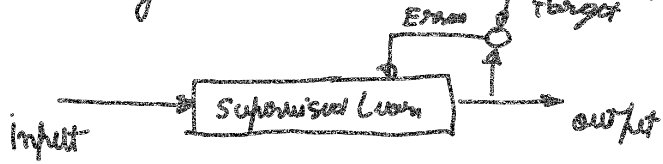
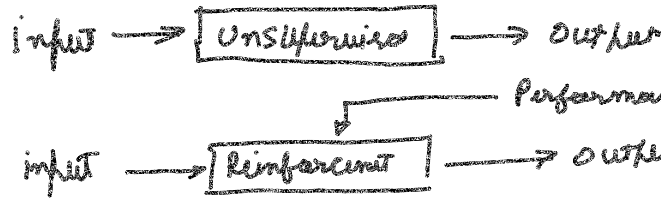
Math of Intelligence - 1.

Machine learning is mathematical optimization.





Easiest way is to learn from labeled data.

Domain X ; Every point of X has features

Label set Y

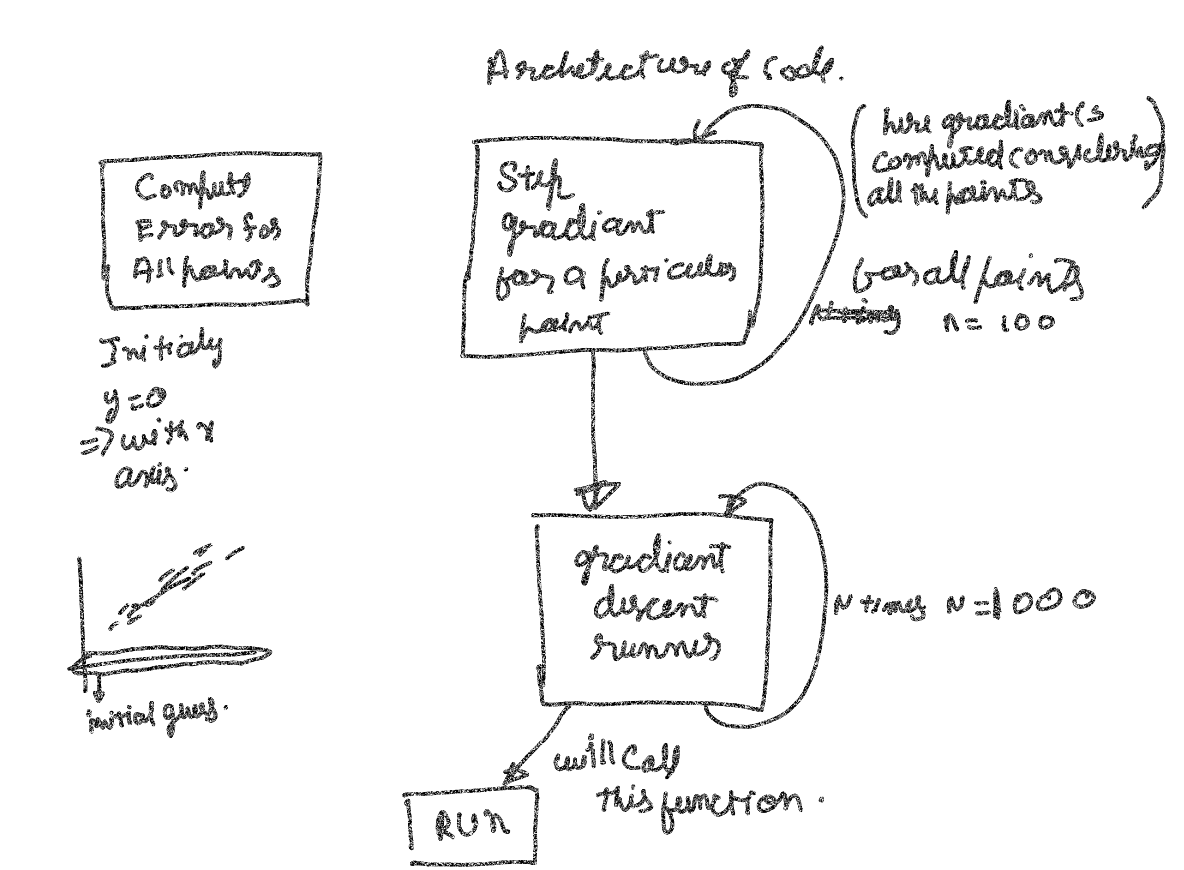
Data, a set of labeled examples $S = \{(x_1, y_1) \ldots (x_n, y_n)\} \subseteq X \times Y$

Output → a prediction rule / hypothesis.

Performance metric. → Error^2

we keep minimizing Error.

Architecture of code.



