat{y}) \frac{\partial}{\partial \theta_k}(\theta^T x)$

$$\begin{bmatrix} 1 & X_1 & \cdots & X_n \\ X_1 & X_1^2 & \cdots & X_1 X_n \\ \vdots & \vdots & \ddots & \vdots \\ X_n & X_n X_1 & \cdots & X_n^2 \end{bmatrix}$$

$= -x_j(\hat{y}-\hat{y}^2)x_k$

$= -(\hat{y}-\hat{y}^2)x_j x_k$

$H_J(\theta) = -(\hat{y}-\hat{y}^2)\, x \cdot x^T$

What can go wrong ??

$\theta^{(i+1)} \leftarrow \theta^{(i)} - H_J^{-1}(\theta^{(i)}) \cdot \nabla_J(\theta^{(i)})$

rank of $H_J(\theta)$
↳ $n+1$: full rank, invertable
↳ $< n+1$: rank deficient, uninvertable

$\text{rank}(XX^T) = 1$

$\text{rank}(X^{(1)}X^{(1)T} + X^{(2)}X^{(2)T}) = 2$

as long as $X^{(1)}$ and $X^{(2)}$ are linearly independent

If $m \leq n$, maximum rank of H is n.

Newton's method requires at least $n+1$

linearly independent training patterns

Generalized Linear Models (GLMs)

up to now :

Linear Regression
$\theta^T x$ is a predictor of the mean of a gaussian distribution
$y \sim N(\theta \cdot x, \sigma^2)$

Logistic Regression
$\frac{1}{1+e^{-\theta^T x}}$ is a predictor of the probability that x comes from class 1, $\left(1 - \frac{1}{1+e^{-\theta^T x}}\right)$ is a predictor of the probability that x comes from class 0
$y \sim \text{Bernoulli}\left(\frac{1}{1+e^{-\theta^T x}}\right)$

Example Bernoulli : fair coin toss chance of the mean is Bernoulli(1/2)

Both gaussian and bernoulli distributions are members of the exponential family of distributions , any members of the family can be written in canonical form

$P(y;\eta) = b(y)e^{(\eta^T T(y) - a(\eta))}$

η is the natural parameter of the distributions

$T(y)$ is the sufficient ($T(y) = y$ for us)

$a(\eta)$ is the log partition function (Normalizer)

$b(y)$ is an arbitrary scalar function

Bernoulli (ϕ) is a member of the exponential family

$P(y=1;\phi) = \phi \quad P(y=0;\phi) = 1-\phi$

$P(y;\phi) = \phi^y(1-\phi)^{1-y}$

substite $z = e^{\log z}$

$= e^{\log(\phi^y(1-\phi)^{1-y})} = e^{\log\phi^y + \log(1-\phi)^{1-y}}$

substite $z = \log e^z$

$= e^{y\log\phi + (1-y)\log(1-\phi)} = e^{y\log\phi + \log(1-\phi) - y\log(1-\phi)}$

$= e^{(\log\phi - \log(1-\phi))y + \log(1-\phi)}$

$b(y)e^{(\eta^T T(y) - a(\eta))} = e^{\left(\log\frac{\phi}{1-\phi}\right)y + \log(1-\phi)}$

$T(y) = y$
$b(y) = 1$
$a(\eta) = ?$

$\eta = \log\frac{\phi}{1-\phi}$ (scalar $\eta^T = \eta$)

$\eta = \log\frac{\phi}{1-\phi}$

$\eta = \log\phi - \log(1-\phi)$

$\log\phi = \eta + \log(1-\phi)$

$\phi = e^{\eta + \log(1-\phi)}$
$= e^{\eta}e^{\log(1-\phi)}$
$= e^{\eta}(1-\phi)$
$\phi = e^{\eta} - e^{\eta}\phi$

$\therefore \phi = e^{\eta}/(1+e^{\eta})$

$a(\eta) = \log(1-\phi)$
$= \log(1 - e^{\eta}/(1+e^{\eta}))$
$= \log(1+e^{\eta} - e^{\eta})/(1+e^{\eta})$
$= \log 1/(1+e^{\eta})$
$= \log 1 - \log(1+e^{\eta}) - \log(1+e^{\eta})$
$a(\eta) = \log(1+e^{\eta})$

this prove that Bernoulli distribution with parameter ϕ is in exponential family

Gaussian distribution with $\sigma^2 = 1$ is a member of the exponential family

$\eta = \mu$, $T(y) = y$, $a(\eta) = \frac{\mu^2}{2} = \frac{\eta^2}{2}$

$b(y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-y^2/2}$

$P(y;\mu) = \frac{1}{\sqrt{2\pi}} e^{\frac{-(y-\mu)^2}{2}}$
$= \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}} e^{(\mu y - \frac{1}{2}\mu^2)}$

What is this GLMs anyway ?
Assumes 3 things

1. $P(y|x;\phi) = \text{exponential family}(\eta)$
2. Given X, we want to predict an expected value of $T(y)$ (or y in most cases) Given X
3. η is linear in x, i.e., $\eta_i = \theta_i^T X$

expected → Logistic Regression

Logistic Regression

$\phi = \frac{e^{\eta}}{1+e^{\eta}} = \frac{1}{1+e^{-\eta}}$
$= \frac{1}{1+e^{-\theta^T x}}$

$E(y|x) = 0 \cdot P(y=0|x)$
$+ 1 \cdot P(y=1|x)$
$= P(y=1|x)$
$[\phi]$

GLMs principle says that if we want to predict ϕ for a Bernoulli distributions as $P(y=1|x)$ then we should use $P(y=1|x)=1/(1+e^{-\theta^T x})$

For Linear Regression, predict
$E(y)=\mu=\theta^T x$
why? Because $\eta=\mu$ give us the exponential family representation of the normal distribution.

For Logistic Regression, predict
$E(y)=P(y=1|x)=1/(1+e^{-\theta^T x})$
why? Because $\eta=\log \phi/(1-\phi)$ give us the exponential family representation of the bernoulli distribution.

GLM procedure:
write target y as result of a probabilistic experiment.
$y \sim P(\phi)$
convert $P(\phi)$ to a member of the exponential family with parameter η.
Plug $\eta_i=\theta_i^T x$ into the expression for ϕ

This is give you a Hypothesis function $h_\theta: X \to Y$

Cost function is always negative log Likelihood

$$J(\theta)=\ell(\theta)=P(x^{(i)}|y^{(i)};\theta)$$

Take $\nabla_J(\theta)=0$, use analytical solution or Gradient Descent or Newton's method or...?

New Distribution: Multinomial
Give x, we would like to predict the class of x, where the class is exactly one of K categories, we assume
$y \sim \text{Multinomial}(\phi_1,\phi_2,\ldots,\phi_K)$
Multinomial Regression or Multinomial Logistic Regression

$$h_\theta(x)=\begin{bmatrix} \frac{e^{\theta_1^T x}}{\sum_{j=1}^{K} e^{\theta_j^T x}} \\ \vdots \\ \frac{e^{\theta_k^T x}}{\sum_{j=1}^{K} e^{\theta_j^T x}} \end{bmatrix}$$
→ Soft max Function
→ Cross Entropy Cost Function

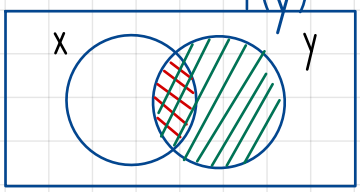
Generative Learning algorithm

So far, Discriminative models that learn $P(y|x)$ directly

Logistic Regression $P(y=1|x;\theta)=1/(1+e^{-\theta^T x})$
never consider $P(x|y)$ or $P(x)$ or $P(y)$

Alternative (Generative) approach
use that fact (Baye's Rule)
$$P(y|x)=\frac{P(x|y)P(y)}{P(x)}$$
(why? Definition of condition)

tell us $P(x|y)=\frac{P(x \wedge y)}{P(y)}=\frac{\text{Area of x}}{\text{Area of u}}$
Venn Diagram



$P(x|y)=$ ▨/▨

$P(x|y)=\frac{P(x \wedge y)}{P(y)} \to P(x \wedge y)=P(x|y)P(y)$
$P(y|x)=\frac{P(y \wedge x)}{P(x)} \to P(y \wedge x)=P(y|x)P(x)$
} equal

$P(x|y)P(y)=P(y|x)P(x)$

$$P(y|x)=\frac{P(x|y)P(y)}{P(x)}$$
Baye's Rule

If we want, given a feature vector x, to estimate a probabilistic distribution over y, we can calculate $P(y|x)$ directly (Logistic Regression) or as $\frac{P(x|y)P(y)}{P(x)}$
Discriminative Generative

Generative approach supposes that we actually model the process by which the dataset was generated

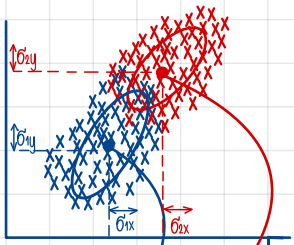
for $i=1$ to n
$y^{(i)} \sim \{0,1\}$ according to prior distribution $P(y)$
$x^{(i)} \sim P(x|y)$
Given x, what is the maximum a posterior estimate of y
$y^*=\arg\max_y P(y|x)$
$=\arg\max_y \frac{P(x|y)P(y)}{P(x)}$ ← does not depend on y
$=\arg\max_y P(x|y)P(y)$

$P(y|x)$ posterior
$P(x|y)$ class-conditional probability
$P(y)$ prior
$P(x)$ data probability (useless constant)

Example $P(x|y)$ is multivariate gaussian

$Y = \{\text{male}, \text{female}\}$

$\mathcal{X} = \mathbb{R}^2$ two features: Person's height and weight



Each class is governed by parameters: mean, covariance

$\mu_1, \Sigma_1, \mu_2, \Sigma_2$

$\mu_1 \quad \mu_2 \quad \begin{bmatrix}\mu_{1x}\\ \mu_{1y}\end{bmatrix} \begin{bmatrix}\sigma_{1x}^2 & \sigma_{1x}\sigma_{1y}\\ \sigma_{1x}\sigma_{1y} & \sigma_{1y}^2\end{bmatrix} \begin{bmatrix}\mu_{2x}\\ \mu_{2y}\end{bmatrix} \begin{bmatrix}\sigma_{2x}^2 & \sigma_{2x}\sigma_{2y}\\ \sigma_{2x}\sigma_{2y} & \sigma_{2y}^2\end{bmatrix}$

5 parameters for each class

$\sigma_{1x}\sigma_{1y}$ how correlated x,y are for class 1

$\sigma_{2x}\sigma_{2y}$ how correlated x,y are for class 2

(PDF)

Covariance matrix specifies elliptical ISO contours of the probabilistic distribution function. Major axis of the elliptical contours. For one class is the eigenvector of the covariance matrix corresponding to its largest eigenvalue.

