Lecture 8 EE 559 Feb 5, 2024

Machine Learning I: Supervised Methods

B. Keith Jenkins

Announcements

- Homework 3 is due Friday.
- Lecture poll questions will count towards participation grade starting Wednesday (weather permitting)
- No lecture poll questions today

Today's lecture

- Perceptron: graphical example
  - Using basic sequential GD
- Gradient descent (GD) algorithms
  - Sequential, stochastic
  - Batch, mini-batch
  - General GD and perceptron application
- Example variant of perceptron
  - Perceptron with margin
- Perceptron convergence
  - Convergence conditions
- Multiclass perceptron

Perceptron Learning Algorithm – Basic sequential GD version

$(\underline{x}^{(t)}, \underline{w}^{(t)})$

Update $\underline{w}(i)$ after each data point:

$$\boxed{\begin{gathered}\underline{w}(i+1) = \underline{w}(i) + \eta(i) \, [\![g(z_n \underline{x}_n) \leq 0]\!] \, z_n \underline{x}_n , \quad \eta(i) \geq 0 \;\; \forall i \\ \text{Stop when } \sum_{n=1}^{N} [\![g(z_n \underline{x}_n) \leq 0]\!] \, z_n \underline{x}_n = 0 \\ (\text{Stop when } [\![g(z_n \underline{x}_n) \leq 0]\!] = 0 \text{ for all of the last } N \text{ iterations.})\end{gathered}}$$

Tip: best to randomly shuffle training data (over all classes) before iterating.

Graphical example of perceptron learning

Let $\eta(i) = 1 \ \forall i$; augmented space; reflected data points.

→ Use basic sequential GD

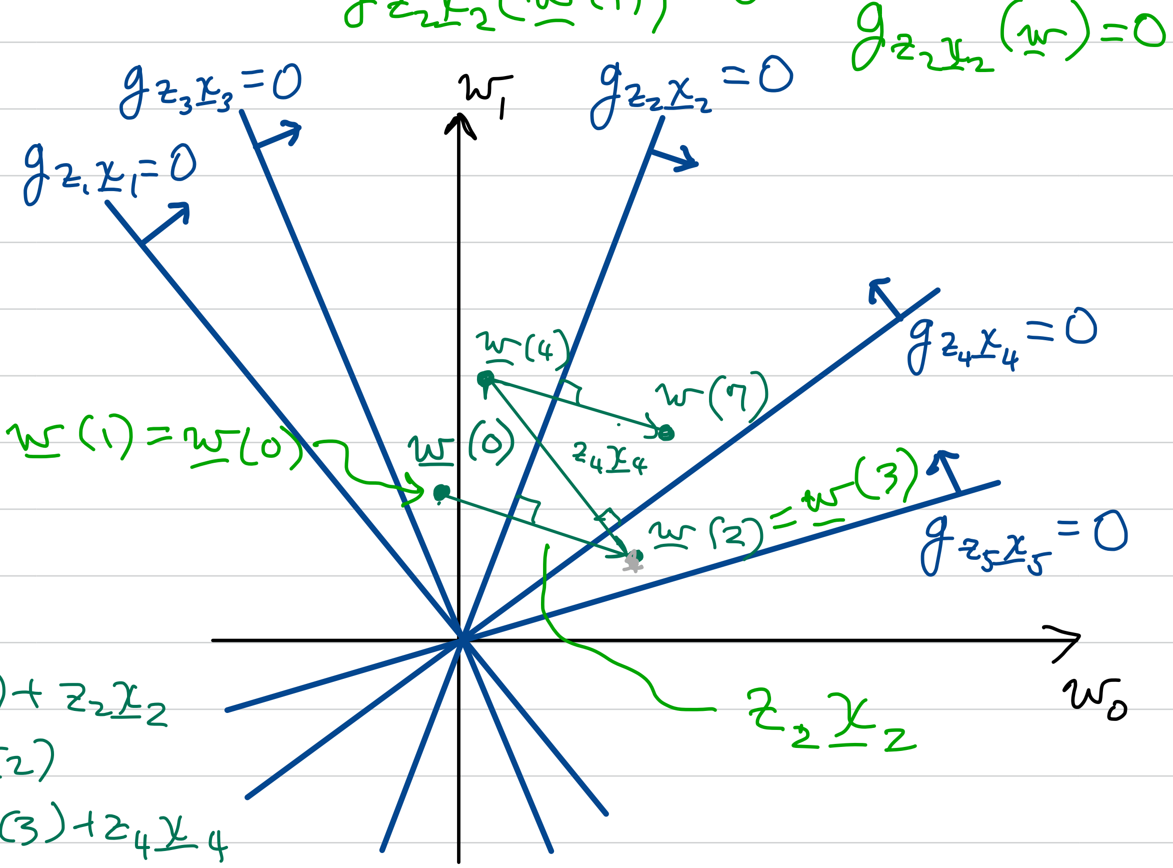
```
Randomly shuffle data point indeces
Initialize $\underline{w}(0)$.
If $\underline{w}^T z_n \underline{x}_n \le 0$
    $\underline{w}(i+1) = \underline{w}(i) + z_n \underline{x}_n$
Else
    $\underline{w}(i+1) = \underline{w}(i)$
```

