Gradient descent

↳ At least a local minimum can be found

change $\vec{\theta}$ gradually as

$\vec{\theta} \to \vec{\theta} + \Delta\vec{\theta}$

$$\Delta\vec{\theta} = -\varepsilon \frac{\partial J}{\partial \theta}$$

ε learning rate

$$\left(\Delta\theta_j = -\varepsilon . \frac{\partial J}{\partial \theta_j}\right)$$

$J(\vec{\theta})$ decreases if ε is sufficiently small

Decomposing the gradient

Cost function

$$g(\omega) = \sum_{p=1}^{P} h(\omega, x_p, y_p)$$

ie sum of individual costs over each data points

for eg with Least square

$$h(\omega, x_p, y_p) = (x_p^T w - y_p)^2$$

or with two class classification

$$h(\omega, x_p, y_p) = \log\left(1 + e^{-y_p x_p^T w}\right)$$

This is simple observation, that machine learning cost functions can be decomposed over individual data points, is important because it allows us to write the gradient itself as a summation

of the gradients of each of the P summands

$$\nabla g(\omega) = \nabla\left(\sum_{p=1}^{P} h(\omega, x_p, y_p)\right) = \sum_{p=1}^{P} \nabla h(\omega, x_p, y_p)$$

Thus when minimizing any cost function via gradient descent we can think about the K^{th} gradient step in terms of these individual gradients as

$$\omega^K = \omega^{K-1} - \alpha_K \nabla g(\omega^{K-1})$$

$$= \omega^{K-1} - \alpha_K \sum_{p=1}^{P} \nabla h(\omega^{K-1}, x_p, y_p)$$

where α_K is - as always - an appropriately chosen length.

for eg

$$\vec{\theta} \to \vec{\theta} + \Delta\vec{\theta} \qquad \Delta\vec{\theta} = -\varepsilon \frac{\partial J}{\partial \theta}$$

ε : learning rate

Cost function $J(\vec{\theta}) = \frac{1}{2} \sum_{k \text{ all samples}} \left(y^{(k)} - f(\vec{x}^{(k)}, \vec{\theta})\right)^2$

$$\frac{\partial J}{\partial \theta_i} = -\sum_{k \text{ all samples}} \left(y^{(k)} - f(\vec{x}^{(k)}; \vec{\theta})\right) \cdot \frac{\partial f(\vec{x}^{(k)}, \vec{\theta})}{\partial \theta}$$

Vanilla or Batch GD

The stochastic gradient descent

The idea of taking a sequence of gradient steps in each data point (as opposed to full / batch gradient step over the entire dataset) leads to a procedure called stochastic gradient descent.

The k^{th} iteration of stochastic gradient descent, sometimes called epoch, consist of P sequential point wise gradient steps written as

$$\omega^{k,p} = \omega^{k,p-1} - \alpha_k \nabla h(\omega^{k,p-1}, x_p, y_p) \qquad p = 1, \ldots, P. \qquad \text{(SGD)}$$

In analogy with the k^{th} batch gradient step in

$$\omega^k = \omega^{k-1} - \alpha_k \nabla g(\omega^{k-1})$$
$$= \omega^{k-1} - \alpha_k \sum_{p=1}^{P} \nabla h(\omega^{k-1}, x_p, y_p)$$

Batch gradient Descent.

Here we have used the double superscript $\omega^{k,p}$, which reads "the p^{th} individual step of the k^{th} stochastic gradient descent iteration". In this notation point of the k^{th} iteration is written as $\omega^{k,0}$ the corresponding sequence of P individual gradient steps as $(\omega^{k,1}, \omega^{k,2}, \ldots, \omega^{k,P})$ and the final output of the k^{th} iteration. (ie the P^{th} stochastic step) as

$w^{k,p} = w^{k+1,0}$. After completing the k^{th} iteration we perform the $(k+1)^{th}$ iteration by cycling through the data again, taking individual gradient steps for $p = 1, \ldots, P$.

It can be regarded as a stochastic approximation of gradient descent optimization, since it replaces the actual gradient (calculated from the entire data set) by as estimate there of (calculated from a randomly selected subset of the data).

Especially in high dimensional optimization problems this reduces the very high computational burden, achieving faster iterations in exchange for lower convergence rate.