Given all this,

$$P(x|y=j) = \frac{1}{(2\pi)^{n/2}|\Sigma_j|^{1/2}} e^{-\frac{1}{2}(x-\mu_j)^T\Sigma_j^{-1}(x-\mu_j)} = N(\mu_j, \Sigma_j)$$

Suppose height = 165, weight = 65

What is y^*? $y^* = \arg\max_y P(x|y)P(y)$

Suppose I calculate

$P(x|y=1) = 0.12 \quad P(x|y=2) = 0.10$

Suppose in general population $P(y) = \phi^y(1-\phi)^{1-y}$

priors

$P(y=1) = 0.51 \quad P(y=2) = 0.49$

$P(y=1|x) \sim 0.12 \cdot 0.51$

$P(y=2|x) \sim 0.10 \cdot 0.49$

Suppose the sample is among students DSAI program at AIT

female male

$P(y=1) = 0.4 \quad P(y=2) = 0.6$

$P(y=1|x) \sim 0.12 \cdot 0.4 = 0.048$

$P(y=2|x) \sim 0.10 \cdot 0.6 = 0.06$

$P(x|y=1) = 0.12 \quad P(x|y=2) = 0.10$

$P(y=1) = 0.4 \quad P(y=2) = 0.6$ $P(x)$???

$P(y=1|x) + P(y=2|x) = 1$

$$\frac{P(x|y=1)P(y=1)}{P(x)} + \frac{P(x|y=2)P(y=2)}{P(x)} = 1$$

$$P(x) = P(x|y=1)P(y=1) + P(x|y=2)P(y=2)$$

$$P(x) = \sum_{y_i \in Y} P(x|y=y_i)P(y=y_i)$$

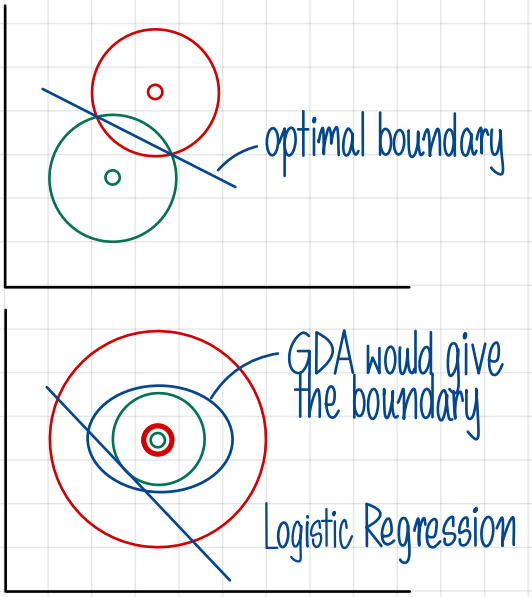
$$P\left(x=\begin{bmatrix}165\\65\end{bmatrix}\right) = 0.12 \cdot 0.4 + 0.10 \cdot 0.6 = 0.108$$

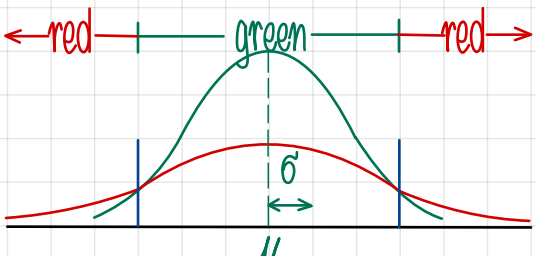
Generative Model with Multivariate Gaussian

GDA: Gaussian Discriminant Analysis

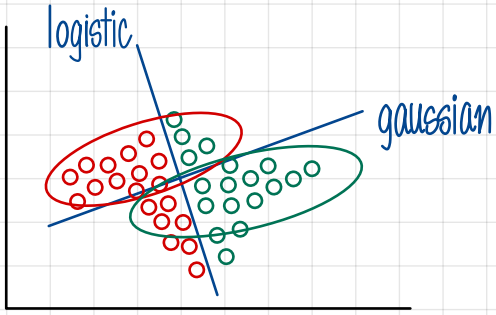
If $P(y=1) = P(y=2)$, $\Sigma_1 = \Sigma_2$

GDA will give the same boundary as Logistic Regression





If you know that the data for each class are approximately gaussian and covariance are not equal, gaussian generative model will outperform the logistic Regression model.



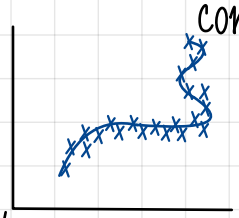
Generative vs Discriminative

$P(x|y)$, $P(y)$ $\quad$ $P(y|x)$

Gaussian Discriminant Analysis using Gaussian to discriminant between classes, Generative model

Generative Models

Gaussian: means, covariance: continuous variables



Example
Non-Gaussian but continuous

What about discrete data?

Example: Car Buying

$Y = \{\text{Buy}, \neg\text{Buy}\}$

$X = [\text{Size}, \text{Colors}, \text{Gender}, \text{Age}]$

{small, medium, large} {red, blue, black, white} {male, female, prefer not to say} {0-18, 19-39, 40-59, 60+}

Generative Approach:

$P(y|x) \propto P(x|y)P(y)$ $\quad 4\times3\times5\times3=180$

$[X_1, X_2, X_3, X_4]$

Size, Age

$P(a,b) = P(a)P(b|a)$ conditional probability $\quad P(b|a) = \frac{P(a,b)}{P(b)}$

$P(b|a)P(b) = P(a,b)$

exactly

$P([X_1, X_2, X_3, X_4]) = P(X_1)P(X_2|X_1)P(X_3|X_1, X_2)P(X_4|X_1, X_2, X_3)$

Factorizing the Joint distribution

$P(\mathbf{x}) = \prod_{i=1}^{m} P(x_i|x_1, \ldots, x_{i-1})$

$P(\mathbf{x}|y) = \prod_{i=1}^{m} P(x_i|x_1, \ldots, x_{i-1}, y)$

$P(y|\mathbf{x}) \propto P(x_1|y)P(x_2|x_1, y)P(x_3|x_1, x_2, y)P(x_4|x_1, x_2, x_3, y)P(y)$

$4\times3\times5\times2$
360 entries in the table!

exactly

Naive Bayes Assumption:

$P(x_j|x_1, \ldots, x_{i-1}, y) \approx P(x_j|y)$

$P(\text{Colors} = \text{red}|\text{Size} = \text{medium}, \text{Outcome} = \text{Buy}) \approx$

$P(\text{Colors} = \text{red}|\text{Outcome} = \text{Buy})$

$P(y|\mathbf{x}) \propto P(x_1|y)P(x_2|y)P(x_3|y)P(x_4|y)P(y)$

approx

Complicated Joint Distribution is approximated by a product of simple conditional probabilities

Advantage: fewer parameters

Parameters:

$x_1 = \{S, M, L\}$ $\quad y = \{B, \neg B\}$

$P(X_1|y)$

X_1	B	$\neg$B
S	0.3	0.3
M	0.3	0.5
L	0.4	0.2
total	1.0	1.0

Size		6 parameters	
Colors	4×2	8 parameters	$P(X_2|y)$
Gender	3×2	6 parameters	$P(X_3|y)$
Age	5×2	10 parameters	$P(X_4|y)$
Outcome		2 parameters	$P(y)$

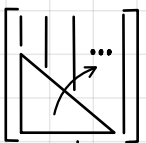
32 parameters total!
vs 362 for exact calculation!
(180×2+2 $P(x|y)P(y)$)

Generally speaking, A simple model will generalize better (less chance to overfit). Naive Bayes approximation is a Regularizer. Naive Bayes usually much better than full bayesian Classification.

How many parameters in Gaussian model?

mean, covariance

$\mathcal{X} = \mathbb{R}^2$ $\quad$ mean k vectors of length N, $k \cdot N$ parameters

$Y = \{1, \ldots, k\}$ $\quad$ covariance  $\quad \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

n, k $\quad k\frac{n(n+1)}{2}$ (dominated by kN^2)

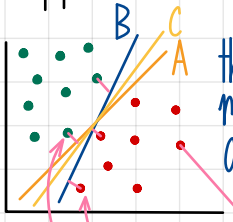
2 classes, 10 parameters $\quad 2\cdot10 \quad \frac{2\cdot10\cdot11}{2}$

$20+110 = 130$ (40 for Diagonal covariance matrices)

Simplification: Diagonal covariance matrices (off diagonal terms = 0): $2 \cdot k \cdot N$ $\quad k \cdot N$ $\quad k \cdot N$

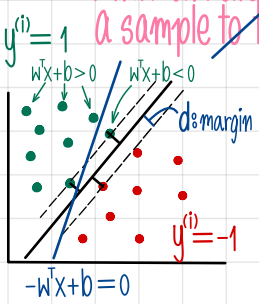
mean, covariance

Support Vector Machines (SVM)



the best classifier is the boundary that maximizes the minimum distance of any sample to the boundary

minimum distance of a sample to the boundary



maximizes the minimum distance to the hyperplane

maximizes the margin

Line in 2D: $ax_1+bx_2+c=0$

Line in 3D: $ax_1+bx_2+cx_3+d=0$

weights $w_1\ w_2\ w_3$ bias b

$$w=\begin{bmatrix}w_1\\w_2\\w_3\end{bmatrix}\ b$$

parameters to estimate: w, b

$$w^*, b^* = \underset{w,b}{\text{argmax}}\ \text{"margin"}$$

parameters w,b define the separating hyperplane

Geometric margin γ

Function margin $\hat{\gamma}$

Assume $Y=\{-1, 1\}$

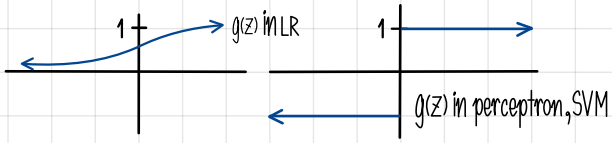
$X=\mathbb{R}^2$

Logistic Regression

$h_{w,b}(x)=g(w^Tx+b)$ $\quad h_\theta(x)=g(\theta^Tx)$

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

augmented with 1



Margin

Functional margin: $y^{(i)}=1\ w^Tx+b>0$ $\quad y^{(i)}=-1\ w^Tx+b<0$

$$\hat{\gamma}^{(i)}=y^{(i)}(w^Tx^{(i)}+b)$$

$+1$ ≥ 0 for A+1 prediction

-1 < 0 for A−1 prediction

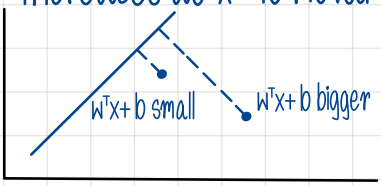
If $\hat{\gamma}^{(i)}>0$ for $i\in 1\dots m$ $\quad h(x^{(i)})=y^{(i)}$

All training examples are correctly classified.

