EECS 445

Introduction to Machine Learning

Perceptron Revisited
Loss Functions

Prof. Kutty

Announcements

- HW 1 released last week
- HW1 topics
  - Vectors & Planes Review
  - **numpy** Exercises
  - Perceptron Algorithm → see updated specs / announcement
  - Perceptron Convergence Proof
  - SGD with Logistic Loss

Homework 1 (50 pts)
Due: Tuesday, January 30th at 10:00pm

Submission: Please upload your completed assignment to Gradescope. Your submission can be either handwritten or typed. Assign all pages (including pages containing code) to their respective questions. Incorrectly assigned pages will not be graded.

Reminder: Honor Code

Honor Code and Collaboration:

Unless otherwise specified in an assignment, all submitted work must be your own, original work. If you are referencing others' work, put it in quotes. If you are directly quoting, or building on others' writing, provide a citation. See the Rackham Graduate policy on Academic and Professional Integrity for the definition of plagiarism, and associated consequences. Violations of the Honor Code will be taken seriously: Please see details: https://elc.engin.umich.edu/honor-council/ Students are **encouraged to collaborate** on conceptual understanding (except when taking exams). Please use Piazza to this effect and for other technical discussions. However, students are expected to **write their solutions on their own** and **should not look at any other student's write-up**.

Note on use of Generative AI (GenAI) tools

GenAI is changing rapidly, and new tools are becoming increasingly prevalent. Any and all use of tools that emulate human capabilities (including but not restricted to ChatGPT, Stable Diffusion, DALL-E, etc.) to perform assignments or other works in the course will be treated similarly to having another person perform your work and constitutes a clear honor code violation. See details above.

In effect, using a Generative AI tool *to understand the concepts* is akin to discussing from class material with another person; if it will aid your understanding and you are able to critically assess the input, you may use it (with attribution). **Please note that unattributed use of these tools, or using the tools to solve your assignment constitutes a clear honor code violation.**

Today's Agenda

- Recap:
  - Linear Classifiers
  - Classification Algorithm: Perceptron
- The Perceptron Update
  - Perceptron with Offset
- Non-linear separability and linear classifiers
- And later…
  - Speeding things up with Stochastic Gradient Descent

https://forms.gle/ffiBvNbPjHF8ghi77

training dataset

$S_n = \{(\bar{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$

$\bar{x}^{(i)} \in \mathbb{R}^d$

$y^{(i)} \in \{+1, -1\}$

Recap: The Model

decision boundary

$\bar{\theta} \cdot \bar{x} + b = 0$

parameter

$\bar{\theta} \in \mathbb{R}^d$

offset

$b \in \mathbb{R}$

$\bar{x}^{(test)}$

$\downarrow$

predict label

$\text{sign}(\bar{\theta} \cdot \bar{x}^{(test)} + b)$

Linear Classifier

Given: training data $S_n = \left\{\left(\bar{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{n}$

$\bar{x}^{(i)} \in \mathbb{R}^d$

$y^{(i)} \in \{-1, +1\}$

Goal: Learn a linear decision boundary (through the origin) that minimizes training error

$$h(\bar{x}; \bar{\theta}) = \text{sign}(\bar{\theta} \cdot \bar{x})$$

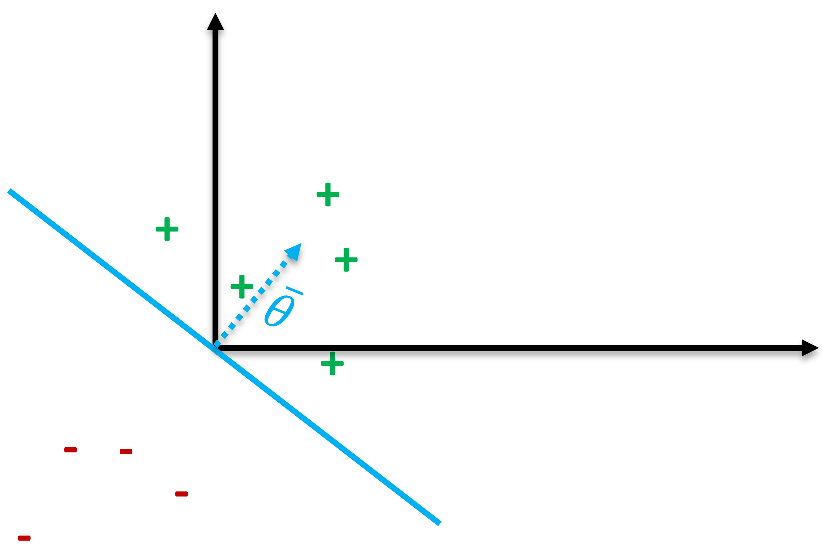
$\bar{\theta} \in \mathbb{R}^d$

misclassification rate

$$E_n(\bar{\theta}) = \frac{1}{n}\sum_{i=1}^{n}[\![y^{(i)} \neq h(\bar{x}^{(i)}; \bar{\theta})]\!]$$

$$= \frac{1}{n}\sum_{i=1}^{n}[\![y^{(i)}\left(\bar{\theta} \cdot \bar{x}^{(i)}\right) \leq 0]\!]$$