Support Vector Machine

used mainly for classification

we will create a hyperplane. to seperate a data

The dimension of a hyper plane is $n-1$

input space ⟶ Feature Space (kernel trick)

Loss function → Hing loss function.

mainly comes to maximum-margin classification.

$$c(x, y, f(x)) = (1 - y * f(x))_+$$

$$c(x, y, f(x)) = \begin{cases} 0 & \text{if } y * f(x) > 1 \\ 1 - y * f(x) & \text{else.} \end{cases}$$

objective function

$$\min_n \lambda \|w\|^2 + \sum_{i=1}^{n} (1 - y_i (x_i w))_+$$

regularisation term → its a tuning knob

If this is too large it might result in overfit if not underfit.

$$\frac{\delta}{\delta w_k} \lambda \|w\|^2 = 2\lambda w_k$$

$$\frac{\delta}{\delta w_k} (1 - y(x_i, w))_+ = \begin{cases} 0 & \text{if } y_i (x_i w) \geq 1 \\ -y_i x_{ik} & \text{else.} \end{cases}$$

gradient is a vector which consists of change in it contains direction of movement

updating weight $w = w + \eta (-2\lambda w)$

Gradient is a vector whose components contains partial derivatives all they represent the changes

Second Order Optimization

function → represents mapping

Machine learning → process of optimization.

Gradient Descent

Step 1 → Define a model

Step 2 → Define a loss fn. $\frac{1}{N} \Sigma (y - y_i)^2$

Step 3 → compute the gradient

use the gradient to update the weights.

all these are first order optimization

next we have

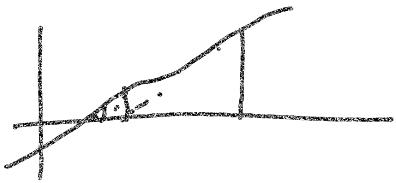
Second order optimization.

* They do not ignore curvature of the surface.
* Better performance in a single optimization step.

Newtons Method

* Finding the roots
* Optimization.

First lets code root finding.



keep iteratively find where $f' = 0$

In optimization method we will go to find where the derivative of the function is 0.

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

The minimizer of Quadratic f'' is used.

Here we will replace first derivative with

* gradiant
* Second derivative with Hessian. (matrix of second order derivatives. Describes a local curvature of the function).

→ Derivative → $\partial f / \partial x$

→ Gradient → $\left[\partial f/\partial x_1 , \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial x_3} \cdots \right]^T$

→ Hessian → $\begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots \\ \vdots & \ddots & \\ \cdots & & \partial^2 f / \partial x_n \end{bmatrix}$ → Jacobian → $\begin{bmatrix} \partial f/\partial x_1 & \partial f/\partial x_2 & \cdots \\ \vdots & \ddots & \\ & & \partial f/\partial x_n \end{bmatrix}$

first order → uses Jacobian

second order → uses Hessian

newtons method. is for second order method.

Architecture of SVM code

x-val
y-val
bias

associated Label
y

Loss function.

$$C(x, y, f(x)) = (1 - y * f(x))_+$$

C → Loss fn x → sample y → true label

f(x) → predicted label

$$c(x, y, f(x)) = \begin{cases} 0, & \text{if } y * f(x) \geq 1 \\ 1 - y * f(x) & \text{else} \end{cases}$$

objective fn

$$\min_w \underbrace{\lambda \|w\|^2}_{\text{regularizer}} + \sum_{i=1}^{n} \underbrace{(1 - y_i \langle x_i w \rangle)_+}_{\text{Loss}}$$

Minimizing the objective by getting its gradients for gradient descent.

$$\frac{\delta}{\delta w_k} \lambda \|w\|^2 = 2\lambda w_k$$

$$\frac{\delta}{\delta w_k} (1 - y_i \langle x_i w \rangle)_+ = \begin{cases} 0 & \text{if } y_i \langle x_i w \rangle \geq 1 \\ -y_i x_{ik} & \text{else} \end{cases}$$

⇒ If we misclassified the sample

→ Then we update weight vector w using gradients of both terms

→ Else if classified correctly we will just update w by the gradient of the regulisor.

Miss Classification Condition

$$y_i \langle x_i, w \rangle < 1$$

→ update rule for our weights (miss classified)

$$w = w + \eta (y_i x_i - 2\lambda w)$$

$\eta \rightarrow$ learning rate (This is the length of steps algorithm makes down the gradient of the error curve)

$\lambda \rightarrow$ regulaisor.

$\eta \rightarrow$ high $\rightarrow$ algorithm might overshoot

$\eta \rightarrow$ low $\rightarrow$ algorithm takes long time to converge

---

Regulaisor will control trade off between achieving low training error and low Testing Error (ie for giving ability to generalise the classifier to unseen data) as regularising parameter we will choose $1/\text{epochs}$ such that this number will decrease as no. of epoch increase.

update rule for weights when classified correctly.

$$w = w + \eta(-2\lambda w)$$