The perceptron algorithm

Christos Dimitrakakis

October 3, 2025

Outline

Guessing gender from height

- Feature space $\mathcal{X} \subset \mathbb{R}$: e.g. height
- Label space $\mathcal{Y} = \{-1, 1\}$: e.g. gender
- Can we find some $\beta_1 \in \mathbb{R}$ and a direction $\beta_0 \in \{-1, +1\}$ so as to separate the genders?

Online learning: At time t

- We choose a separator β_0^t, β_1^t
- We observe a new datapoint x_t, y_t
- We make a mistake at time t if:

$$\beta^t x_t - \beta_0^t \leq 0.$$

- If we stop making mistakes, then we are classifying everything perfectly.

Can you find a threshold that makes a small number of mistakes?

`./src/Perceptron/perceptron_simple.py`

Non-separable classes

./fig/histogram_heights.png

A more complex example

- Feature space $\mathcal{X} \subset \mathbb{R}^n$: e.g. height and weight for $n = 2$
- Label space $\mathcal{Y} = \{-1, 1\}$: e.g. gender
- Can we find some line so as to separate the genders?

-./src/Perceptron/show_class_data_labels.py

- Is there an algorithm for doing so?

A linear classifier

The separating hyperplane

We now have parameters $\beta_0 \in \mathbb{R}$ and $\beta \in \mathbb{R}^n$ defining a hyperplane $f(x) = 0$ in $\mathbb{R}^n$

$$f(x) = \beta_0 + \beta^\top x = \beta_0 + \sum_{i=1}^{n} \beta_i x_i.$$

A linear classifier

The separating hyperplane

We now have parameters $\beta_0 \in \mathbb{R}$ and $\beta \in \mathbb{R}^n$ defining a hyperplane $f(x) = 0$ in $\mathbb{R}^n$

$$f(x) = \beta_0 + \beta^\top x = \beta_0 + \sum_{i=1}^{n} \beta_i x_i.$$

If we augment x with an additional component $x_0 = 1$, we can write

$$f(x) = \beta^\top x = \sum_{i=0}^{n} \beta_i x_i.$$

A linear classifier

The separating hyperplane

We now have parameters $\beta_0 \in \mathbb{R}$ and $\beta \in \mathbb{R}^n$ defining a hyperplane $f(x) = 0$ in $\mathbb{R}^n$

$$f(x) = \beta_0 + \beta^\top x = \beta_0 + \sum_{i=1}^{n} \beta_i x_i.$$

If we augment x with an additional component $x_0 = 1$, we can write

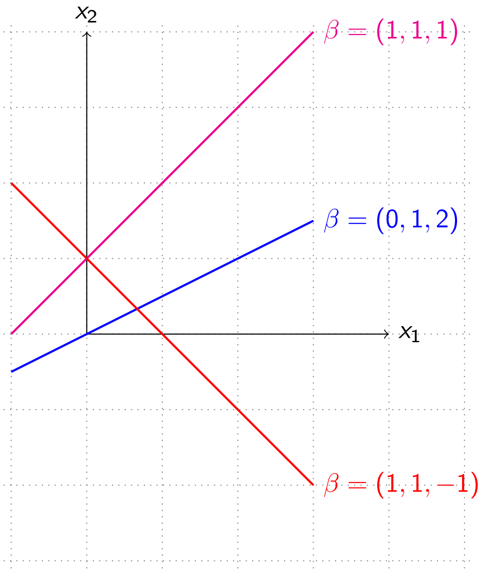
$$f(x) = \beta^\top x = \sum_{i=0}^{n} \beta_i x_i.$$

The classifier

The perceptron decision rule is $\pi(x) = \mathrm{sign}(f(x))$

- If $f(x) > 0$, we assign class +1
- If $f(x) < 0$, we assign class -1

Hyperplanes in 2 dimensions (lines)



These lines are the solution to $f(x) = 0$

The Perceptron

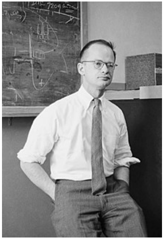


Figure: Pitts

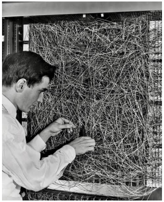


Figure: Rosenblatt

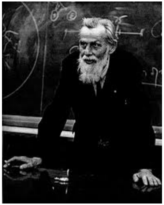


Figure: McCulloch

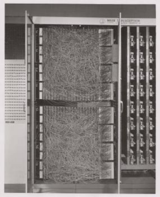


Figure: Perceptron Mark I

The perceptron algorithm

Input

- Feature space $X \subset \mathbb{R}^n$.
- Label space $Y = \{-1, 1\}$.
- Data (x_t, y_t), $t \in [T]$, with $x_t \in X, y_t \in Y$.