fraction of misclassified points



simplifying assumptions: linear separability

Recap: The Algorithm

Review: The Perceptron Algorithm

$k = 0, \overline{\boldsymbol{\theta}}^{(k)} = \overline{\mathbf{0}}$

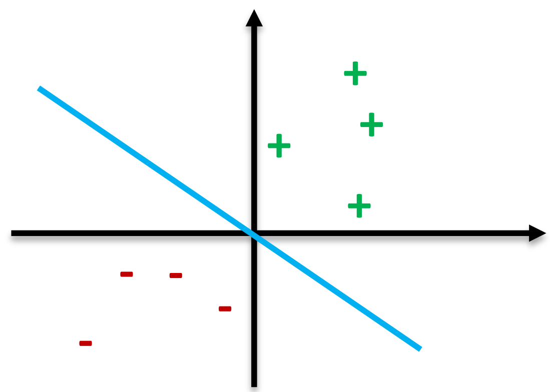
while there exists a misclassified point

  for $i = 1, \ldots, n$

    if $\boldsymbol{y}^{(i)}\left(\overline{\boldsymbol{\theta}}^{(k)} \cdot \overline{\boldsymbol{x}}^{(i)}\right) \leq \mathbf{0}$

      $\overline{\boldsymbol{\theta}}^{(k+1)} = \overline{\boldsymbol{\theta}}^{(k)} + \boldsymbol{y}^{(i)}\overline{\boldsymbol{x}}^{(i)}$

      k++

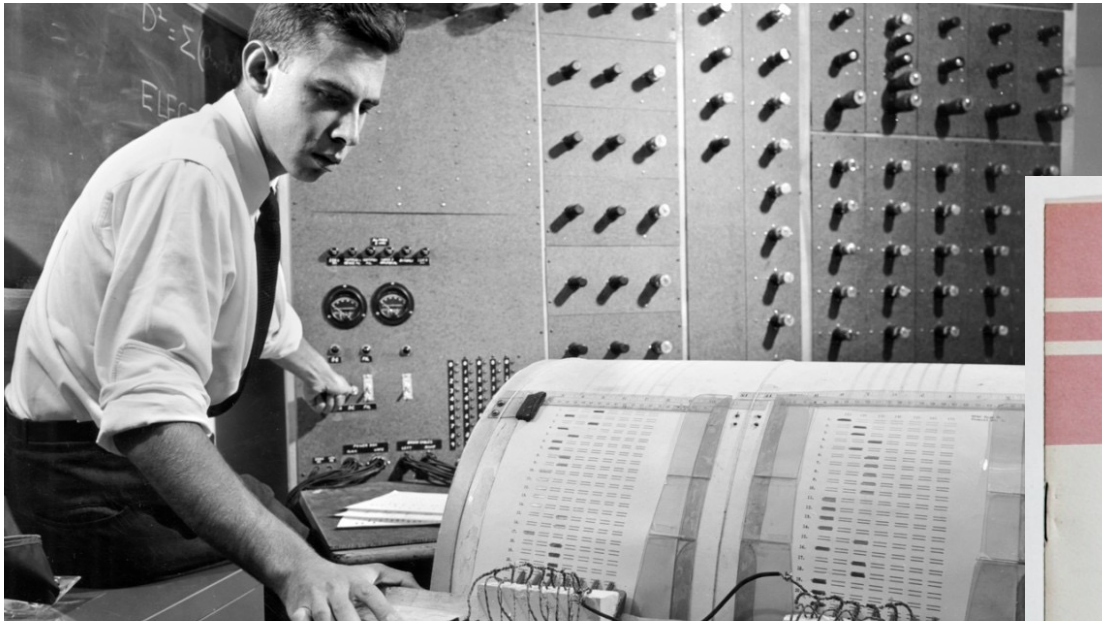


Theorem: The perceptron algorithm *converges* after a finite number of steps when the training examples are linearly separable

- Zero training error at convergence

What if examples are not linearly separable?

- algorithm does not converge
- moreover, does not find classifier with smallest training error



“the first machine which is capable of having an original idea”

- Frank Rosenblatt

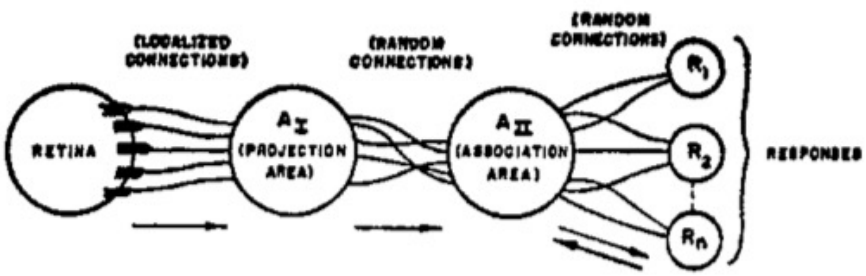


FIG. 1. Organization of a perceptron.

Vol. VI, No. 2, Summer 1958

research trends

CORNELL AERONAUTICAL LABORATORY, INC., BUFFALO 21, NEW YORK

The Design of an Intelligent AUTOMATON

by FRANK ROSENBLATT

Introducing the perceptron — A machine which senses, recognizes, remembers, and responds like the human mind.

STORIES about the creation of machines having human qualities have long been a fascinating province in the realm of science fiction. Yet we are now about to witness the birth of such a machine — a machine capable of perceiving, recognizing, and identifying its surroundings without any human training or control.

Development of that machine has stemmed from a search for an understanding of the physical mechanisms which underlie human experience and intelligence. The question of the nature of these processes is at least as ancient as any other question in western science and philosophy, and, indeed, ranks as one of the greatest scientific challenges of our time.