If $\hat{\gamma}^{(i)}<0$ for any i,

then samples i is incorrectly classified

$\hat{\gamma}^{(i)}$ increases as $x^{(i)}$ is moved further from the boundary



We said our goal was to maximizes the margin

First attempt (incorrect!!)

$$w^*, b^* = \underset{w,b}{\text{argmax}}\ \min_i \hat{\gamma}^{(i)}$$

Problem: $\|w\|$ is not bounded range of

If w,b give $\min_i \hat{\gamma}^{(i)}=\varepsilon>0$,

then sw, sb give $\min_i \hat{\gamma}^{(i)}=s\varepsilon>0$ where $s>1$

Instead of w,b we would like to normalize:

$$w^*, b^* = \underset{w,b}{\text{argmax}}\ \min_i y^{(i)}\left[\frac{w^Tx}{\|w\|}+\frac{b}{\|w\|}\right]$$

Geometric margin: of hyperplane (w,b)

$$\gamma^{(i)}=y^{(i)}\left[\left(\frac{w}{\|w\|}\right)^Tx^{(i)}+\frac{b}{\|w\|}\right]$$

Now we can revise the objection to

$$w^*, b^* = \underset{w,b}{\text{argmax}}\ \min_i \gamma^{(i)}$$

Let $\gamma=\min_i \gamma^{(i)} \rightarrow w^*, b^*=\underset{w,b}{\text{argmax}}\ \gamma$

Goal (Revised)

$\max_{\gamma,w,b}\gamma$ nonlinear

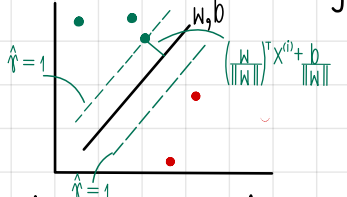
subjet to $y^{(i)}(w^Tx^{(i)}+b)\geq\gamma,\ i=1\dots m$ $\quad \|w\|=1$

$\max_{\gamma,w,b}\frac{\hat{\gamma}}{\|w\|}$ } non-convex $\quad \gamma=\frac{\hat{\gamma}}{\|w\|}$

subjet to $y^{(i)}(w^Tx^{(i)}+b)\geq\hat{\gamma},\ i=1\dots m$

→ instead of requiring $\|w\|=1$

force $\|w\|=$ whatever give us $\hat{\gamma}=1$



$\hat{\gamma}=\min_i \hat{\gamma}^{(i)}$ $\quad \hat{\gamma}=y^{(i)}(w^Tx^{(i)}+b)$

If $\gamma = 1.8$, for example

If we let $w \leftarrow \frac{w}{\|w\| \cdot 1.8}$ $b \leftarrow \frac{b}{\|w\| \cdot 1.8}$

$\hat{\gamma} = 1$

$\hat{\gamma} = 1$

$\Rightarrow \min_{\gamma, w, b} \frac{1}{2}\|w\|^2$

subject to $y^{(i)}(w^T x^{(i)} + b) \geq 1$, $i = 1 \dots m$

$\frac{1}{2}\|w\|^2 = \frac{1}{2}(w_1^2 + w_2^2 + w_3^2 + \dots + w_n^2)$

Objection Function Quadratic in parameters
constraints that are linear, QP
Quadratic Programming

QP given

$c \in \mathbb{R}^n$ $Q \in \mathbb{R}^{n \times n}$

$A \in \mathbb{R}^{m \times n}$ $b \in \mathbb{R}^m$

Find $x^* \in \mathbb{R}^n = \underset{x}{\operatorname{argmin}} \frac{1}{2} x^T Q x + c^T x$ (# of parameters)

subject to $Ax \leq b$

QP ⟷ SVM

$x \longleftrightarrow \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ b \end{bmatrix}$

$\min \frac{1}{2} x^T Q x + c^T x \longleftrightarrow \min \frac{1}{2}(w_1^2 + \dots + w_n^2)$

$Q = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1\ 0 \\ 0 & 0 & \cdots & 0\ 0 \end{bmatrix}$ $c = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$

$Ax \leq b$

$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} \leq \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{bmatrix}$

$y^{(i)}(w^T x^{(i)} + b) \geq 1$

$\begin{bmatrix} y^{(i)}x_1^{(i)} & y^{(i)}x_2^{(i)} & \dots & y^{(i)}x_n^{(i)} & y^{(i)} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ b \end{bmatrix} \geq [1]$

$\begin{bmatrix} -y^{(i)}x_1^{(i)} & -y^{(i)}x_2^{(i)} & \dots & -y^{(i)}x_n^{(i)} & -y^{(i)} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ b \end{bmatrix} \leq [-1]$

$A = \begin{bmatrix} -y^{(1)}x^{(1)T} & -y^{(1)} \\ \vdots & \\ -y^{(m)}x^{(m)T} & -y^{(m)} \end{bmatrix}$ $b = \begin{bmatrix} -1 \\ \vdots \\ -1 \end{bmatrix}$

To find w, b maximizes the margin, let

$Q = \begin{bmatrix} I_{n \times n} & 0 \\ 0 & 0 \end{bmatrix}$ $c = 0_{n \times 1}$ $\begin{bmatrix} x_1^{(1)} & \cdots & x_n^{(1)} & 1 \\ & \vdots & & \\ x_1^{(m)} & \cdots & x_n^{(m)} & 1 \end{bmatrix}$

$b = -1_{m \times 1}$ Python Numpy Broadcast ↘ $A = -y * X$ ←

pip install cvxopt ← "convex optimization"

☺ we can maximizes the margin! ☹ Not for big m or big n! 1920×1080?

☺ There must be a linear hyperplane "shattering" the training set

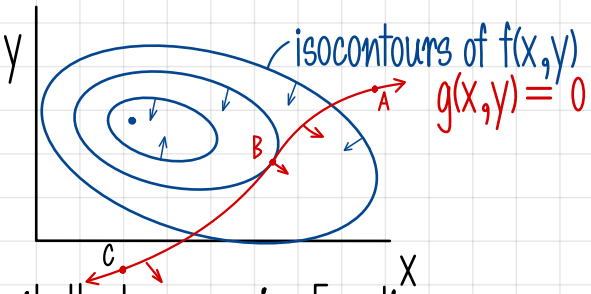
$X^T Q$

$\begin{bmatrix} w_1 & w_2 & \cdots & w_n & 0 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ 0 \end{bmatrix}$ $w_1^2 + \dots + w_n^2 + 0$

Lagranagian Optimization

Example : maximize $f(x,y)$ subject to $g(x,y) = 0$

constrained maximization



We write the Lagranagian Function containing a Lagrange multiplior λ

$$\mathcal{L}(x,y,\lambda) = f(x,y) - \lambda g(x,y)$$

Theorem : If (x^*,y^*) is a maximum of $f(x,y)$ satisfying $g(x^*,y^*) = 0$, then there exist A λ^* such that (x^*,y^*,λ^*) is a stationary point of $\mathcal{L}(x,y,\lambda)$

That means that (x^*,y^*,λ^*) satisfies

$$\frac{\partial \mathcal{L}}{\partial x} = 0, \quad \frac{\partial \mathcal{L}}{\partial y} = 0, \quad \frac{\partial \mathcal{L}}{\partial \lambda} = 0$$

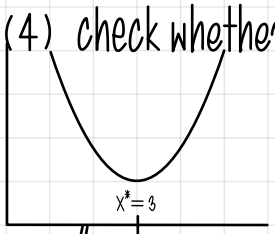
To find max of $f(x,y)$ such that $g(x,y) = 0$

(1) write the Lagranagian

(2) set derivative w.r.t x, y, λ to 0

(3) solve !!

(4) check whether the solution is a max or not



$$y = (x-3)^2+2$$
$$y' = 2(x-3)$$
$$0 = 2x-6$$
$$x^* = 3$$
$$y'' = 2 \leftarrow \text{positive } 2^{nd} \text{ derivative indicates } y^* \text{ is a minimum}$$

For a multivariate Function,

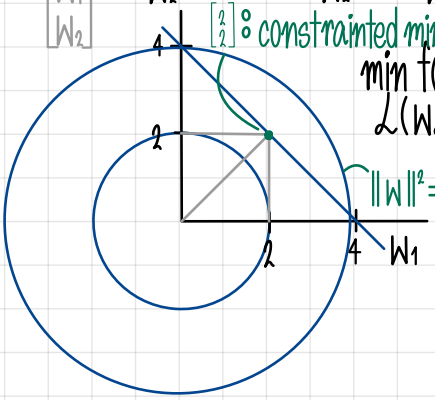
H_f : Hessian of f. Unconstrianted optimization
positive eigenvalues for minima,
negative eigenvalues for maxima

H_L : Constrianted optimization
all stationary point are "saddle" points

Specific example :

$$f(w) = \|w\|^2, g(w) = w_1 + w_2 - 4$$
$$0 = w_1 + w_2 - 4$$
$$w_2 = -w_1 + 4$$

$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$: constrainted min.



$$\min f(w) \text{ s.t. } g(w) = 0$$
$$\mathcal{L}(w,\lambda) = f(w) - \lambda g(w)$$
$$= w_1^2 + w_2^2 - \lambda(w_1 + w_2 - 4)$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = 2w_1 - \lambda \qquad \frac{\partial \mathcal{L}}{\partial w_2} = 2w_2 - \lambda$$
$$\frac{\partial \mathcal{L}}{\partial \lambda} = -(w_1 + w_2 - 4)$$

$2w_1 - \lambda = 0 \rightarrow \lambda = 2w_1 \rightarrow \lambda = 4$

$2w_2 - \lambda = 0 \rightarrow 2w_2 - 2w_1 = 0 \rightarrow w_1 = w_2 \rightarrow w_2 = 2$