Mini batch GD randomly choose a batch $\{\vec{x}^{(l)}, y^{(l)}\}_{l \in \Delta}$

$$\Delta\vec{\theta} = \frac{-\varepsilon \partial J_\Delta}{\partial \vec{\theta}}$$

$$J_\Delta(\vec{\theta}) = \frac{1}{2} \sum_{l \in \Delta} J(y^{(l)} - f(\vec{x}^{(l)}; \vec{\theta}))^2$$

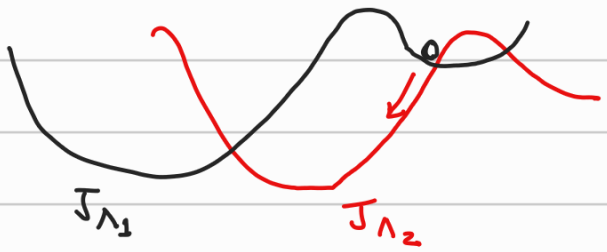
Advantages of Mini Batch GD

① If the batch is sufficiently large, J and J_Δ should not be different too much.
(as long as few outliers are dominant)

By using J_Λ instead of J, we can reduce the cost at each step.

(eg total 10K samples in training data batch size 8, 16, or 32)

The location of minima depends on batch

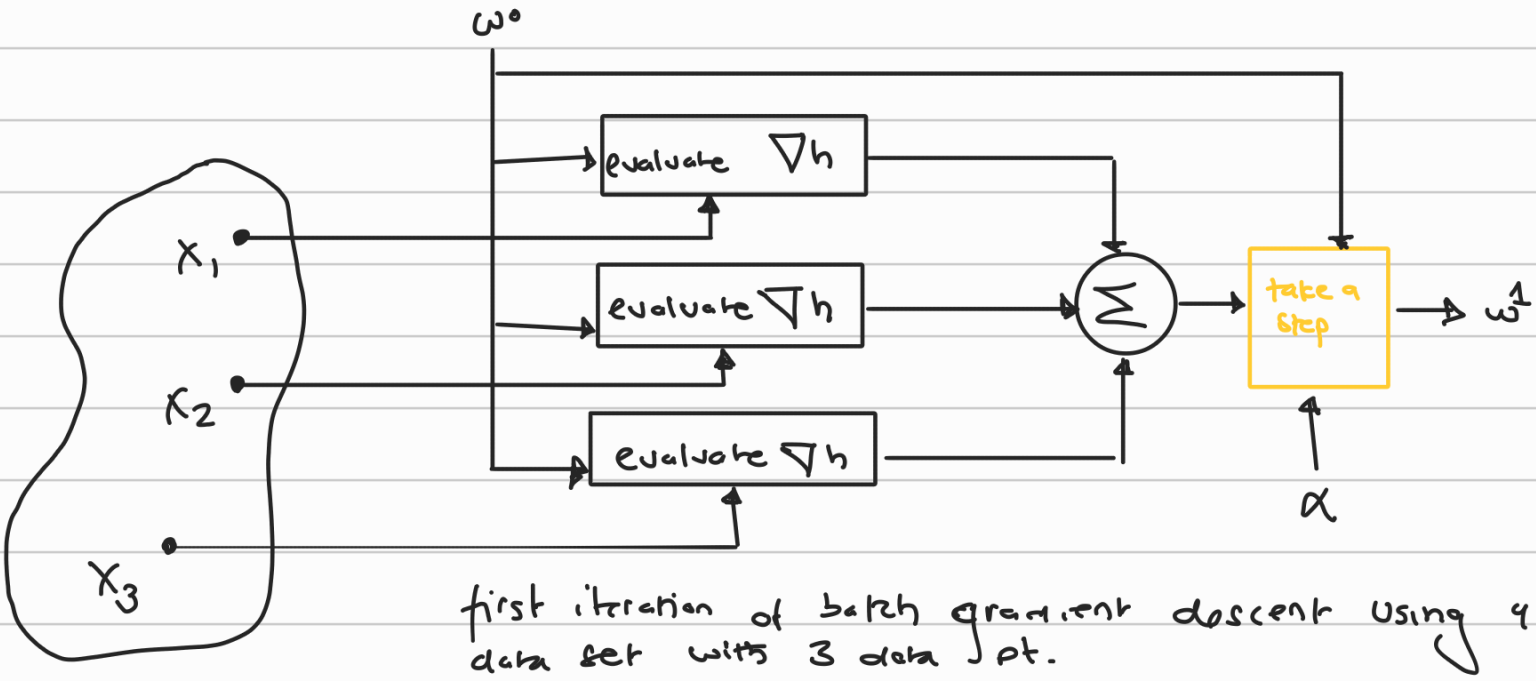


⇒ It is less likely to be trapped in a local minimum.