Our understanding of this problem has gone perhaps as far as had the development of physics before Newton. We have some excellent descriptions of the phenomena to be explained, a number of interesting hypotheses, and a little detailed knowledge about events in the nervous system. But we lack agreement on any integrated set of principles by which the functioning of the nervous system can be understood.

We believe now that this ancient problem is about to yield to our theoretical investigation for three reasons:

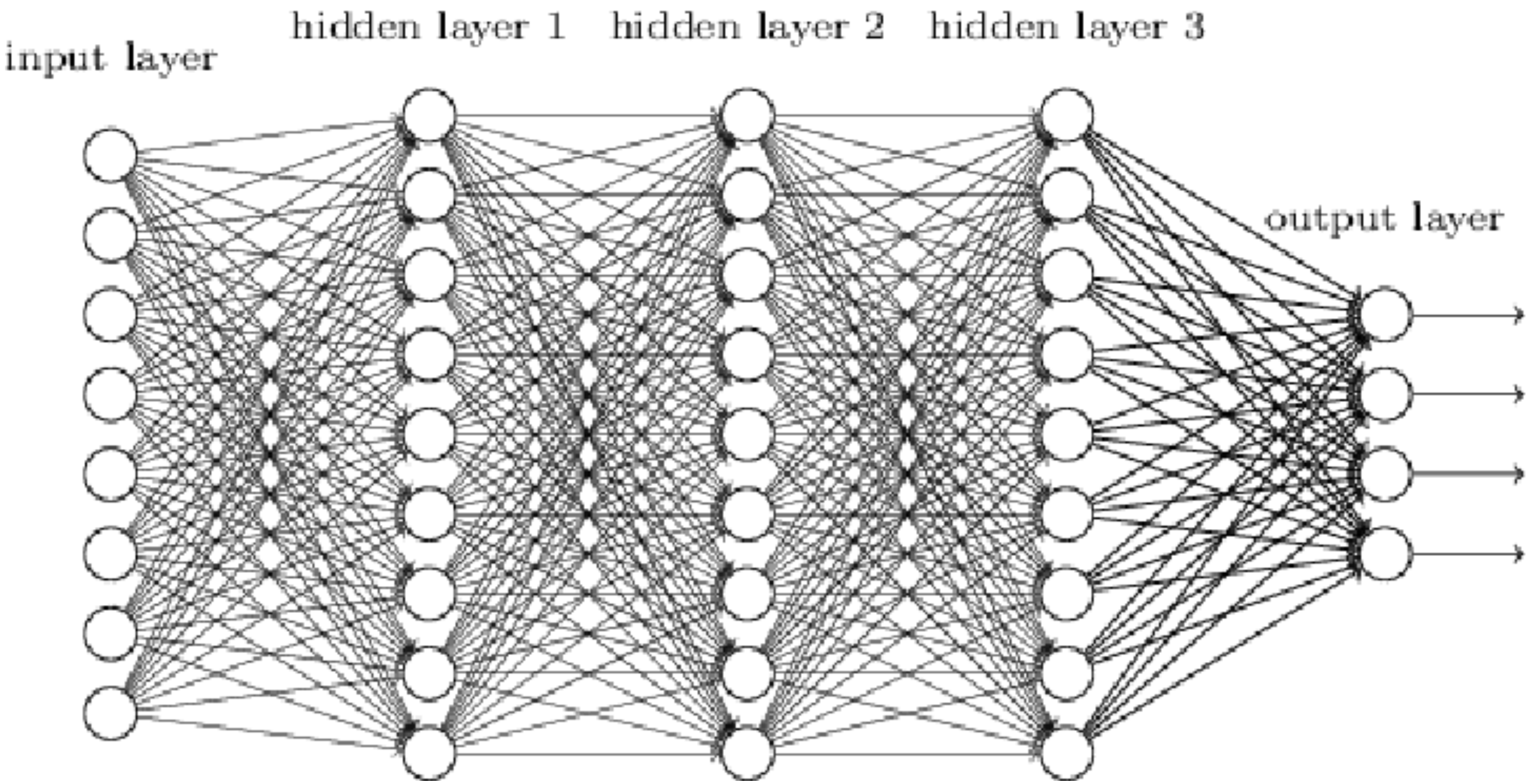
First, in recent years our knowledge of the functioning of individual cells in the central nervous system has vastly increased.

Second, large numbers of engineers and mathematicians are, for the first time, undertaking serious study of the mathematical basis for thinking, perception, and the handling of information by the central nervous system, thus providing the hope that these problems may be within our intellectual grasp.

Third, recent developments in probability theory and in the mathematics of random processes provide new tools for the study of events in the nervous system, where only the gross statistical organization is known and the precise cell-by-cell “wiring diagram” may never be obtained.

Receives Navy Support

In July, 1957, Project PARA (Perceiving and Recognizing Automaton), an internal research program which had been in progress for over a year at Cornell Aeronautical Laboratory, received the support of the Office of Naval Research. The program had been concerned primarily with the application of probability theory to


input layer
hidden layer 1
hidden layer 2
hidden layer 3
output layer

$\text{sign}(\bar{\theta}^{(k)} \cdot \bar{x})$

$\bar{\theta}^{(k)} \rightarrow$ current guess on the decision boundary

The Perceptron Update

$$\text{if } y^{(i)}\left(\bar{\boldsymbol{\theta}}^{(k)} \cdot \bar{\boldsymbol{x}}^{(i)}\right) \leq \mathbf{0}$$

$$\bar{\boldsymbol{\theta}}^{(k+1)} = \bar{\boldsymbol{\theta}}^{(k)} + \boldsymbol{y}^{(i)}\bar{\boldsymbol{x}}^{(i)}$$

is $\bar{x}^{(i)}$ misclassified by $\text{sign}(\bar{\theta}^{(k)} \cdot \bar{x})$

Why does this update make sense?