$-(w_1 + w_2 - 4) = 0 \rightarrow -(w_1 + w_1 - 4) = 0 \rightarrow 2w_1 = 4 \rightarrow w_1 = 2$

$$w_1 = 2 \quad w_2 = 2 \quad \lambda = 4$$

$\min_w f(w)$
s.t. $g_i(w) \le 0, i \in 1 \dots k$
$h_i(w) = 0, i \in 1 \dots l$

If w satisfies constriant
$h_i(w) = 0$, $g_i(w) \le 0$

The Generalized Lagranagian

$$\mathcal{L}(w,\alpha,\beta) = f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{i=1}^{l} \beta_i h_i(w)$$

Primal Optimization

≤ 0 If w satisfies

$$\theta_p(w) = \max_{\alpha,\beta,\alpha_i \ge 0} \mathcal{L}(w,\alpha,\beta)$$
$$= \max_{\alpha,\beta,\alpha_i \ge 0} f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{i=1}^{l} \beta_i h_i(w)$$

Claim : If w violates any constriant, then $\theta_p(w) = \infty$

If w satisfies constriant, then $\theta_p(w) = f(w)$

So far, we only max w.r.t α, β, w is fixed ($\theta_p(w) = f(w)$ or ∞)

Let w^* be the w satisfying

$$\min_w \theta_p(w) = \min_w \max_{\alpha,\beta,\alpha_i \ge 0} \mathcal{L}(w,\alpha,\beta)$$

$p^* = \min_w \theta_p(w)$ is called the value of the primal problem

Primal is what we want to solve, but it's not easy always.
Instead, sometimes, the dual problem is easier.

$$\theta_D(\alpha,\beta) = \min_w \mathcal{L}(w,\alpha,\beta)$$

Dual problem:

$$\max_{\alpha,\beta,w\geq 0} \theta_D(\alpha,\beta) = \max_{\alpha,\beta,\alpha_i\geq 0} \min_w \mathcal{L}(w,\alpha,\beta)$$

$$d^* = \max_{\alpha,\beta,\alpha_i\geq 0} \theta_D(\alpha,\beta)$$

is called the value of the dual problem

Under Specific Circumstances

$$p^* = d^*$$

$p^* = d^*$ Under these conditions

f is convex

The g_i's are convex

The h_i's are affine

The g_i's are strictly feasible

For SVM linearly seperable data

($\exists w$ s.t. $g_i(w) \leq 0$ for all)

When these conditions hold, $\exists w^*, \alpha^*, \beta^*$

(1) w^* is the solution of the primal problem

(2) α^*, β^* are the solution of the dual problem

(3) $p^* = d^* = \mathcal{L}(w^*,\alpha^*,\beta^*)$

(4) w^*, α^*, β^* satisfy the KKT conditions

Lagranagian Duality

$$\frac{\partial}{\partial w_i}\mathcal{L}(w^*,\alpha^*,\beta^*) = 0, \quad i\in 1\dots n$$

$$\frac{\partial}{\partial \beta_i}\mathcal{L}(w^*,\alpha^*,\beta^*) = 0, \quad i\in 1\dots l$$

$$\alpha_i^* g_i(w^*) = 0, \quad i\in 1\dots k$$

for every i, $\alpha_i = 0$ or $g_i(w^*) = 0$

$$g_i(w^*) \leq 0, \quad i\in 1\dots k$$

$$\alpha_i^* \geq 0, \quad i\in 1\dots k$$

Back to SVMs!

$$\min_{w,b} \frac{1}{2}\|w\|^2$$

$$\text{s.t} \quad y^{(i)}(w^T x^{(i)} + b) \geq 1, \quad i = 1\dots m$$

inequalities, affine g_i's

$$g_i(w) = -y^{(i)}(w^T x^{(i)} + b) + 1 \leq 0$$

We will have one α_i for each training sample

KKT tell us $\alpha_i = 0$ or $g_i(w) = 0$

$$\mathcal{L}(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^m \alpha_i [y^{(i)}(w^T x^{(i)} + b) - 1]$$

Generalized Lagranagian for the SVM maximum margin problem

Find $\frac{\partial \mathcal{L}}{\partial w_j}$

$$w_1^2 + \dots + w_j^2 + \dots + w_n \qquad w_1^2 x_1^{(i)} + \dots + w_j^2 x_j^{(i)} + \dots + w_n^2 x_n^{(i)}$$

$$\frac{\partial \mathcal{L}(w,b,\alpha)}{\partial w_i} = \frac{1}{2}\|w\|^2 - \sum_{i=1}^m \alpha_i [y^{(i)}(w^T x^{(i)} + b) - 1]$$

$$= w_j - \sum_{i=1}^m \alpha_i y^{(i)} x_j^{(i)}$$

$$\nabla_w \mathcal{L}(w,b,\alpha) = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial w_1} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial w_n} \end{bmatrix} = \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^m \alpha_i y^{(i)} x_1^{(i)} \\ \vdots \\ \sum_{i=1}^m \alpha_i y^{(i)} x_n^{(i)} \end{bmatrix}$$

$$= w - \sum_{i=1}^m \alpha_i y^{(i)} x^{(i)}$$

0 For $x^{(i)}$ far from boundary, $\alpha_i \geq 0$ For points defining boundary

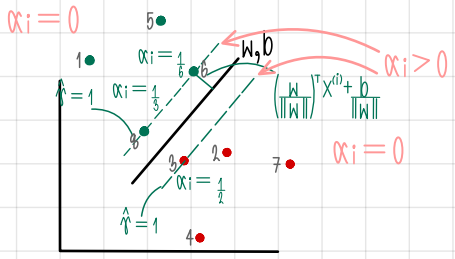
$$\nabla_w \mathcal{L}(w,b,\alpha) = 0$$

$$w = \sum_{i=1}^m \alpha_i y^{(i)} x^{(i)}$$

$-1, +1$

Find $\frac{\partial \mathcal{L}}{\partial \beta} = -\sum_{i=1}^m \alpha_i y^{(i)} = 0$

the sum of α_i's for positives must be equal to the sum of α_i's for negatives



S = "support vectors" = {3, 6, 8}

Plotting w^* into $\mathcal{L}(w,b,\alpha)$, we obtain

$$\mathcal{L}(w,b,\alpha)=\frac{1}{2}\|w\|^2-\sum_{i=1}^{m}\alpha_i[y^{(i)}(w^Tx^{(i)}+b)-1]$$

$$=\frac{1}{2}\|\sum_{i=1}^{m}\alpha_i y^{(i)}x_j^{(i)}\|^2-\sum_{i=1}^{m}\alpha_i[y^{(i)}((\sum_{j=1}^{m}\alpha_i y^{(j)}x^{(j)})^Tx^{(i)}+b)-1]$$

$$=\frac{1}{2}\left[\sum_{i=1}^{m}\alpha_i y^{(i)}x^{(i)}\right]^T\left[\sum_{j=1}^{m}\alpha_j y^{(j)}x^{(j)}\right]-\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i y^{(i)}\alpha_j y^{(j)}x^{(j)T}x^{(i)}-\sum_{i=1}^{m}\alpha_i y^{(i)}b+\sum_{i=1}^{m}\alpha_i$$

$$=\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i y^{(i)}\alpha_j y^{(j)}x^{(i)T}x^{(j)}-\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i y^{(i)}\alpha_j y^{(j)}x^{(j)T}x^{(i)}-\sum_{i=1}^{m}\alpha_i y^{(i)}b+\sum_{i=1}^{m}\alpha_i$$

$b\sum_{i=1}^{m}\alpha_i y^{(i)}\leftarrow 0$

$$=\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y^{(i)}y^{(j)}x^{(i)T}x^{(j)}-b\sum_{i=1}^{m}\alpha_i y^{(i)}$$

$$\mathcal{L}(w,b,\alpha)=\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}y^{(i)}y^{(j)}\alpha_i\alpha_j x^{(i)T}x^{(j)}$$

Generalized Lagranagian Dual problem :

$$\max_{\alpha} W(\alpha)=\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{m}y^{(i)}y^{(j)}\alpha_i\alpha_j\langle x^{(i)},x^{(j)}\rangle$$ inner (dot) product

such that $\alpha_i\geq 0$, $i\in 1\dots m$ $\quad p^*=d^*$, α's will solve the primal problem

$\sum_{i=1}^{m}\alpha_i y^{(i)}=0$ ← equality constriant

$$b^*=\frac{\max_{i:y^{(i)}=-1}w^{*T}x^{(i)}+\min_{i:y^{(i)}=1}w^{*T}x^{(i)}}{2}$$

$$b^*=\frac{1}{N_s}\sum_{i\in S}(y^{(i)}-\sum_{j\in S}\alpha_j y^{(j)}\langle x^{(i)},x^{(j)}\rangle)$$

S: set of indices of "support vectors" (i s.t $\alpha_i\geq 0$)

Inference problem
given new point x what is $\hat{y}$?

$$h_{w,b}(x)=\begin{cases}+1 \text{ if } w^Tx^{(i)}+b\geq 0\\ -1 \text{ otherwise}\end{cases}$$

$$w^Tx^{(i)}+b=\sum_{i=1}^{m}\alpha_i y^{(i)}\langle x^{(i)},x\rangle+b$$

S.V label ↑ S.V ↑ input examples

We can answer the question of which side of the boundary w, b is a point x is on without ever calculated w only need $\alpha^{(i)}$'s, $y^{(i)}$'s, $x^{(i)}$'s (only for the s.v's) and b

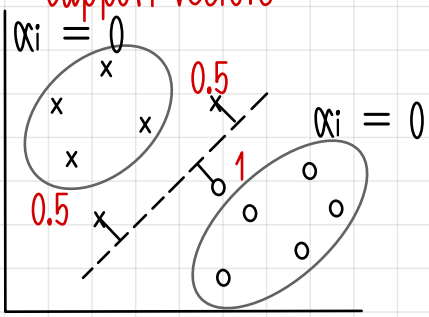
Formulated the SVM problem
maximize the margin

Functional $\hat{\gamma}$ Geometric γ

Primal, Dual Lagranagian Optimization

Solution in terms of α_i's

>0 support vectors $\quad 0$ Insignificant examples



$$\max_{\alpha} W(\alpha)=\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{m}y^{(i)}y^{(j)}\alpha_i\alpha_j\langle x^{(i)},x^{(j)}\rangle$$

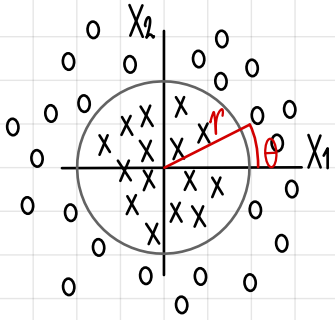
such that $\alpha_i\geq 0$, $i\in 1\dots m$ $\quad \sum_{i=1}^{m}\alpha_i y^{(i)}=0$

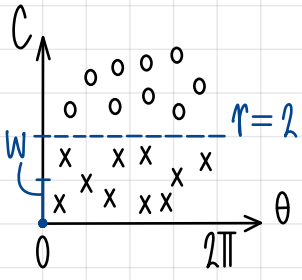
Problem
1 large scale
2 linearly seperable training set
3 not much different from Logistic Regression (nonlinear boundary)

Solution to Problem #3 nonlinear transformation into "Feature Space"

$\phi: X\rightarrow \mathbb{R}^2$ $\quad \phi=\begin{bmatrix}\text{atan2}(x_2,x_1)\\ \sqrt{x_1^2+x_2^2}\end{bmatrix}$

"annilus"





Cartesian → Polar Idea: transform x_1, x_2 into θ, r new space where the boundary is linear

$$\max_{\alpha} W(\alpha)=\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{m}y^{(i)}y^{(j)}\alpha_i\alpha_j\langle\phi(x^{(i)}),\phi(x^{(j)})\rangle$$

what are w, b?