Comment: $z_n \underline{x}_n \perp g_{z_n \underline{x}_n}(\underline{w}) = 0$

$i=0$: $g_{z_1 \underline{x}_1}(\underline{w}) > 0$.

$i=1$: $g_{z_2 \underline{x}_2}(\underline{w}(1)) < 0$.

Iteration i	Data pt. $\underline{x}_n$	$\underline{w}(i+1)$
0	$\underline{x}_1$	$\underline{w}(1) = \underline{w}(0)$
1	$\underline{x}_2$	$\underline{w}(2) = \underline{w}(1) + z_2 \underline{x}_2$
2	$\underline{x}_3$	$\underline{w}(3) = \underline{w}(2)$
3	$\underline{x}_4$	$\underline{w}(4) = \underline{w}(3) + z_4 \underline{x}_4$
4	$\underline{x}_5$	$\underline{w}(5) = \underline{w}(4)$
5	$\underline{x}_1$	$\underline{w}(6) = \underline{w}(5)$
6	$\underline{x}_2$	$\underline{w}(7) = \underline{w}(6) + z_2 \underline{x}_2$



$g_{z_2 \underline{x}_2}(\underline{w}) = 0$
$g_{z_3 \underline{x}_3} = 0$
w_1
$g_{z_2 \underline{x}_2} = 0$
$g_{z_1 \underline{x}_1} = 0$
$g_{z_4 \underline{x}_4} = 0$
$\underline{w}(4)$
$\underline{w}(7)$
$\underline{w}(1) = \underline{w}(0)$
$\underline{w}(0)$
$z_4 \underline{x}_4$
$\underline{w}(2) = \underline{w}(3)$
$g_{z_5 \underline{x}_5} = 0$
w_0
$z_2 \underline{x}_2$

$g_{z_n \underline{x}_n} = 0$ means $g_{z_n \underline{x}_n}(\underline{w}) = 0$

No updates in next 5 iterations. $\Rightarrow$ halt. $\hat{\underline{w}} = \underline{w}(7)$.

p.4

EE 559 Jenkins

Summary of Gradient Descent Algorithms

Spring 2024

1. General case

For unconstrained minimization of convex, differentiable $J(\underline{w})$.
Let i = iteration index.

Batch (true) gradient descent (GD)

Initialize $\underline{w}(0)$

Iterate until convergence

$$\underline{w}(i+1) = \underline{w}(i) - \eta(i)\nabla_{\underline{w}} J(\underline{w}) \,, \quad \text{in which } \eta(i) \geq 0 .$$

For sequential and stochastic GD

If $J(\underline{w})$ can be written:

$$J(\underline{w}) = \sum_{n=1}^{N} J_n(\underline{w})$$

in which each $J_n(\underline{w})$ is a function of data point $\underline{x}_n$ but no other data points, then we can define sequential gradient descent and stochastic gradient descent algorithms as follows.

Basic Sequential GD

Randomly shuffle the order of training data points

Initialize $\underline{w}(0)$

Define seq_epoch m as:

For n in $\{1,2,\cdots,N\}$ $m =$ epoch number $= 1,2,3,\cdots$

$$i = (m-1)N + n - 1$$

$$\underline{w}(i+1) = \underline{w}(i) - \eta(i)\nabla_{\underline{w}} J_n(\underline{w}), \quad \eta(i) \geq 0$$

Iterate seq_epoch over $m = 1,2,3,\cdots$ until convergence.