Algorithm

- $\beta^0 \sim \mathrm{Normal}^n(0, I)$. % Initialise parameters
- For $t = 1, \ldots, T$
  - $a_t = \mathrm{sgn}(\beta^t \cdot x_t)$. % Classify example
  - If $a_t \neq y_t$
    - $\beta^t = \beta^{t-1} + y_t x_t$ % Move hyperplane
  - Else
    - $\beta^t = \beta^{t-1}$ % Do nothing for correct examples
  - EndIf
- Return β^T

Perceptron examples

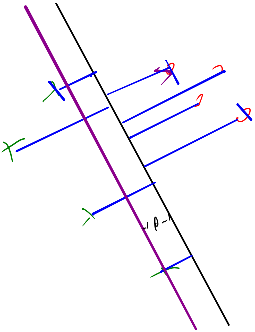
Example 1: One-dimensional data

- Done on the board
- Shows how the algorithm works.
- Demonstrates the idea of a margin

Example 2: Two-dimensional data

- See in-class programming exercise

Margins and the perceptron theorem



- The hyperplane β^* separates the examples
- The margin ρ is the minimum distance ρ between β^* and any point.

Theorem (Perceptron theorem)

The number of mistakes is bounded by ρ^{-2}, where $\|x_t\| \leq 1$, $\rho \leq y_t(x_t^\top \beta^*)$ for some margin ρ and hyperplane β^* with $\|\beta^*\| = 1$.

Simple proof

- Scale data: $\|x\| \leq 1$

Simple proof

- Scale data: $\|x\| \leq 1$
- Separating plane: $y_t(x_t \cdot \beta^*) \geq \rho \forall t$, $\|\beta^*\| = 1$.

Simple proof

- Scale data: $\|x\| \leq 1$
- Separating plane: $y_t(x_t \cdot \beta^*) \geq \rho \forall t$, $\|\beta^*\| = 1$.
- When we make an update: $y_t(x_t \cdot \beta^t) \leq 0$.

Simple proof

- Scale data: $\|x\| \leq 1$
- Separating plane: $y_t(x_t \cdot \beta^*) \geq \rho \forall t$, $\|\beta^*\| = 1$.
- When we make an update: $y_t(x_t \cdot \beta^t) \leq 0$.
- At each mistake, $\beta^t \cdot \beta^*$ grows by at least ρ.

Simple proof

- Scale data: $\|x\| \leq 1$
- Separating plane: $y_t(x_t \cdot \beta^*) \geq \rho \forall t$, $\|\beta^*\| = 1$.
- When we make an update: $y_t(x_t \cdot \beta^t) \leq 0$.
- At each mistake, $\beta^t \cdot \beta^*$ grows by at least ρ.

Simple proof

- Scale data: $\|x\| \leq 1$
- Separating plane: $y_t(x_t \cdot \beta^*) \geq \rho \forall t$, $\|\beta^*\| = 1$.
- When we make an update: $y_t(x_t \cdot \beta^t) \leq 0$.
- At each mistake, $\beta^t \cdot \beta^*$ grows by at least ρ.

  $$\beta^{t+1} \cdot \beta^* = (\beta^t + y x_t) \cdot \beta^* = \beta^t \cdot \beta^* + y(x_t \cdot \beta^*) \geq \beta^t \cdot \beta^* + \rho$$

Simple proof

- Scale data: $\|x\| \leq 1$
- Separating plane: $y_t(x_t \cdot \beta^*) \geq \rho \forall t$, $\|\beta^*\| = 1$.
- When we make an update: $y_t(x_t \cdot \beta^t) \leq 0$.
- At each mistake, $\beta^t \cdot \beta^*$ grows by at least ρ.

  $$\beta^{t+1} \cdot \beta^* = (\beta^t + yx_t) \cdot \beta^* = \beta^t \cdot \beta^* + y(x_t \cdot \beta^*) \geq \beta^t \cdot \beta^* + \rho$$

- At each mistake, $\beta \cdot \beta$ grows by at most 1.

$$\beta^{t+1} \cdot \beta^{t+1} = (\beta^t + yx_t) \cdot (\beta^t + yx_t) = \beta^t \cdot \beta^t + 2y(\beta^t \cdot x_t) + y^2(x_t \cdot x_t) \leq \beta^t \cdot \beta^t + 1$$