$\begin{bmatrix}0\\1\end{bmatrix}$ $\quad -2$

Possible transforms:

Polynomial

$$\phi(x) = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \\ x_1^2 \\ \vdots \end{bmatrix}$$

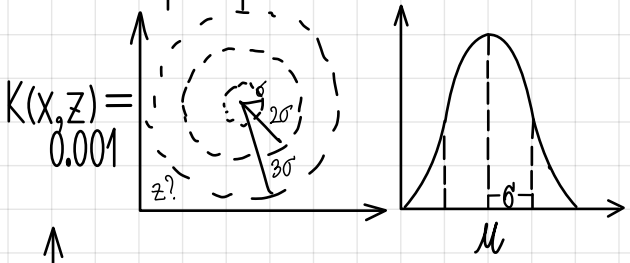
Others:
Kernel function
$K: \mathbb{R}^n \cdot \mathbb{R}^n \to \mathbb{R}^2$
Linear Kernel
$K(x,x') = x_1x'_1 + \dots + x_n x'_n = \langle x, x' \rangle$

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j K(x^{(i)}, x^{(j)})$$

RBF (Radial Basis Function) Kernel:

$K(x,x') = e^{-\|x-x'\|^2/2\sigma^2}$ or $e^{-\gamma\|x-x'\|^2}$ gamma

Intution
A Kernel says how "similar" two vectors in the input space are



$$\cos\theta = \frac{x \cdot z}{\|x\|\|z\|}$$

$$x \cdot z = \|x\|\|z\|\cos\theta$$

Linear Kernel does measure similarity in direction

Once we have our α_i's then interference with the Kernel formulation:

$$\text{Class}(x) = \text{sgn}\left[\sum_{i \in S} \alpha_i y^{(i)} K(x^{(i)}, x^{(j)}) + b\right]$$ sum over entire training set!

Logistic Regression: $[\theta_1 \cdots \theta_n]^T$, θ_0

$$\text{Class}(x) = \text{sgn}(w^T x^{(i)} + b)$$

$K(x,z) = x_1z_1 + \dots + x_nz_n$

Linear: corresponds to $\phi(x) = x$

transform presentation $\sum\sum y^{(i)} y^{(j)} \alpha_i \alpha_j \langle \phi(x^{(i)}), \phi(x^{(j)}) \rangle$

$K(x,z) = e^{-\|x-z\|^2/(2\sigma^2)}$

ϕ = transforms into an infinite dimensional space!

Are there Kernel function that don't correspond to any ϕ at all? Yes

THM: Kernel function $K(x\ z)$, corresponds to features space transformation ϕ IFF

The gram matrix

is always positive semidefinite $K = \begin{bmatrix} K(x^{(1)}x^{(1)}) & K(x^{(1)}x^{(2)}) & \cdots \\ K(x^{(2)}x^{(1)}) & K(x^{(2)}x^{(2)}) & \cdots \\ \vdots & \vdots & \\ K(x^{(m)}x^{(1)}) & K(x^{(m)}x^{(2)}) & \cdots \end{bmatrix}$

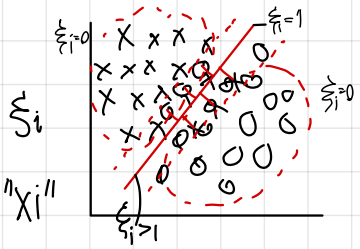
In practice

Linear → Fast (minimal Support Vector n+1) but strictly linear, worst accuracy least overfitting

Polynomial → Medium speed $\approx n^2$ Support Vectors for quadatic, etc. Medium accuracy Medium overfitting depend in degree

RBF → Slow $\approx n$ Support Vectors Best accuracy (training set) Most overfitting

Solution to Problem # 2 Overlap between classes



For RBF
Seperating slack cars the sum has two Hyperparameter r, C

"SLACK" variables expressing How far across the boundary a point lies

Add ξ_i's to optimization for $\epsilon > 0$

Introdule Hyperparameter

$$\frac{C}{C\sum_{i=1}^{m} \xi_i}$$ strength of the "Don't overlap hestric compared to maximize margin OBJECTION

0 linearly seperable

positive value for non linearly seperable data

To be minimized

New Objective:

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j K(x^{(i)}, x^{(j)})$$

such that $0 \leqslant \alpha_i \leqslant C$ $\quad \sum_{i=1}^{m} \alpha_i y^{(i)} = 0$

Unsupervised Learning

Supervised : given $(\mathbf{x}, y)$ pairs

, find best $h_\theta : \mathcal{X} \to y$

predicting target y given input $\mathbf{x}$

Unsupervised : given $\mathbf{x}$, ... ? Unbalanced data

1. Find cluster $\mathbf{x}$ belongs to
2. Masked Language Modeling (MLM)

$\mathbf{x}$: "the quick fix jumped over the lazy brown dog"

$\mathbf{x}$: "the ____ fix jumped over the lazy brown dog"
(quick)

3. Dimension Reduction (Latent embedding)

$f : \mathbb{R}^n \to \mathbb{R}^m$, $m \ll n$

such that

$f' : \mathbb{R}^m \to \mathbb{R}^n$ is as accurate as possible

$$f, f' \leftarrow \underset{f, f'}{\operatorname{argmin}} \| \mathbf{x} - f'(f(\mathbf{x})) \|$$

4. Density estimation

$f : \mathbb{R}^n \to \mathbb{R}$

$f(\mathbf{x})$ predicts $p(\mathbf{x} = x)$

K-mean clustering

input : training set $S = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$

$\mathbf{x}^{(i)} \in \mathcal{X} = \mathbb{R}^n$

Goal : Group the $\mathbf{x}^{(i)}$ into clusters

Algorithm :

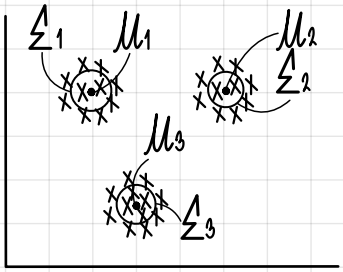
1. Randomly initialize K cluster centroids $\mu_1, \mu_2, \dots, \mu_K \in \mathbb{R}^n$
2. Repeat until convence

colour the points

1. for $i = 1, \dots, m$, $c^{(i)} \leftarrow \underset{j}{\operatorname{argmin}} \| \mathbf{x}^{(i)} - \mu_j \|^2$
2. for $j = 1, \dots, K$ $\mu_j \leftarrow \dfrac{\sum_{i=1}^{m} \delta(c^{(i)} = j) \mathbf{x}^{(i)}}{\sum_{i=1}^{m} \delta(c^{(i)} = j)}$

move the cluster centroids

Gaussian mixture model



$\phi_1 = \frac{10}{32}$

$\phi_2 = \frac{9}{32}$

$\phi_3 = \frac{13}{32}$

Generative model for each sample :

1. Select a component 1-K according to a multinomial distribution with parameters $\phi_1, \dots, \phi_K$ (priors), ($\sum_{i=1}^{m} \phi_i = 1$)

Let i be the selected component

2. sample the datum $\mathbf{x}_j \in \mathbb{R}^n$ according to

$\mathbf{x}_j \sim N(\mu_i, \Sigma_i)$

Learning Objective :

estimate priors $\phi_1, \dots, \phi_K$
means $\mu_1, \dots, \mu_K$
covariances $\Sigma_1, \dots, \Sigma_K$

Cost function : negative log likelihood

Steps : 1. write Cost function

2. take derivatives

3. try to solve the resulting system of equations

Log likelihood function

$$\ell(\phi, \mu_1, \dots, \mu_K, \Sigma_1, \dots, \Sigma_K) = \sum_{i=1}^{m} \log P(\mathbf{x}^{(i)}, z^{(i)}; \phi, \mu_1, \dots, \Sigma_1, \dots)$$

could have been generated by any of the components in the mixture.