Stochastic GD – variant 1 (randomly shuffle before each epoch)

Initialize $\underline{w}(0)$

Iterate over $m = 1,2,3,\cdots$ until convergence:

(i) Randomly shuffle dataset

(ii) Perform seq_epoch m

Stochastic GD – variant 2 (choose each data point randomly, with replacement)

Initialize $\underline{w}(0)$

Iterate over $i = 0,1,2,\cdots$ (data points) until convergence:

(i) Randomly pick a training data point (with replacement); call it $\underline{x}_n$

(ii) Perform single-sample update:

$$\underline{w}(i+1) = \underline{w}(i) - \eta(i)\nabla_{\underline{w}} J_n(\underline{w}), \quad \eta(i) \geq 0$$

Mini-Batch GD – variant 1 (compromise between batch GD and stochastic GD)

Initialize $\underline{w}(0)$, M

Iterate until convergence:

(i) Pick M data points out of N points in the training set, at random

(ii) Perform batch GD on the M data points (1 iteration)

(iii) Replace the data points to the training set

ʃ

Mini-Batch GD – variant 2 (splitting into non-overlapping mini-batches)

Initialize $\underline{w}(0)$, M

Iterate until convergence (below is one epoch):

(i) Shuffle the data order (with labels attached). This is optional, but is most commonly done.

(ii) Partition the training data into B mini-batches, each of size M data points. This can be done by taking the first M data points for mini-batch 1, the second M data points for mini-batch 2, etc. Note that your last mini-batch may have fewer than M data points if N is not an integer multiple of M.

(iii) For $b = 1, 2, \ldots B$: Perform batch GD on the M data points from mini-batch b (1 iteration)

This is the way that tensorflow.model.fit works with shuffling done by default.

Notes – general GD (any of the above GD algorithms)

1. *Halting criterion* may depend on choice of $J(\underline{w})$; one generic criterion is when the norm of the update $\eta(i)\nabla_{\underline{w}}J(\underline{w})$ (batch GD), or norm of $\eta(i)\nabla_{\underline{w}}J_n(\underline{w})$ accumulated over a number of data points in some way (stochastic GD), is below some threshold.
2. *Choice of* $\eta(i)$ will depend on $J(\underline{w})$ and possibly also the data
3. *Initial condition* $\underline{w}(0)$ is commonly chosen so that each component $w_j(0)$ is random (iid) but small in magnitude.

2. Perceptron Learning case

Batch gradient descent (GD)

Initialize $\underline{w}(0)$

Iterate until convergence

$$\underline{w}(i+1)=\underline{w}(i)+\eta(i)\sum_{n=1}^{N}\left[\!\left[\underline{w}^{T} z_{n}\underline{x}_{n} \leq 0\right]\!\right] z_{n}\underline{x}_{n} \ , \quad \eta(i)\geq 0 .$$

Stop when $\sum_{n=1}^{N}\left[\!\left[\underline{w}^{T} z_{n}\underline{x}_{n} \leq 0\right]\!\right] z_{n}\underline{x}_{n}=0$.

(

For sequential and stochastic GD

$$J(\underline{w})=\sum_{n=1}^{N}-\underline{w}^{T} z_{n}\underline{x}_{n}\left[\!\left[\underline{w}^{T} z_{n}\underline{x}_{n} \leq 0\right]\!\right]=\sum_{n=1}^{N} J_{n}(\underline{w})$$

$$\nabla_{\underline{w}} J_{n}(\underline{w})=-z_{n}\underline{x}_{n}\left[\!\left[\underline{w}^{T} z_{n}\underline{x}_{n} \leq 0\right]\!\right] \quad \forall \underline{w} \text{ such that } \underline{w}^{T} z_{n}\underline{x}_{n} \neq 0$$

(For $\underline{w}$ at $\underline{w}^{T} z_{n}\underline{x}_{n}=0$ we can use a left-sided derivative to get the same expression for $\nabla_{\underline{w}} J_{n}(\underline{w})$.)