Simple proof

- Scale data: $\|x\| \leq 1$
- Separating plane: $y_t(x_t \cdot \beta^*) \geq \rho \forall t$, $\|\beta^*\| = 1$.
- When we make an update: $y_t(x_t \cdot \beta^t) \leq 0$.
- At each mistake, $\beta^t \cdot \beta^*$ grows by at least ρ.

  $$\beta^{t+1} \cdot \beta^* = (\beta^t + yx_t) \cdot \beta^* = \beta^t \cdot \beta^* + y(x_t \cdot \beta^*) \geq \beta^t \cdot \beta^* + \rho$$

- At each mistake, $\beta \cdot \beta$ grows by at most 1.

$$\beta^{t+1} \cdot \beta^{t+1} = (\beta^t + yx_t) \cdot (\beta^t + yx_t) = \beta^t \cdot \beta^t + 2y(\beta^t \cdot x_t) + y^2(x_t \cdot x_t) \leq \beta^t \cdot \beta^t + 1$$

Putting it together

After M mistakes:

- $\beta^t \cdot \beta^* \geq M\rho$

Simple proof

- Scale data: $\|x\| \leq 1$
- Separating plane: $y_t(x_t \cdot \beta^*) \geq \rho \forall t$, $\|\beta^*\| = 1$.
- When we make an update: $y_t(x_t \cdot \beta^t) \leq 0$.
- At each mistake, $\beta^t \cdot \beta^*$ grows by at least ρ.

  $$\beta^{t+1} \cdot \beta^* = (\beta^t + yx_t) \cdot \beta^* = \beta^t \cdot \beta^* + y(x_t \cdot \beta^*) \geq \beta^t \cdot \beta^* + \rho$$

- At each mistake, $\beta \cdot \beta$ grows by at most 1.

$$\beta^{t+1} \cdot \beta^{t+1} = (\beta^t + yx_t) \cdot (\beta^t + yx_t) = \beta^t \cdot \beta^t + 2y(\beta^t \cdot x_t) + y^2(x_t \cdot x_t) \leq \beta^t \cdot \beta^t + 1$$

Putting it together

After M mistakes:

- $\beta^t \cdot \beta^* \geq M\rho$
- $\beta^t \cdot \beta^t \leq M$

Simple proof

- Scale data: $\|x\| \leq 1$
- Separating plane: $y_t(x_t \cdot \beta^*) \geq \rho \forall t$, $\|\beta^*\| = 1$.
- When we make an update: $y_t(x_t \cdot \beta^t) \leq 0$.
- At each mistake, $\beta^t \cdot \beta^*$ grows by at least ρ.

  $$\beta^{t+1} \cdot \beta^* = (\beta^t + yx_t) \cdot \beta^* = \beta^t \cdot \beta^* + y(x_t \cdot \beta^*) \geq \beta^t \cdot \beta^* + \rho$$

- At each mistake, $\beta \cdot \beta$ grows by at most 1.

$$\beta^{t+1} \cdot \beta^{t+1} = (\beta^t + yx_t) \cdot (\beta^t + yx_t) = \beta^t \cdot \beta^t + 2y(\beta^t \cdot x_t) + y^2(x_t \cdot x_t) \leq \beta^t \cdot \beta^t + 1$$

Putting it together

After M mistakes:

- $\beta^t \cdot \beta^* \geq M\rho$
- $\beta^t \cdot \beta^t \leq M$
- So $M\rho \leq \beta^t \cdot \beta^* \leq \|\beta^t\| = \sqrt{\beta^t \cdot \beta^t} \leq \sqrt{M}$.

Simple proof

- Scale data: $\|x\| \leq 1$
- Separating plane: $y_t(x_t \cdot \beta^*) \geq \rho \forall t$, $\|\beta^*\| = 1$.
- When we make an update: $y_t(x_t \cdot \beta^t) \leq 0$.
- At each mistake, $\beta^t \cdot \beta^*$ grows by at least ρ.

  $$\beta^{t+1} \cdot \beta^* = (\beta^t + yx_t) \cdot \beta^* = \beta^t \cdot \beta^* + y(x_t \cdot \beta^*) \geq \beta^t \cdot \beta^* + \rho$$

- At each mistake, $\beta \cdot \beta$ grows by at most 1.

$$\beta^{t+1} \cdot \beta^{t+1} = (\beta^t + yx_t) \cdot (\beta^t + yx_t) = \beta^t \cdot \beta^t + 2y(\beta^t \cdot x_t) + y^2(x_t \cdot x_t) \leq \beta^t \cdot \beta^t + 1$$

Putting it together

After M mistakes:

- $\beta^t \cdot \beta^* \geq M\rho$
- $\beta^t \cdot \beta^t \leq M$
- So $M\rho \leq \beta^t \cdot \beta^* \leq \|\beta^t\| = \sqrt{\beta^t \cdot \beta^t} \leq \sqrt{M}$.
- Thus, $M \leq \rho^{-2}$.