Let $z^{(i)}$ be the (hidden) component that generated example i. $z^{(i)}$ could have any value 1 to K

Aside : it is always valid to write probability as a sum

of conditional probabilities

$$P(A) = \sum_{b \in B} P(A|B=b)P(B=b)$$

$$\sum_{i=1}^{m} \log P(x^{(i)}, z^{(i)}; \phi, \mu_1, \dots, \Sigma_1, \dots)$$

$$= \sum_{i=1}^{m} \log \sum_{j=1}^{k} P(x^{(i)}|z^{(i)}=j; \mu_j, \Sigma_j) P(z^{(i)}=j; \phi_j)$$

$$= \sum_{i=1}^{m} \log \sum_{j=1}^{m} \phi_j N(x^{(i)}; \mu_j, \Sigma_j) \quad \text{(given } z^{(i)}\text{'s)}$$

$$= \sum_{i=1}^{m} [\log P(x^{(i)}|z^{(i)}; \mu_{z^{(i)}}, \Sigma_{z^{(i)}}) + \log P(z^{(i)}; \phi_j)]$$

$$= \sum_{i=1}^{m} [\log \phi_{z^{(i)}} + \log N(x^{(i)}; \mu_{z^{(i)}}, \Sigma_{z^{(i)}})]$$

$$\frac{\partial \ell}{\partial \phi_j} = 0 \qquad \sum_{i=1}^{m} \delta(z^{(i)}=j) \frac{1}{\phi_j} = 0$$

$\phi_j = \infty$? or very large? not satisfactory, $\sum_{j=1}^{m} \phi_i = 1 \rightarrow \sum_{j=1}^{m} \phi_i - 1 = 0$

To solve the constrained Optimization, use Lagranagian

$$\ell(\phi, \mu_1, \dots, \Sigma_1, \dots) + \lambda\left(\sum_{j=1}^{m} \phi_i - 1\right)$$

$$\sum_{i=1}^{m} \frac{N(x^{(i)}; \mu_{z^{(1)}}, \Sigma_{z^{(1)}})}{\sum_{i=1}^{m} \phi_i N(x^{(i)}; \mu_{z^{(1)}}, \Sigma_{z^{(1)}})} + \lambda = 0$$

Skips sumsteps $\lambda = -m$

Also task derivatives $\frac{\partial \ell}{\partial \mu_j} = 0$

with some works, you get

$$\phi_j = \frac{1}{m} \sum_{i=1}^{m} \delta(z^{(i)}=j)$$

$$\mu_j = \frac{\sum_{i=1}^{m} \delta(z^{(i)}=j) x^{(i)}}{\sum_{i=1}^{m} \delta(z^{(i)}=j)}$$

$$\Sigma_j = \frac{\sum_{i=1}^{m} \delta(z^{(i)}=j)(x^{(i)}-\mu_j)(x^{(i)}-\mu_j)^T}{\sum_{i=1}^{m} \delta(z^{(i)}=j)}$$

Just like the GDA Model

Expectation maximization (EM)

Algorithm: parameters estimation under uncertain hidden variable

Two Steps

- E-Steps (Expectation): guess values of hidden variables ($z^{(i)}$)
- M-Steps (Maximization): maximum likelihood estimation (MLE) based on the guess

EM Algorithm:

Repeat untill convence:

E-Steps:

for each $i \in 1, \dots, m$, $j \in 1, \dots, k$,

$$w_j^{(i)} \leftarrow P(z^{(i)}=j | x^{(i)}; \phi, \mu_1, \dots, \Sigma_1, \dots)$$

↑ the responsibility of component j for example i

M-Steps:

$$\phi_j = \frac{1}{m} \sum_{i=1}^{m} w_j^{(i)}$$

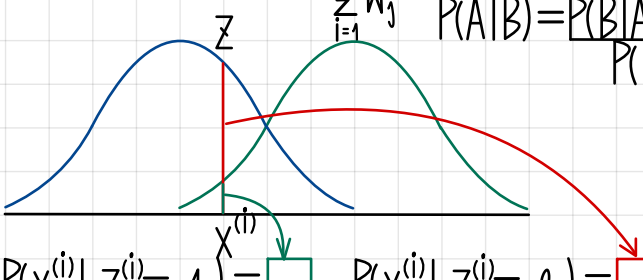
optimal values of parameters under "soft" assignments $w_j^{(i)}$

$$\mu_j = \frac{\sum_{i=1}^{m} w_j^{(i)} x^{(i)}}{\sum_{i=1}^{m} w_j^{(i)}}$$

$w_j^{(i)}$ ← soft; $\delta(z^{(i)}=j)$ ← hard

$$\Sigma_j = \frac{\sum_{i=1}^{m} w_j^{(i)}(x^{(i)}-\mu_j)(x^{(i)}-\mu_j)^T}{\sum_{i=1}^{m} w_j^{(i)}}$$

$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$



$P(x^{(i)}|z^{(i)}=1) = \square$ $P(x^{(i)}|z^{(i)}=2) = \square$

$\frac{\square}{\square+\square}$ $P(z^{(i)}=1|x^{(i)}) = \frac{P(x^{(i)}|z^{(i)}=1)P(z^{(i)}=1)}{P(x)}$

$\frac{\square}{\square+\square}$ $P(z^{(i)}=2|x^{(i)}) = \frac{P(x^{(i)}|z^{(i)}=2)P(z^{(i)}=2)}{P(x)}$

$P(x^{(i)}) = P(x^{(i)}|z^{(i)}=j)P(z^{(i)}=j)$

$$w_j^{(i)} = \frac{P(x^{(i)}|z^{(i)}=j;\mu_j,\Sigma_j)P(z^{(i)}=j;\phi_j)}{\sum_{l=1}^{k} P(x^{(i)}|z^{(i)}=l;\mu_j,\Sigma_j)P(z^{(i)}=l;\phi_l)}$$

$N(x^{(i)};\mu_j,\Sigma_j)$

EM is guaranteed to monotonically increase

$\ell(\phi,\mu_1,\dots,\Sigma_1,\dots)$

But can converge to local maximum

or become degenerate ($\Sigma_j \to 0$)

Deep Learning (a.k.a neural networks)

$f: X \to y$

$\mathbb{R} \to \mathbb{R}^+$

$f(x) = \max(ax+b, 0)$

ReLu: Rectified Linear Unit

It is possible to model any function to arbitrary accuracy with a sufficiently complex network of such unit

Many types of neural networks are universal approximates

(Note: "output" layer may not use the same activation function as all others)

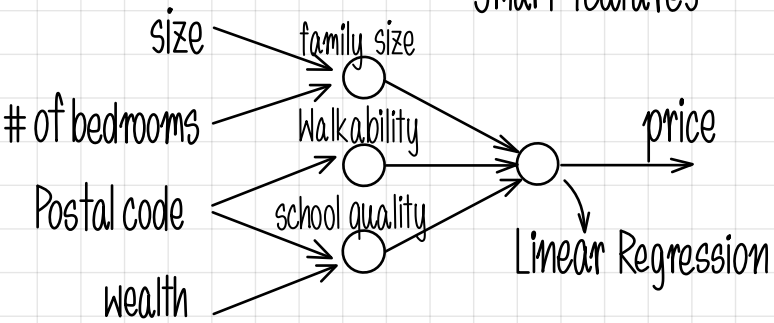
Example: Predict housing prices

Based on: Postal code, size, # of bedrooms, wealth of neighborhood

In data science applications, we often combine raw feature into "Smart features"

Postal code, wealth → Walkability, school quality

Smart features



Loan underwriting: Demographic info, credit info

Predict: will they pay?

Logistic Regression: works well if payers & non-payers are linearly seperable

Smart variable/feature: debt-to-income ratio

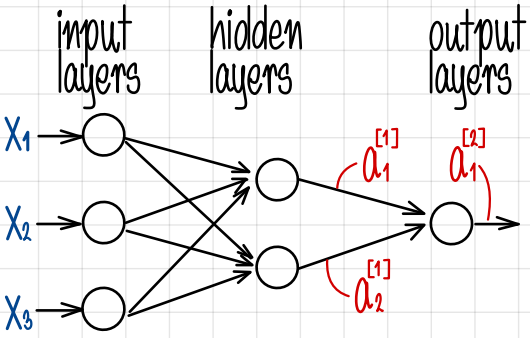
Deep neural networks perform end-to-end learning, from raw data input to output

Notation:

input feature: $x_1 + x_2 + \dots + x_n$

units in layers:

$a_i^{[j]}$ for the activation of the i^{th} units in the j^{th} layers



If Classification

$a_1^{[2]} = \frac{1}{1+e^{-w^T x+b}}$

Fully connected: all unit in one layer connect to all unit in next layer

$x_1 = a_1^{[0]}$ $x_2 = a_2^{[0]}$ ...

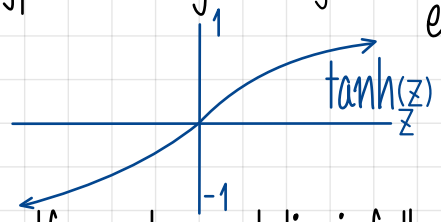
Activation - 2 steps:

1. compute Linear activation $z = w^T x + b$
2. compute activation function $a = \sigma(z)$

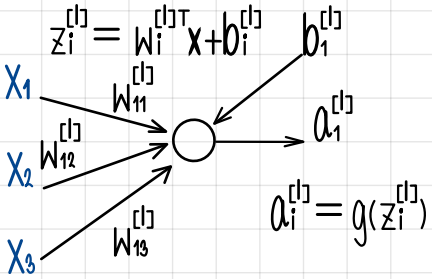
output of a Classifier : $\sigma(z) = \frac{1}{1+e^{-z}}$

Relu (mostly for hidden units) : $\max(z, 0)$

Hyperbolic tangent : $g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$



Feedforward computation in fully connected model



Efficiency Calculation :

Try to make everything a vector or matrix operation (most efficiency calculation)

Numpy

No! Do not do this!

```
for l in range(L):
    z = np.zeros((N[l],1))
    for i in range(N[l]):
        z[i] = b[i]
        for j in range(N[l-1]):
            z[i] = z[i] + w[i,j] * a[l-1,j]
        a[l,i] = g[z[i]]
```

Basic Linear Algebra Subroutines Library

Do this! one called the "bias" library

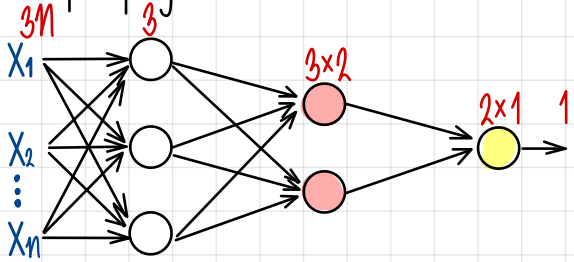
```
for l in range(L):
    z[l] = b[l] + w[l] * a[l-1]
    a[l] = g[z[l]]
```

Inside g(z) : matrix vector

```
def g(z):
    return 1/(1+np.exp(-z))
```

vector

Backpropagation



2 hidden Logistic sigmoid output

parameters : $3n+14$ (weight, bias)

initialization

$w_{jk}^{[i]} \sim N(0, 0.1)$

or xavierihe initialization

$w_{jk}^{[i]} \sim N(0, \sqrt{\frac{2}{n^{[i]} + n^{[i-1]}}})$

variance of activation is preserved across layers

Assume Relu for hidden units

Bias : small positive random or fixed values.