Suppose in the k[th] iteration the algorithm considers the point $\bar{x}^{(i)}$

If $\bar{x}^{(i)}$ is correctly classified

The parameter is not updated

k = 0, $\overline{\boldsymbol{\theta}}^{(\boldsymbol{k})}$ = $\overline{\mathbf{0}}$

while there exists a misclassified point

    for i = 1, …,n

        if $\boldsymbol{y}^{(\boldsymbol{i})}\left(\overline{\boldsymbol{\theta}}^{(\boldsymbol{k})} \cdot \overline{\boldsymbol{x}}^{(\boldsymbol{i})}\right) \leq \mathbf{0}$

            $\overline{\boldsymbol{\theta}}^{(\boldsymbol{k}+\mathbf{1})} = \overline{\boldsymbol{\theta}}^{(\boldsymbol{k})} + \boldsymbol{y}^{(\boldsymbol{i})}\overline{\boldsymbol{x}}^{(\boldsymbol{i})}$

            k++

Why does this update make sense?

Suppose in the kth iteration the algorithm considers the point $\bar{x}^{(i)}$

If $\bar{x}^{(i)}$ is misclassified

And the parameter is updated as $\overline{\boldsymbol{\theta}}^{(\boldsymbol{k}+\mathbf{1})} = \overline{\boldsymbol{\theta}}^{(\boldsymbol{k})} + \boldsymbol{y}^{(\boldsymbol{i})}\overline{\boldsymbol{x}}^{(\boldsymbol{i})}$

After the update $\boldsymbol{y}^{(\boldsymbol{i})}\left(\overline{\boldsymbol{\theta}}^{(\boldsymbol{k}+\mathbf{1})} \cdot \overline{\boldsymbol{x}}^{(\boldsymbol{i})}\right) = \boldsymbol{y}^{(\boldsymbol{i})}\left(\left(\overline{\boldsymbol{\theta}}^{(\boldsymbol{k})} + \boldsymbol{y}^{(\boldsymbol{i})}\overline{\boldsymbol{x}}^{(\boldsymbol{i})}\right) \cdot \overline{\boldsymbol{x}}^{(\boldsymbol{i})}\right)$ $= y^{(i)}\left(\bar{\theta}^{(k)} \cdot \bar{x}^{(i)} + y^{(i)}\, \bar{x}^{(i)} \cdot \bar{x}^{(i)}\right)$

$$= y^{(i)}\left(\bar{\theta}^{(k)} \cdot \bar{x}^{(i)}\right) + \underbrace{\cancel{y^{(i)}}^{2}}_{=1} \left\|\bar{x}^{(i)}\right\|_2^{\,2}$$

$$= y^{(i)}\left(\bar{\theta}^{(k)} \cdot \bar{x}^{(i)}\right) + \left\|\bar{x}^{(i)}\right\|_2^{\,2}$$

k = 0, $\overline{\boldsymbol{\theta}}^{(\boldsymbol{k})} = \overline{\mathbf{0}}$

while there exists a misclassified point

    for i = 1, …,n

        if $\boldsymbol{y}^{(\boldsymbol{i})}\left(\overline{\boldsymbol{\theta}}^{(\boldsymbol{k})} \cdot \overline{\boldsymbol{x}}^{(\boldsymbol{i})}\right) \le \mathbf{0}$

            $\overline{\boldsymbol{\theta}}^{(\boldsymbol{k}+\mathbf{1})} = \overline{\boldsymbol{\theta}}^{(\boldsymbol{k})} + \boldsymbol{y}^{(\boldsymbol{i})}\overline{\boldsymbol{x}}^{(\boldsymbol{i})}$

            k++

Observations

- The perceptron algorithm updates based on a single (misclassified) point at a time
- The algorithm moves the hyperplane in the 'right' direction based on that point