Promise of the perceptron

1958 New York Times...

NEW NAVY DEVICE LEARNS BY DOING

Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

WASHINGTON, July 7 (UPI) —The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The embryo—the Weather Bureau's $2,000,000 "704" computer—learned to differentiate between right and left after fifty attempts in the Navy's demonstration for newsmen.

The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of $100,000.

Dr. Frank Rosenblatt, designer of the Perceptron, conducted the demonstration. He said the machine would be the first device to think as the human brain. As do human beings, Perceptron will make mistakes at first, but will grow wiser as it gains experience, he said.

Dr. Rosenblatt, a research psychologist at the Cornell Aeronautical Laboratory, Buffalo, said Perceptrons might be fired to the planets as mechanical space explorers.

Without Human Controls

The Navy said the perceptron would be the first non-living mechanism "capable of receiving, recognizing and identifying its surroundings without any human training or control."

The "brain" is designed to remember images and information it has perceived itself. Ordinary computers remember only what is fed into them on punch cards or magnetic tape.

Later Perceptrons will be able to recognize people and call out their names and instantly translate speech in one language to speech or writing in another language, it was predicted.

Mr. Rosenblatt said in principle it would be possible to build brains that could reproduce themselves on an assembly line and which would be conscious of their existence.

In today's demonstration, the "704" was fed two cards, one with squares marked on the left side and the other with squares on the right side.

Learns by Doing

In the first fifty trials, the machine made no distinction between them. It then started registering a "Q" for the left squares and "O" for the right squares.

Dr. Rosenblatt said he could explain why the machine learned only in highly technical terms. But he said the computer had undergone a "self-induced change in the wiring diagram."

The first Perceptron will have about 1,000 electronic "association cells" receiving electrical impulses from an eye-like scanning device with 400 photo-cells. The human brain has 10,000,000,000 responsive cells, including 100,000,000 connections with the eyes.

Promise versus reality

Focus on classification

- Rosenblatt only consider classification problems
- Many problems in learning and AI are not simply classification problems
- Classification requires labels. These are not always easily available.

Separable representation assumption

- Rosenblatt assumed that there was a representation available that would allow us to distinguish classes.
- However, it is not clear *a priori* how to obtain such a data representation from the data. Progress followed roughly these steps:
  - Hand-crafted features
  - Random features
  - Multi-layer perceptrons, hand-crafted architectures, and backpropagation
  - Attention mechanisms

The gradient descent method: one dimension

- Function to minimise $f : \mathbb{R} \to \mathbb{R}$.
- Derivative $\frac{d}{d\beta} f(\beta)$

Gradient descent algorithm

- Input: initial value β^0, learning rate schedule α_t
- For $t = 1, \ldots, T$
  - $\beta^{t+1} = \beta^t - \alpha_t \frac{d}{d\beta} f(\beta^t)$
- Return β^T

Properties

- If $\sum_t \alpha_t = \infty$ and $\sum_t \alpha_t^2 < \infty$, it finds a local minimum β^T, i.e. there is $\epsilon > 0$ so that
$$f(\beta^T) < f(\beta), \forall \beta : \|\beta^T - \beta\| < \epsilon.$$

Gradient methods for expected value

Estimate the expected value

$x_t \sim P$ with $\mathbb{E}_P[x_t] = \mu$.

Gradient methods for expected value

Estimate the expected value

$x_t \sim P$ with $\mathbb{E}_P[x_t] = \mu$.

Objective: mean squared error

Here $\ell(x, \beta) = (x - \beta)^2$.

$$\min_{\beta} \mathbb{E}_P[(x_t - \beta)^2].$$

Gradient methods for expected value

Estimate the expected value

$x_t \sim P$ with $\mathbb{E}_P[x_t] = \mu$.