Basic Sequential GD

Randomly shuffle the order of training data points

Initialize $\underline{w}(0)$

Define seq_epoch_perc *m* as:

For *n* in $\{1,2,\cdots,N\}$

$i=(m-1)N+n-1$

$$\begin{cases} \underline{w}(i+1) = \underline{w}(i) + \eta(i) z_n \underline{x}_n & \text{if } \underline{w}(i)^T z_n \underline{x}_n \le 0 \\ \underline{w}(i+1) = \underline{w}(i) & \text{if } \underline{w}(i)^T z_n \underline{x}_n > 0 \end{cases}, \quad \eta(i) \ge 0$$

Iterate seq_epoch_perc over $m = 1,2,3,\cdots$ until no change in $\underline{w}$ over 1 full epoch

Stochastic GD – variant 1 (randomly shuffle before each epoch)

Initialize $\underline{w}(0)$

Iterate over $m = 1,2,3,\cdots$ until no change in $\underline{w}$ over 1 full epoch:

(i) Randomly shuffle dataset

(ii) Perform seq_epoch_perc m

Stochastic GD – variant 2 (choose each data point randomly, with replacement)

Initialize $\underline{w}(0)$

Iterate over $i = 0,1,2,\cdots$ (data points) until halting condition is satisfied:

(i) Randomly pick a training data point (with replacement) ; call it $\underline{x}_n$

(ii) Perform single-sample update

$$\begin{cases} \underline{w}(i+1) = \underline{w}(i) + \eta(i) z_n \underline{x}_n & \text{if } \underline{w}(i)^T z_n \underline{x}_n \le 0 \\ \underline{w}(i+1) = \underline{w}(i) & \text{if } \underline{w}(i)^T z_n \underline{x}_n > 0 \end{cases}, \quad \eta(i) \ge 0$$

Mini-Batch GD (compromise between batch GD and stochastic GD)

Initialize $\underline{w}(0)$

Iterate until halting condition is satisfied:

(i) Pick M data points out of N points in the training set, at random

(ii) Perform batch GD on the M data points (1 iteration)

(iii) Replace the data points to the training set

[

Mini-Batch GD – variant 2 (splitting into non-overlapping mini-batches)

Initialize $\underline{w}(0)$

Iterate until convergence (below is one epoch):

(i) Shuffle the data order (including labels). This is optional, but is most commonly done.

(ii) Partition the training data into B mini-batches, each of size M data points. This can be done by taking the first M data points for mini-batch 1, the second M data points for mini-batch 2, etc. Note that your last mini-batch may have fewer than M data points if N is not an integer multiple of M.

(iii) For $b = 1, 2, \ldots B$: Perform batch GD on the M data points from mini-batch b (1 iteration)

Notes – Perceptron GD

1. *Halting condition.* All algorithms that give convergence criteria (Batch, Sequential, and Stochastic var-1) can have ad hoc halting conditions added to stop iterations when convergence hasn't been (or can't be) reached.

   Halting conditions for the other algorithms (Stochastic variant 2 and Mini-Batch) can be modeled after the generic halting conditions outlined in the general case.

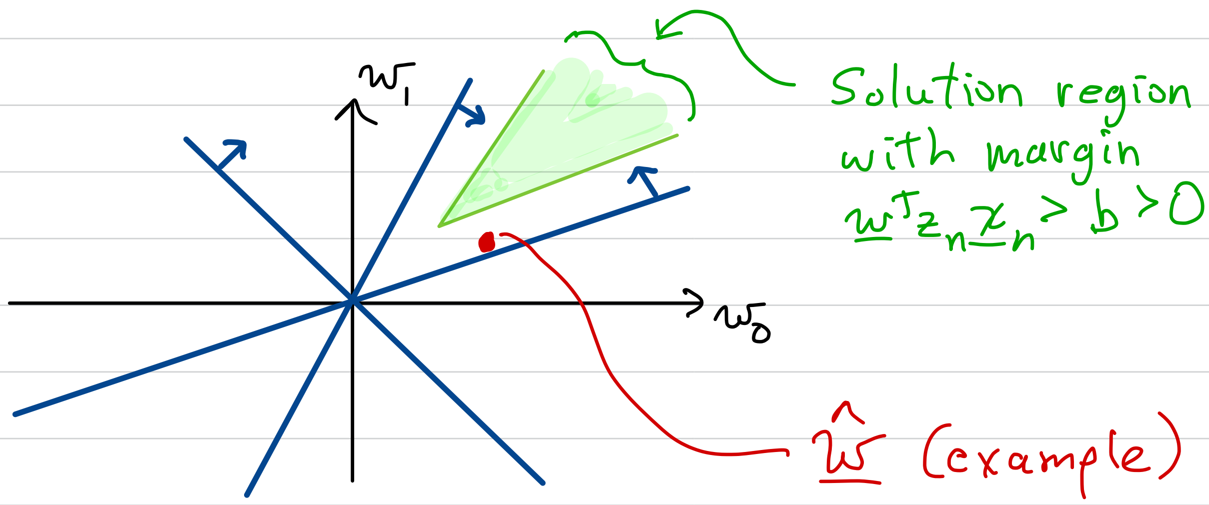
2. *Choice of $\eta(i)$ for convergence.* For linearly separable data, fixed increment perceptron converges for constant $\eta(i) = \eta(0) > 0 \quad \forall i$. Proof of this, and more general convergence criteria, are covered in lecture.