Since for a misclassified point $\bar{x}^{(i)}$,

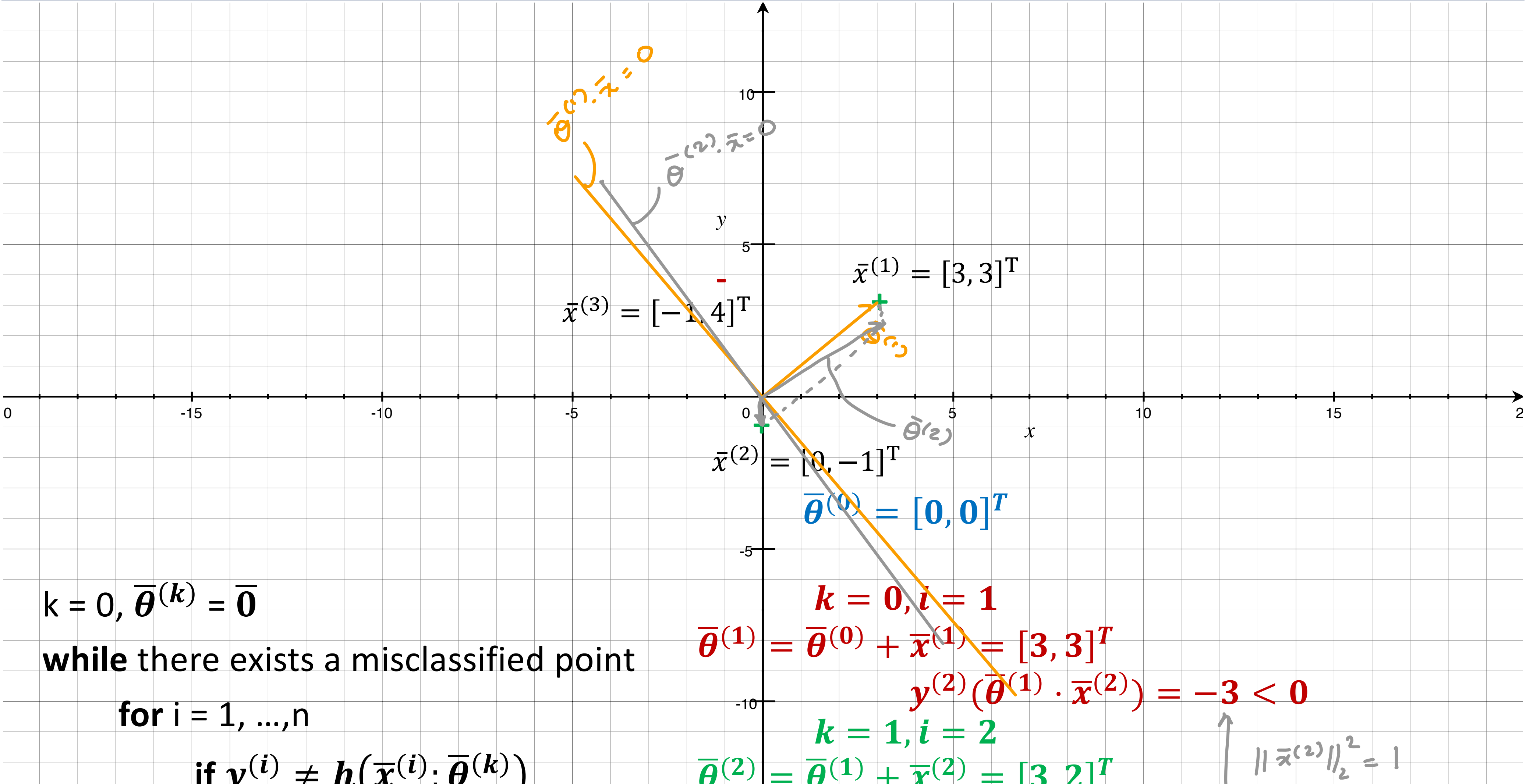
$$\bar{\theta}^{(k+1)} = \bar{\theta}^{(k)} + y^{(i)}\bar{x}^{(i)}$$

$$y^{(i)}\left(\bar{\theta}^{(k+1)} \cdot \bar{x}^{(i)}\right) = y^{(i)}\left(\bar{\theta}^{(k)} \cdot \bar{x}^{(i)}\right) + \left\|\bar{x}^{(i)}\right\|_2^{2}$$

So if $\left\|\bar{x}^{(i)}\right\|_2^2 > 0$

$$y^{(i)}\left(\bar{\theta}^{(k+1)} \cdot \bar{x}^{(i)}\right) > y^{(i)}\left(\bar{\theta}^{(k)} \cdot \bar{x}^{(i)}\right)$$

Misclassifying despite an update



k = 0, $\overline{\boldsymbol{\theta}}^{(k)} = \overline{\mathbf{0}}$

while there exists a misclassified point

for i = 1, …,n

if $y^{(i)} \neq h\left(\overline{\boldsymbol{x}}^{(i)}; \overline{\boldsymbol{\theta}}^{(k)}\right)$

$\overline{\boldsymbol{\theta}}^{(k+1)} = \overline{\boldsymbol{\theta}}^{(k)} + y^{(i)}\overline{\boldsymbol{x}}^{(i)}$

k++

$\boldsymbol{k = 0, i = 1}$

$\overline{\boldsymbol{\theta}}^{(1)} = \overline{\boldsymbol{\theta}}^{(0)} + \overline{\boldsymbol{x}}^{(1)} = [3,3]^{T}$

$y^{(2)}(\overline{\boldsymbol{\theta}}^{(1)} \cdot \overline{\boldsymbol{x}}^{(2)}) = -3 < 0$

$\boldsymbol{k = 1, i = 2}$

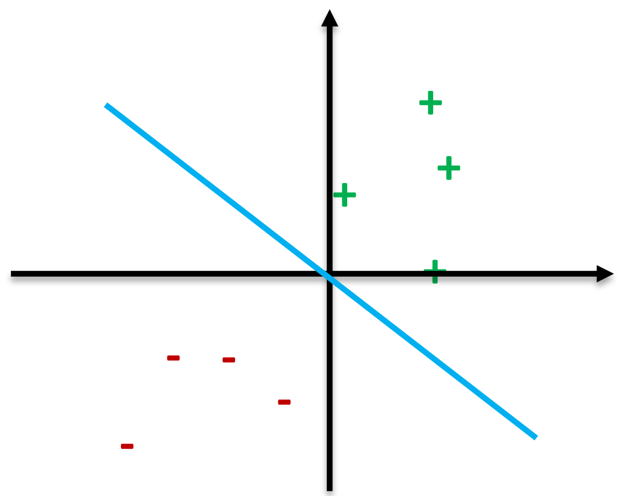
$\overline{\boldsymbol{\theta}}^{(2)} = \overline{\boldsymbol{\theta}}^{(1)} + \overline{\boldsymbol{x}}^{(2)} = [3,2]^{T}$