Output loss function for single example (Stochastic Gradient Descent)

$L(\hat{y}, y) = -[(1-y)\log(1-\hat{y}) + y \log \hat{y}]$

For all parameters, we want

$$w^{[l]} \leftarrow w^{[l]} - \alpha \frac{\partial L}{\partial w^{[l]}}$$

$$b^{[l]} \leftarrow b^{[l]} - \alpha \frac{\partial L}{\partial b^{[l]}}$$

Learning rate, Gradient

This in layer 3 :

$$\frac{\partial L}{\partial w^{[3]}} = \frac{\partial [-(1-y)\log(1-\hat{y}) - y \log \hat{y}]}{\partial w^{[3]}}$$

$$= -(1-y) \frac{\partial}{\partial w^{[3]}} \log(1 - g(w^{[3]} a^{[2]} + b^{[3]})) - y \frac{\partial}{\partial w^{[3]}} \log(g(w^{[3]} a^{[2]} + b^{[3]}))$$

$$= \frac{-(1-y)(-1) g'(w^{[3]} a^{[2]} + b^{[3]}) a^{[2]}}{1 - g(w^{[3]} a^{[2]} + b^{[3]})} - \frac{y g'(w^{[3]} a^{[2]} + b^{[3]}) a^{[2]}}{g(w^{[3]} a^{[2]} + b^{[3]})}$$

$g(z)$: sigmoid $\sigma(z)$

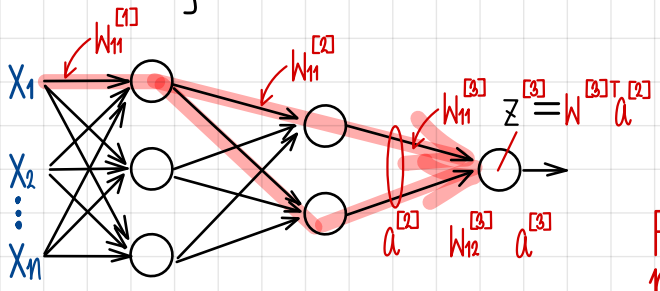
$\sigma(z)' = \sigma(z)(1-\sigma(z))$

$= (1-y)\sigma(w^{[3]} a^{[2]} + b^{[3]}) a^{[2]} - y(1 - \sigma(w^{[3]} a^{[2]} + b^{[3]})) a^{[2]}$

$$\frac{\partial L}{\partial w^{[3]}} = (a^{[3]} - y) a^{[2]}$$

Remember from Logistic Regression : $(\hat{y} - y)x$?

What about layer 2?



Chain Rule: $f(z)$ where $z = g(x)$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial z}\frac{\partial g}{\partial x}$$

$a^{[3]}$ is a function of $z^{[3]}$
$z^{[3]}$ is a function of $a^{[2]}$
$a^{[2]}$ is a function of $z^{[2]}$
$z^{[2]}$ depend on $w^{[2]}$

$$\frac{\partial \mathcal{L}}{\partial w_{ij}^{[2]}} = \frac{\partial \mathcal{L}}{\partial a^{[3]}}\frac{\partial a^{[3]}}{\partial z^{[3]}}\frac{\partial z^{[3]}}{\partial a^{[2]}}\frac{\partial a^{[2]}}{\partial z^{[2]}}\frac{\partial z^{[2]}}{\partial w_{ij}^{[2]}}$$

$$\frac{\partial \mathcal{L}}{\partial w^{[3]}} = \frac{\partial \mathcal{L}}{\partial a^{[3]}}\frac{\partial a^{[3]}}{\partial z^{[3]}}\frac{\partial z^{[3]}}{\partial w^{[3]}} = (a^{[3]} - y)\, a^{[2]}$$

Reused terms: "Delta" $\delta^{[3]}, \delta^{[2]}, \ldots$

$$\delta^{[3]} = \frac{\partial \mathcal{L}}{\partial a^{[3]}}\frac{\partial a^{[3]}}{\partial z^{[3]}}\frac{\partial z^{[3]}}{\partial w^{[3]}} \qquad \frac{\partial \mathcal{L}}{\partial w^{[3]}} = \delta^{[3]} a^{[2]}$$

$$\delta^{[2]} = \delta^{[1]}\frac{\partial z^{[3]}}{\partial a^{[2]}}\frac{\partial a^{[2]}}{\partial z^{[2]}} \qquad \frac{\partial \mathcal{L}}{\partial w^{[2]}} = \delta^{[2]} a^{[1]}$$

($\frac{\partial z^{[3]}}{\partial a^{[2]}} \to w^{[3]}$, $\frac{\partial a^{[2]}}{\partial z^{[2]}} \to g^{[2]}$)

Chain Rule for an effect on an output through multiple paths:

$y = f(z) \quad u = g(z)$

$$\frac{\partial y}{\partial x_i} = \sum_j \frac{\partial y}{\partial u_j}\frac{\partial u_j}{\partial x_i}$$

$$\frac{\partial \mathcal{L}}{\partial w_{ij}^{[2]}} = \sum_k \frac{\partial \mathcal{L}}{\partial a_k^{[2]}}\frac{\partial a_k^{[2]}}{\partial w_{ij}^{[2]}}$$

$$\delta^{[1]} = \sum_k \delta_k^{[2]}\frac{\partial z_k^{[2]}}{\partial a_j^{[1]}}\frac{\partial a_j^{[1]}}{\partial z_j^{[1]}}$$

($\frac{\partial z_k^{[2]}}{\partial a_j^{[1]}} \to w^{[2]}$, $\frac{\partial a_j^{[1]}}{\partial z_j^{[1]}} \to g^{[1]}$)

$$= \sum_k \underbrace{\frac{\partial \mathcal{L}}{\partial a^{[3]}}\frac{\partial a^{[3]}}{\partial z^{[3]}}\frac{\partial z^{[3]}}{\partial a_k^{[2]}}\frac{\partial a_k^{[2]}}{\partial z_k^{[2]}}}_{\delta^{[2]}}\frac{\partial z_k^{[2]}}{\partial a_j^{[1]}}\frac{\partial a_j^{[1]}}{\partial z_j^{[1]}}\frac{\partial z_j^{[1]}}{\partial w_{ij}^{[1]}}$$

FC BP (SGD: Single Pattern x, y)

Forward propagation

given x, y
$a^{[0]} \leftarrow x$
for l in $1 \ldots L$:
# linear activation
$z^{[l]} \leftarrow w^{[l]} a^{[l-1]} + b^{[l]}$
nonlinear activation
$a^{[l]} \leftarrow g^{[l]}(z^{[l]})$
$a^{[L]}$ is $\hat{y}$!

Linear
Relu
Sigmoid
Tanh
More exotic ones!

$\delta^{[L]} \leftarrow \frac{\partial \mathcal{L}}{\partial z^{[L]}}$ # $a^{[L]} - y$ for Logistic sigmoid

for l in $L \ldots 1$:
$\frac{\partial \mathcal{L}}{\partial w^{[l]}} = \delta^{[l]} a^{[l-1]T}$
$\frac{\partial \mathcal{L}}{\partial b^{[l]}} = \delta^{[l]}$
if $l > 1$:
$\delta^{[l-1]} \leftarrow g^{[l-1]\prime}(z^{[l-1]}) \circ w^{[l]} \delta^{[l]}$
$\mathrm{diag}(g^{[l-1]\prime}(z^{[l-1]})) \cdot w^{[l]} \delta^{[l]}$

What about Batch Gradient Descent
More accurate than SGD
More slow (in terms of data use) than SGD

Get best of Both worlds Mini Batch Gradient Descent
1. sample a subset of training set
2. calculate Gradients over the whole Mini Batch
3. take a step

Iteration: one step for a Mini Batch
Epoch: one pass through entire training set
Example: training set size 1000, Batch size 10
we say "100 Iteration per Epoch"

Momentum: Hyperparameter in SGD
(Along with learning rate, Batch size)
Aiming to reduce the noise due to sampling

Instantaneous Stochastic Gradient

optimum
Calculate a "velocity"

$V_{dW}^{[l]} \leftarrow \beta V_{dW}^{[l]} - (1-\beta)\frac{\partial J}{\partial W^{[l]}}$

Moving Average of the Stochastic steps

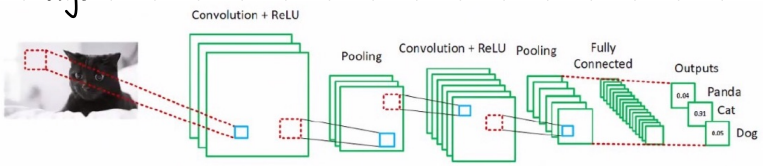
$W^{[l]} \leftarrow W^{[l-1]} - \alpha V_{dW}^{[l]}$

Smoothed Gradient

ADAM: Adaptive Sophisticated Adaptive Momentum
Optimize become another Hyperparameter

Convolutional Neural Networks (CNN)

Layers of a CNN



Reinforcement Learning

Supervised Learning
data driven → pair every input with target

Reinforcement Learning
data driven → reward, delayed
more "hand-off"
↘ autonomous robot ↘ creating VR environment
↘ game playing → Atari 2600 → video input
joystick output
reward score
David Silver → GO
→ online multiplayer games
Starcraft
Protein folding prediction ← paper

Markov decision process

A tuple $(S, A, \{P_{sa}\}, \gamma, R)$

S : Set of states

A : Set of actions (things the agent can do)

$\{P_{sa}\}$: States transition probabilities

For each states $s \in S$, each action $a \in A$,

P_{sa} gives a distribution over S

$P_{s_i a_j}(s_k)$: probabilities that when taking action a_j in state s_i that the government moves to state s_k

$\gamma \in [0,1]$ discount factor : how important is it to get reward early rather than late

$R : S \times A \to \mathbb{R}$ reward function

MDP (Markov decision process) :

Start in some state s_0

Agent chooses action $a_0 \in A$

Environment samples $s_1 \sim P_{s_0 a_0}$

[continue choosing actions, samplings states]

Total payoff : $R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \dots$

What is the best action to take in each state?