3. *Initial condition* $\underline{w}(0)$ is arbitrary (*e.g.*, can be $\underline{0}$, $a\underline{1}$, or random within some range)

Variants of perceptron learning algorithm

There are many variants.

Ex: Perceptron with margin



w_1
w_0
Solution region with margin
$\underline{w}^T z_n \underline{x}_n > b > 0 \quad \forall n.$
$\underline{\hat{w}}$ (example)

Algorithm: $\underline{w}(i+1) = \underline{w}(i) + \eta(i) z_n \underline{x}_n [\![\underline{w}(i)^T z_n \underline{x}_n \leq b]\!]$

Perceptron Learning Algorithm: Convergence

Assumptions: (i) $C=2$ classes
(ii) Training dataset is linearly separable

1. Fixed increment perceptron: $\eta(i)=\eta_0>0$
using sequential GD.
$\longrightarrow$ convergence (to a solution vector $\hat{\underline{w}}$) is guaranteed.
This has been proved [1].

2. Variable increment perceptron: $\eta(i)$ is some specified function of i.
using sequential GD or batch GD.
Set of sufficient conditions for convergence [1]:

(i) $\eta(i)\geq 0 \quad \forall i$

(ii) $\lim\limits_{m\to\infty} \sum_{i=1}^{m} \eta(i)=\infty$

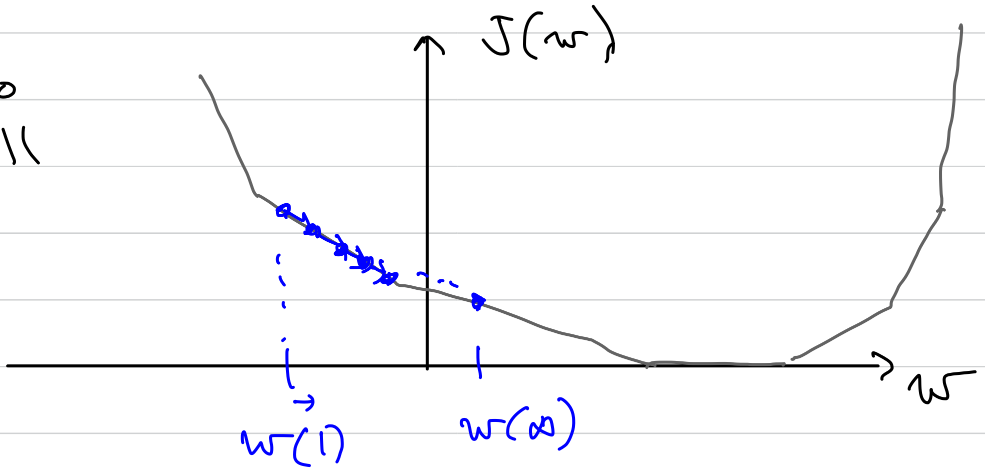
and (iii) $\lim\limits_{m\to\infty} \dfrac{\sum_{i=1}^{m}\eta^2(i)}{\left[\sum_{i=1}^{m}\eta(i)\right]^2}=0$

(i)–(iii) are satisfied if, e.g.:

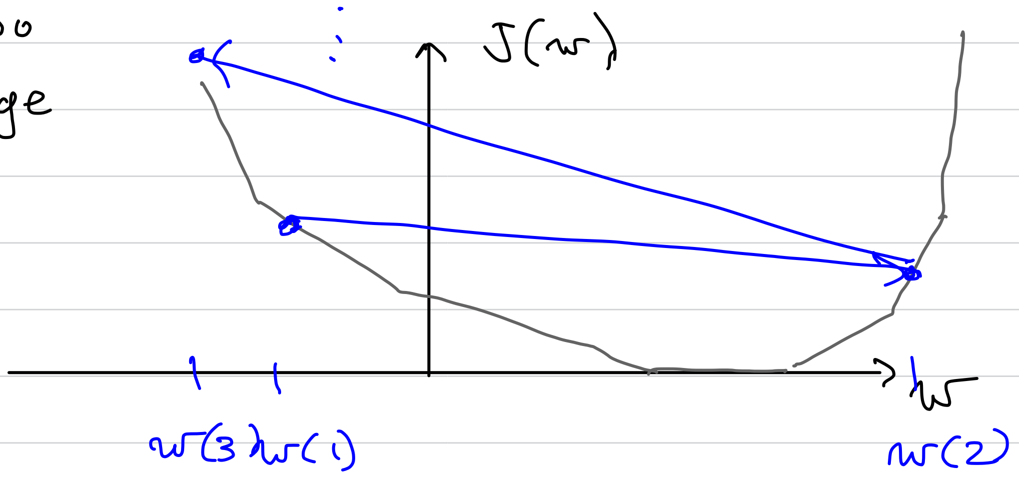
- $\eta(i)=\eta_0>0$.
- $\eta(i)\sim\frac{1}{i}$

Ref: 1. Duda, Hart, and Stork, Pattern Classification, Second Ed., Sec. 5.5.

GD: $\eta(i) =$ too small



$\eta(i) =$ too large



Multiclass Perceptron Algorithm Using Sequential Gradient Descent

- For $C > 2$, using maximal value method
- Decision rule: $g_k(\underline{x}) > g_j(\underline{x}) \ \forall j \neq k \ \Rightarrow \ \underline{x} \in S_k$
- Augmented space
- $g_k(\underline{x}) = \underline{w}_k^T \underline{x}, \quad k = 1, 2, \cdots, C$

Criterion function:

$$J(\underline{w}_1, \underline{w}_2, \cdots, \underline{w}_C) = -\sum_{n=1}^{N} \underbrace{[[l_n \neq k_n]]}_{\underline{x}_n \text{ is misclassified}} (\underline{w}_k^T \underline{x}_n - \underline{w}_l^T \underline{x}_n)$$

$$k_n = \text{given class label index of } \underline{x}_n; \ \ l_n = \text{argmax}_m\{g_m(\underline{x}_n)\}$$

← label assigned by classifier

Algorithm:

1. Shuffle the order of training data points
2. For each data point $\underline{x}^{(k)}$ [Given $\underline{x}^{(k)} \in S_k$]:

   If $g_k\left(\underline{x}^{(k)}\right) > g_j\left(\underline{x}^{(k)}\right) \ \forall j \neq k$ ($\underline{x}^{(k)}$ is correctly classified)

   then: $\underline{w}^{(m)}(i+1) = \underline{w}^{(m)}(i) \ \ \forall m$

   else: $\left\{ \underline{w}^{(k)}(i+1) = \underline{w}^{(k)}(i) + \eta(i)\underline{x}^{(k)} \right.$

p. 16

class label chosen for $\underline{x}^{(k)}$ by classifier

Let $l = \arg\max_{j \neq k} \left\{ g_j\left(\underline{x}^{(k)}\right) \right\}$ (if more than one possible l, pick any one)

$\underline{w}^{(l)}(i+1) = \underline{w}^{(l)}(i) - \eta(i)\underline{x}^{(k)}$; $\underline{w}^{(m)}(i+1) = \underline{w}^{(m)}(i) \quad \forall m \neq l,k$

(update $\underline{w}^{(l)}$ of incorrect class chosen by classifier)

until: all training data points are correctly classified.

Convergence: Convergence proven for $\eta(i) = \text{constant} > 0$, for linearly separable data[1].

Ref: 1. Duda, Hart, and Stork, *Pattern Classification, Second Ed.*, Sec. 5.12.2.

Perceptron Learning Algorithm

+ low computational complexity
+ guaranteed convergence (on linearly separable datasets)
+ there exists guidance on how to choose $\eta(i)$.

- difficulty in handling data that's not linearly sep'le.
- often don't know a priori whether the data is linearly separable.
- don't know in advance how long it will take to converge.