$y^{(2)}(\overline{\boldsymbol{\theta}}^{(2)} \cdot \overline{\boldsymbol{x}}^{(2)}) = -2 < 0$

Linear Classifier: (initial) simplifying assumptions

Goal: Learn a linear decision boundary

i.e., constrain possible choices $\mathcal{H}$ to hyperplanes



simplifying assumptions:

- constrain $\mathcal{H}$ to be the set of all hyperplanes that go through the origin
  - e.g., in $\mathbb{R}^2$ this is the set of lines that go through the origin

- constrain problem to datasets that are linearly separable

Perceptron with Offset

Linearly separable with offset

Given training examples

$$S_n = \{(\bar{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$$

we say the data are linearly separable

if there exists b ∈ **R** and $\bar{\theta} \in \mathbb{R}^d$

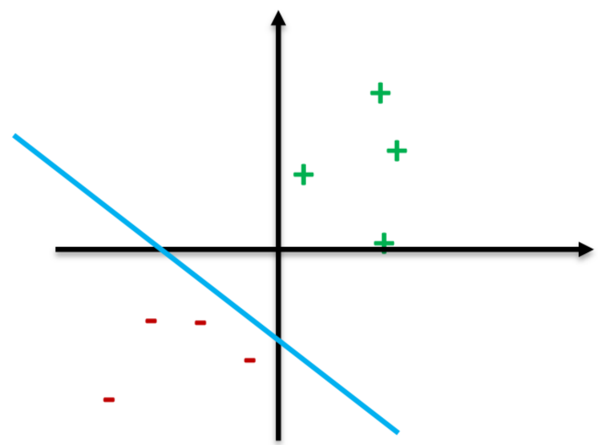
such that

$$y^{(i)}(\bar{\theta} \cdot \bar{x}^{(i)} + b) > 0$$

for i = 1, …,n. We refer to

$$\{\bar{x} : \bar{\theta} \cdot \bar{x} + b = 0\}$$

as a separating hyperplane



Goal: Learn a linear decision boundary

that minimizes training error

$$h(\bar{x}; \bar{\theta}) = \text{sign}(\bar{\theta} \cdot \bar{x} + b)$$

, b

$$E_n(\bar{\theta}) = \frac{1}{n}\sum_{i=1}^{n} [\![y^{(i)} \left(\bar{\theta} \cdot \bar{x}^{(i)} + b\right) \leq 0]\!]$$

Perceptron Algorithm (Modified)

e.g., $\bar{x}^{(2)} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$

$\Rightarrow \bar{x}^{(2)'} = \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix}$

$\Rightarrow \bar{\theta}^{(10)'} = \begin{bmatrix} 5 \\ 4 \\ 7 \end{bmatrix}$

$\bar{\theta}^{(10)'} \cdot \bar{x}^{(2)'} = \begin{bmatrix} 5 \\ 4 \\ 7 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix}$

$5 \cdot 1 + \underbrace{\begin{bmatrix} 4 \\ 7 \end{bmatrix}}_{\bar{\theta}^{(10)}} \cdot \underbrace{\begin{bmatrix} 3 \\ 2 \end{bmatrix}}_{\bar{x}^{(2)}}$

$\bar{\boldsymbol{x}}^{(\boldsymbol{i})} \in \mathbb{R}^d$

Let $\bar{\boldsymbol{x}}^{(\boldsymbol{i})'} = \left[\mathbf{1},\ \bar{\boldsymbol{x}}^{(\boldsymbol{i})}\right]^{\boldsymbol{T}}$ $\in$

$S_n = \{(\bar{x}^{(i)}, y^{(i)})\}_{i=1}^n$

$\downarrow$

$S'_n = \{(\bar{x}^{(i)\prime}, y^{(i)})\}_{i=1}^n$

$k = 0$, $\bar{\boldsymbol{\theta}}^{(k)'} = \bar{0}$

while there exists a misclassified point

for i = 1, …, n

if $\boldsymbol{y}^{(\boldsymbol{i})}\left(\bar{\boldsymbol{\theta}}^{(\boldsymbol{k})'} \cdot \bar{\boldsymbol{x}}^{(\boldsymbol{i})\prime}\right) \leq \mathbf{0}$

$\bar{\boldsymbol{\theta}}^{(\boldsymbol{k}+\mathbf{1})'} = \bar{\boldsymbol{\theta}}^{(\boldsymbol{k})'} + \boldsymbol{y}^{(\boldsymbol{i})}\bar{\boldsymbol{x}}^{(\boldsymbol{i})'}$

k++

$\begin{bmatrix} \theta_0^{(k+1)} \\ \bar{\theta}^{(k+1)} \end{bmatrix} = \begin{bmatrix} \theta_0^{(k)} \\ \bar{\theta}^{(k)} \end{bmatrix} + y^{(i)} \begin{bmatrix} 1 \\ \bar{x}^{(i)} \end{bmatrix}$