Policy is a function $\pi : S \mapsto A$

Execute the policy by taking, in States s, to action $a = \pi(s)$ (peterministic or randomized)

A policy π has a value function $V^\pi(s)$

$$V^\pi(s) = \mathbb{E}[R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \dots \mid s_0 = s; \pi]$$

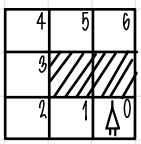
↑ the average of the expression over an infinite number of trials

Famous recursive relationship : Bellman equations

$$V^\pi(s) = \underbrace{R(s)}_{\text{immediate reward}} + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^\pi(s')$$

$$V^\pi(s) = R(s) + \gamma \mathbb{E}[R(s_1) + \gamma R(s_2) + \dots \mid \pi]$$

If $|s| < \infty$, write down $V^\pi(s)$ for each $s \in S$, R, P_{sa}, π fixed & known, $|s|$ Linear equation in $|s|$ unknowns



$S = \{0, 1, 2, 3, 4, 5, 6\}$

$A = \{L, R, U, D\}$

$\gamma = 0.9$

$R = 0 \mapsto 0,\ 1 \mapsto 0,\ \vdots,\ 5 \mapsto 0,\ 6 \mapsto 100$

$P_{SA} = 0, R \mapsto [1, 0, \dots, 0]$ $\quad 1, U/D \mapsto [0, 1, 0, \dots, 0]$

$0, U \mapsto [1, 0, \dots, 0]$ $\quad 1, L \mapsto [0, 0.1, 0.9, 0, \dots]$

$0, D \mapsto [1, 0, \dots, 0]$ $\quad 1, R \mapsto [0.9, 0.1, 0, \dots]$

$0, L \mapsto [0.1, 0.9, 0, \dots, 0]$

π : 0 ↦ L 1 ↦ L 2 ↦ U 3 ↦ U 4 ↦ R 5 ↦ R 6 ↦ R $S \mapsto A$

$P_{0,L}(0) = 0.1$

$P_{1,L}(0) = 0.9$

$V^\pi(0) = 0 + 0.9 \cdot 0.1\, V^\pi(0) + 0.9 \cdot 0.9 V^\pi(1)$

$V^\pi(1) = 0 + 0.9 \cdot 0.1\, V^\pi(1) + 0.9 \cdot 0.9 V^\pi(2)$

$V^\pi(2) = 0 + 0.9 \cdot 0.1\, V^\pi(2) + 0.9 \cdot 0.9 V^\pi(3)$

$V^\pi(3) = 0 + 0.9 \cdot 0.1\, V^\pi(3) + 0.9 \cdot 0.9 V^\pi(4)$

$V^\pi(4) = 0 + 0.9 \cdot 0.1\, V^\pi(4) + 0.9 \cdot 0.9 V^\pi(5)$

$V^\pi(5) = 0 + 0.9 \cdot 0.1\, V^\pi(5) + 0.9 \cdot 0.9 V^\pi(6)$

$V^\pi(6) = 100$

$0.91\, V^\pi(0) - 0.81 V^\pi(1) = 0$

$0.91\, V^\pi(1) - 0.81 V^\pi(2) = 0$

$0.91\, V^\pi(2) - 0.81 V^\pi(3) = 0$

$0.91\, V^\pi(3) - 0.81 V^\pi(4) = 0$

$0.91\, V^\pi(4) + 0.81 V^\pi(5) = 0$

$0.91\, V^\pi(5) + 0.81 V^\pi(6) = 0$

$V^\pi(6) = 1$

$Ax = b \qquad x = A^{-1}b$

$$A = \begin{bmatrix} 0.91 & -0.81 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.91 & -0.81 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.91 & -0.81 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.91 & -0.81 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.91 & 0.81 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.91 & 0.81 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$x = \begin{bmatrix} V^\pi(0) \\ \vdots \\ V^\pi(6) \end{bmatrix}$$

Optimal value function $V^*(s)$

$V^*(s) = \max_\pi V^\pi(s)$

$V^*(s) = R(s) + \max_{a \in A} \gamma \sum_{s' \in S} P_{sa}(s') V^*(s')$

expected future reward

immediate reward

Optimal policy $\pi^* : S \mapsto A$

$\pi^*(s) = \arg\max_a \sum_{s' \in S} P_{sa}(s') V^*(s')$

Value Iteration :

$V^*(s) = V^{\pi^*}(s) \geq V^\pi(s)$

For each state s (estimate, update iteratively)

  $V(s) \leftarrow 0$

Repeat untill convence

  for each state s

    $V(s) \leftarrow R(s) + \max_{a \in A} \gamma \sum_{s' \in S} P_{sa}(s') V(s')$

Theorm : V convences to V^*

Policy Iteration :

Initialize π randomly

Repeat untill convence

  $V(s) \leftarrow V^\pi(s)$ (solve linear system)

  for each state s

    $\pi(s) \leftarrow \arg\max_a \sum_{s' \in S} P_{sa}(s') V(s')$

Theorm : π convences to π^*, V convences to V^*

So far : RL is easy

When # of state is small, # of action is small, P_{sa} is given, R(s) is given, state is observable

R(s) is immediatly observable

In real world problems, P_{sa} is not given

Two ways to deal with unknown environment :

- Model-Based methods estimate P_{sa} via trial and error
- Model-Free methods avoid using P_{sa} in the algorithm

  If world is discrete state & discrete action, just use a table to keep trade of how many times are a_0 to each state s' when executing action a in state s

To explore, we typically use a simply randomized algorithm called ε-greedy to decide what to do

ε-greedy policy:
with probability ε, take a uniform random action
with probability 1-ε, take the current estimate best action

TD Methods (Temporal-Difference) vs Monte Carlo

TD Prediction

similar to policy evaluation estimate V^π

Monte carlo RL method wait until real return are observed:

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$$

G_t: actual return obtained in state S_t following π to end of episode

TD Methods approximate this by performing updates

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

$R_{t+1} + \gamma V(S_{t+1})$: noisy

Tabular TD(0) for V^π

Input policy π, LR α
Initialize $V(s)$ for all $s \in S$ arbitrarily except $V(\text{terminal}) = 0$
For each step in episode:
$a \leftarrow \pi(S)$
$S' \sim P_{sa}$
$V(S) \leftarrow V(S) + \alpha[R(S') + \gamma V(S') - V(S')]$

Convergence is guaranteed if α is small and decrease overtime
We want a model free to method to find π^* when P_{sa} is unknown

Control Problem

SARSA on-policy to policy iteration method instead of estimating V, we estimate Q (action value function):
$q_\pi(s,a)$: expected discounted return after executing action a in state s, following π

$$Q(S_t,a_t) \leftarrow Q(S_t,a_t) + \alpha[R(S_{t+1}) + \gamma Q(S_{t+1},a_{t+1}) - Q(S_t,a_t)]$$

$(S_t, a_t, r_t, S_{t+1}, a_{t+1})$

TD Methods: SARSA Updated Equation

Tabular Methods $Q(s^t, a^t)$: Table lookup

What about infinite state/action spaces?

partial observablity
infinite
discrete continuous ranges
output a parameter distribution

$P_{sa} = \frac{\#\text{ of times we took action a in state s and got to s'}}{\#\text{ of times we took action a in state s}}$

Introduce Q-Learning

Deal with infinite/partial state observation
Maximization bias, double Q-Learning

SARSA

$$Q(S_t,a_t) \leftarrow Q(S_t,a_t) + \alpha[R(S_{t+1}) + \gamma Q(S_{t+1},a_{t+1}) - Q(S_t,a_t)]$$

$(S_t, a_t, r_t, S_{t+1}, a_{t+1})$

a_{t+1}: an action taken according to π

We must execute a_t, a_{t+1} before updating

Q-Learning

$$Q(S_t,a_t) \leftarrow Q(S_t,a_t) + \alpha[R(S_{t+1}) + \gamma \max_a Q(S_{t+1},a) - Q(S_t,a_t)]$$

ε-greedy: $p(1-\varepsilon)$ follow Q, $p(\varepsilon)$ pick random action

$\max_a Q(S_{t+1},a)$: the best possible action according to current Q function

exploration will select action a_{t+1} using ε-greedy
$a_{t+1} \neq a$

Q-Learning is "off-policy" - learns the Optimal policy rather than the Optimal ε-greedy policy

V^*
$q_\pi(s,\pi)$

Maximization Bias

Q-Learning: select a using $\max_a Q(S_{t+1},a)$
the Maximization leads to bias
consider problem with a state s for which $Q(s,a)$ is truly 0 for all a
However, due to noise/random initial values, $\max_a Q(s,a)$ will always be positive
∴ $V(s)$ is always overestimated

Double Q-Learning

$\forall s,a$, initialize $Q_1(s,a), Q_2(s,a)$ arbitrarily, except $Q(\text{terminal},\cdot) = 0$
For each episode:
Initialize s
For each step in the episode:
choose action a, observing r, s'
with probability 1/2:

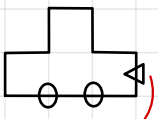
$$Q_1(s,a) \leftarrow Q_1(s,a) + \alpha[R + \gamma Q_2(s', \arg\max_a Q_1(s',a)) - Q_1(s,a)]$$

else:

$$Q_2(s,a) \leftarrow Q_2(s,a) + \alpha[R + \gamma Q_1(s', \arg\max_a Q_2(s',a)) - Q_2(s,a)]$$

Eliminates Maximization Bias

Dealing with infinite & partially observable state spaces
Example: self-driven vehicle



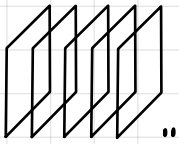
$\phi: S \rightarrow O$
action state

↳ size of space: infinite
environment/state is only partially observable via ovr sensors
↳ use sensor measurements as a proxy for the state

Assume there exists an observation function Q that tranduces the world to a set of sensor measurements with noise

Example sensor: monocular image sequence
sample preprocessing to make observation useful

Mnih et al. (2013) (Deepmind/Goolge)
Atari Games (ALE/Gym)

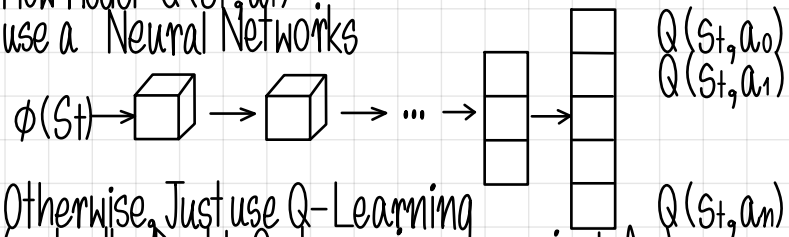


representation of state s^t is a stack of 4 grayscale frames

frames 1 2 3 4 5 ...
colors 210×160 images

COMPACT REPRESENTATION OF ACTUAL STATE
(OPTICAL FLOW CAN GIVE INFO ABOUT TRAJECTORY / VELOCITY). (28,224 elements!)
↳ 256 values per element

How model $Q(s_t, a_t)$?
use a Neural Networks



Otherwise, Just use Q-Learning
(actually, Double Q-Learning or a variant of π)

Replay Buffer: Save
s_t, a_t, r_t, a_{t+1}, done
Tuples in a Replay Buffer. Q-Learning is dine in batches sampled from the Buffer