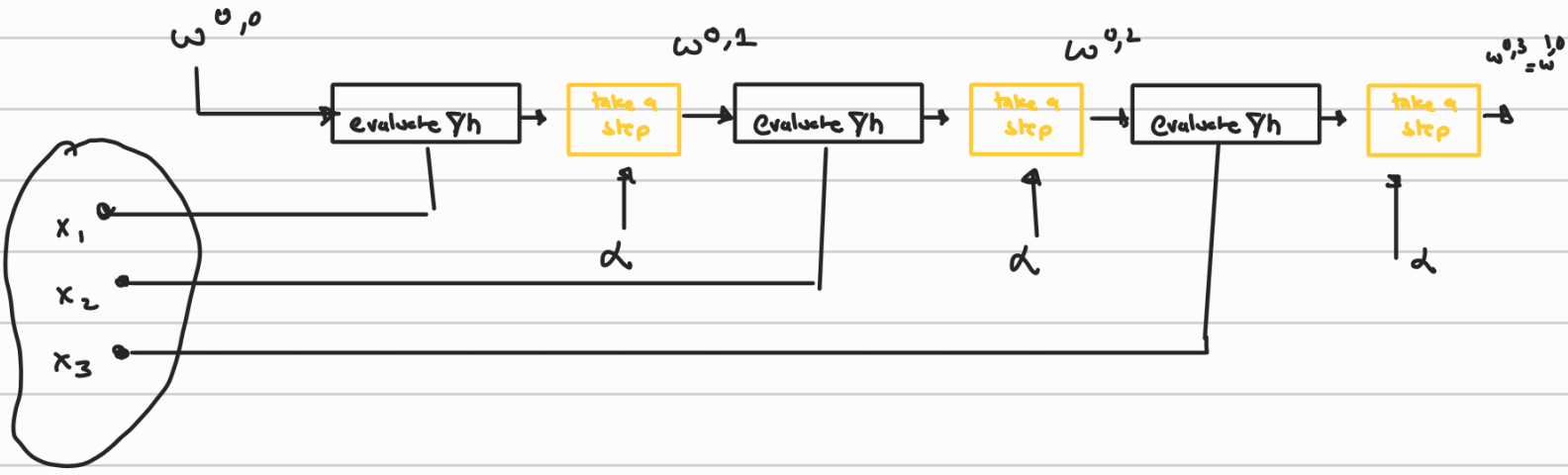
At least, we have a better chance to find a local minimum which is robust under a batch choice, which is likely to be a good one

③ (similar to 2)

Implicit regularization effect preventing overfitting and improve generalization performance.



first iteration of batch gradient descent using 4 data set with 3 data pt.



first iteration of stochastic gradient descent, using a data set

The standard or batch gradient descent we often use "step" and "iteration" interchangably ie each iteration consists of one full gradient step in all P data points simultaneously

Conversely with stochastic method we refer to single "iteration" or epoch" as consisting of all P individual gradient steps one in each data point, executed sequentially for $p = 1, \ldots, P$.

The mini batch gradient descent.

The cost function

$$g(w) = \sum_{p=1}^{P} h(w, x_p, y_p)$$

Can be written more generally as

$$g(w) = \sum_{j=1}^{J} \sum_{p \in \Omega_j} h(w, x_p, y_p)$$

where $\Omega_1, \Omega_2, \Omega_3, \ldots \Omega_J$ denote a certain partitioning of set $\{1, 2, \ldots P\}$ into J index set.

These index sets divide the dataset into J subset, each called a minibatch, where every data point belongs to one and only one mini-batch.

Analogous to

$$\nabla g(w) = \nabla\left(\sum_{p=1}^{P} h(w, x_p, y_p)\right)$$

$$= \sum_{p=1}^{P} \nabla h(w, x_p, y_p)$$

we have

$$\nabla g(w) = \nabla\left(\sum_{j=1}^{J} \sum_{p \in \Omega_j} h(w, x_p, y_p)\right)$$

$$= \sum_{j=1}^{J} \nabla\left(\sum_{p \in \Omega_j} h(w, x_p, y_p)\right)$$

where the gradient is now decomposed over each mini-batch (as opposed to each data point) and mini batch gradient descent is then the algorithm where in we take gradient steps sequentially using each mini-batch. Ideally we want all mini-batch to have the same size. - a parameter we call the batch size. Or be a equally sized as possible when J does not divide P.

Notice, a batch size 1 turns mini-batch gradient descent into stochastic gradient descent, whereas a batch size of P turns it into the standard or batch gradient descent.