$\Rightarrow \theta_0^{(k+1)} = \theta_0^{(k)} + y^{(i)}$

$\bar{\theta}^{(k+1)} = \bar{\theta}^{(k)} + y^{(i)} \bar{x}^{(i)}$

At convergence, $\forall i \;\; y^{(i)}\left(\bar{\theta}^{(k)\prime} \cdot \bar{x}^{(i)\prime}\right) > 0$

i.e., $\forall i \quad y^{(i)} \left(\begin{bmatrix} \theta_0^{(k)} \\ \theta_1^{(k)} \\ \cdots \\ \theta_d^{(k)} \end{bmatrix} \cdot \begin{bmatrix} 1 \\ x_1^{(i)} \\ \cdots \\ x_d^{(i)} \end{bmatrix} \right) > 0$

i.e., $\forall i \quad y^{(i)} \left(\bar{\theta}^{(k)} \cdot \bar{x}^{(i)} + \theta_0^{(k)}\right) > 0$

i.e., $\bar{\theta}^{(k)} \cdot \bar{x}^{(i)} + \theta_0^{(k)} = 0$ is a separating hyperplane for the training data $S_n = \{(\bar{x}^{(i)}, y^{(i)})\}_{i=1}^n$

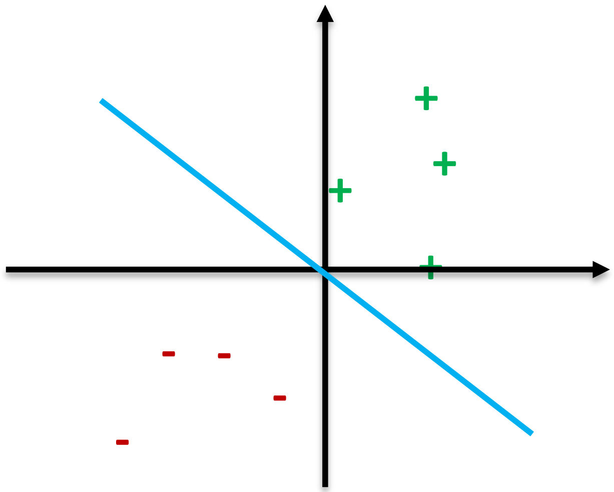
~~Linear separability assumption~~

Loss functions and Gradient Descent

Linear Classifier

Goal: Learn a linear decision boundary

i.e., constrain possible choices $\mathcal{H}$ to hyperplanes



simplifying assumptions:

- constrain $\mathcal{H}$ to be the set of all hyperplanes that go through the origin
  - e.g., in $\mathbb{R}^2$ this is the set of lines that go through the origin

- constrain problem to datasets that *are* linearly separable

When data are *not* linearly separable

Idea: focus on minimizing empirical risk

Goal: Find parameter vector $\bar{\theta}$ that minimizes the empirical risk.

$$R_n(\bar{\theta}) = \frac{1}{n}\sum_{i=1}^{n} [\![y^{(i)} \neq h(\bar{x}^{(i)}; \bar{\theta})]\!]$$

$$= \frac{1}{n}\sum_{i=1}^{n} [\![y^{(i)} (\bar{\theta} \cdot \bar{x}^{(i)}) \leq 0]\!] \qquad \text{with linear classifiers}$$

$$= \frac{1}{n}\sum_{i=1}^{n} \mathrm{Loss}\left(y^{(i)} (\bar{\theta} \cdot \bar{x}^{(i)}) \right)$$

Bad news:

direct minimization of this function is difficult in general

(for 0-1 loss)

(NP hard)

When the data are not linearly separable we need a slightly different approach: the goal is still to generalize well to new examples.

Empirical risk

Goal: Find parameter vector $\bar{\theta}$ that minimizes the empirical risk.

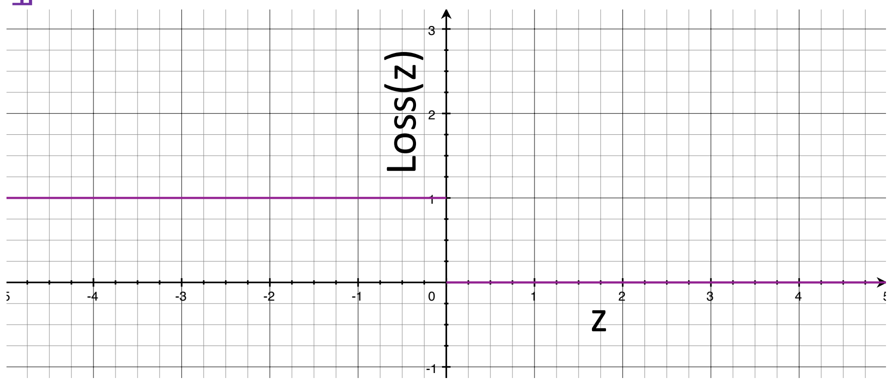
$$R_n(\bar{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \text{Loss}\left(h\left(\bar{x}^{(i)}; \bar{\theta}\right), y^{(i)}\right) = \frac{1}{n}\sum_{i=1}^{n} \text{Loss}\left(y^{(i)}\left(\bar{\theta} \cdot \bar{x}^{(i)}\right)\right)$$

for linear classifiers

Examples of loss functions for linear classifiers :

- 0-1 Loss function

$$R_n(\bar{\theta}) = \frac{1}{n}\sum_{i=1}^{n} [\![y^{(i)}\left(\bar{\theta} \cdot \bar{x}^{(i)}\right) \leq 0]\!]$$



- Hinge loss function

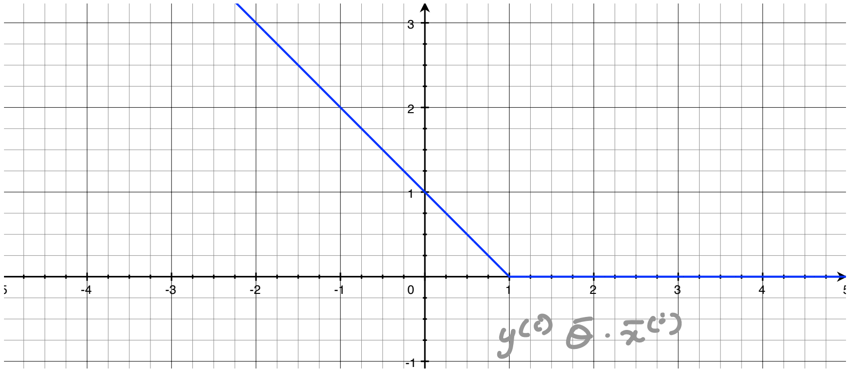
Empirical risk with hinge loss

Good news:

Empirical risk with hinge loss is a convex function

Idea: minimize empirical risk using gradient descent

$$R_n(\bar{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \max\left(\left(1 - y^{(i)}(\bar{\theta} \cdot \bar{x}^{(i)})\right), 0\right)$$



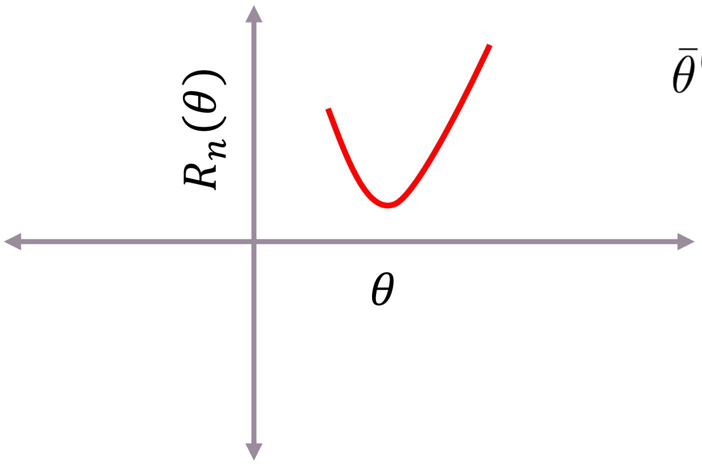
Advantages of the hinge loss function:

incorporates idea of 'worse' mistakes

forces predictions to be more than just a 'little correct'

Gradient Descent (GD)

Gradient Descent (GD) Idea: take a small step in the opposite direction to which the gradient points



$$\bar{\theta}^{(k+1)} = \bar{\theta}^{(k)} - \eta \nabla_{\bar{\theta}} R_n(\bar{\theta})|_{\bar{\theta}=\bar{\theta}^k}$$

informally, a convex function is characteristically 'bowl'-shaped

In general

$$\nabla_{\bar{\theta}} R_n(\bar{\theta}) = \left[\frac{\partial R_n(\bar{\theta})}{\partial \theta_1}, \ldots, \frac{\partial R_n(\bar{\theta})}{\partial \theta_d}\right]^{\mathrm{T}}$$

the gradient points in the direction of the greatest rate of increase

Gradient Descent (GD)

Goal:

Given $S_n = \{(\bar{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$

Find the value of parameter $\bar{\boldsymbol{\theta}}$ that minimizes empirical risk $R_n(\bar{\boldsymbol{\theta}})$

$$R_n(\bar{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \max\{1 - y^{(i)}(\bar{\theta} \cdot \bar{x}^{(i)}), 0\}$$

$k = 0, \bar{\theta}^{(0)} = \bar{0}$

while convergence criteria is not met

$$\bar{\theta}^{(k+1)} = \bar{\theta}^{(k)} - \eta \nabla_{\bar{\theta}} R_n(\bar{\theta})|_{\bar{\theta}=\bar{\theta}^k}$$

k++

Gradient Descent (GD)

1. Keep track of $R_n(\bar{\theta})$
2. Keep track of $\bar{\theta}$
3. Keep track of k

$k \;=\; 0, \bar{\theta}^{(0)} = \bar{0}$

while convergence criteria is not met

$$\bar{\theta}^{(k+1)} = \bar{\theta}^{(k)} - \eta \nabla_{\bar{\theta}} R_n(\bar{\theta})|_{\bar{\theta}=\bar{\theta}^k}$$

k++

Step size for Gradient Descent (GD)

hyperparameter

How do we set η?

- **constant** η

  Issues:

  too large: can cause algorithm to overshoot and 'oscillate'

  too small: can be too slow

- **variable** η

$$\eta_k = 1/(k+1)$$

Gradient Descent (GD)

$$\bar{\theta}^{(k+1)} = \bar{\theta}^{(k)} - \eta \nabla_{\bar{\theta}} R_n(\bar{\theta})|_{\bar{\theta}=\bar{\theta}^k}$$

$$R_n(\bar{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \max\{1 - y^{(i)}(\bar{\theta} \cdot \bar{x}^{(i)}), 0\}$$

Bad news:

Due to the summation involved in calculating the gradient, in order to make a single update, you have to look at *every training example*

If we have a lot of training examples, this will be *slow*