Objective: mean squared error

Here $\ell(x, \beta) = (x - \beta)^2$.

$$\min_\beta \mathbb{E}_P[(x_t - \beta)^2].$$

Exact gradient update

If we know P, then we can calculate

$$\beta^{t+1} = \beta^t - \alpha_t \frac{d}{d\beta} \mathbb{E}_P[(x - \beta^t)^2] \tag{1}$$

$$\frac{d}{d\beta} \mathbb{E}_P[(x - \beta^t)^2] = 2\,\mathbb{E}_P[x] - \beta^t \tag{2}$$

Gradient for mean estimation

- Let us show this in detail

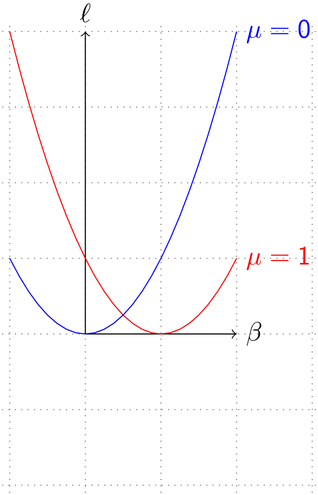
$$\begin{aligned}\frac{d}{d\beta}\mathbb{E}_P[(x-\beta)^2] &= \int_{-\infty}^{\infty} dP(x)\frac{d}{d\beta}(x-\beta)^2 \\ &= \int_{-\infty}^{\infty} dP(x)2(x-\beta) \\ &= 2\,\mathbb{E}_P[x]-2\beta.\end{aligned}$$

- If we set the derivative to zero, then we find the optimal solution:

$$\beta^* = \mathbb{E}_P[x]$$

- How can we do this if we only have data $x_t \sim P$?

Mean-squared error cost function



Here we see a plot of $\ell(\mu, \beta) = (\beta - \mu)^2$.

Stochastic gradient for mean estimation

Theorem (Sampling)

For any bounded random variable f,

$$\mathbb{E}_P[f] = \int_X dP(x) f(x) = \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} f(x_t) = \mathbb{E}_P\left[\frac{1}{T}\sum_{t=1}^{T} f(x_t)\right], \qquad x_t \sim P$$

Example (Sampling)

- If we sample x we approximate the gradient:

$$\frac{d}{d\beta}\mathbb{E}_P[(x-\beta)^2] = \int_{-\infty}^{\infty} dP(x)\frac{d}{d\beta}(x-\beta)^2 \approx \frac{1}{T}\sum_{t=1}^{T}\frac{d}{d\beta}(x_t-\beta)^2 = \frac{1}{T}\sum_{t=1}^{T} 2(x_t -$$

Stochastic gradient for mean estimation

Theorem (Sampling)

For any bounded random variable f,

$$\mathbb{E}_P[f] = \int_X dP(x) f(x) = \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} f(x_t) = \mathbb{E}_P\left[\frac{1}{T}\sum_{t=1}^{T} f(x_t)\right], \qquad x_t \sim P$$

Example (Sampling)

- If we sample x we approximate the gradient:

$$\frac{d}{d\beta}\mathbb{E}_P[(x-\beta)^2] = \int_{-\infty}^{\infty} dP(x) \frac{d}{d\beta}(x-\beta)^2 \approx \frac{1}{T}\sum_{t=1}^{T} \frac{d}{d\beta}(x_t-\beta)^2 = \frac{1}{T}\sum_{t=1}^{T} 2(x_t -$$

- If we update β after each new sample x_t, we obtain:

$$\beta^{t+1} = \beta^t + 2\alpha_t(x_t - \beta^t)$$

The gradient method

- Function to minimise $f : \mathbb{R}^n \to \mathbb{R}$.
- Derivative $\nabla_\beta f(\beta) = \left(\frac{\partial f(\beta)}{\partial \beta_1}, \ldots, \frac{\partial f(\beta)}{\partial \beta_n}\right)$, where $\frac{\partial f}{\partial \beta_n}$ denotes the partial derivative, i.e. varying one argument and keeping the others fixed.

Gradient descent algorithm

- Input: initial value β^0, learning rate schedule α_t
- For $t = 1, \ldots, T$
  - $\beta^{t+1} = \beta^t - \alpha_t \nabla_\beta f(\beta^t)$
- Return β^T

Properties

- If $\sum_t \alpha_t = \infty$ and $\sum_t \alpha_t^2 < \infty$, it finds a local minimum β^T, i.e. there is $\epsilon > 0$ so that

$$f(\beta^T) < f(\beta), \forall \beta : \|\beta^T - \beta\| < \epsilon.$$

Stochastic gradient method

This is the same as the gradient method, but with added noise:

- $\beta^{t+1} = \beta^t - \alpha_t[\nabla_\beta f(\beta^t) + \omega_t]$
- $\mathbb{E}[\omega_t] = 0$ is sufficient for convergence.

Stochastic gradient method

This is the same as the gradient method, but with added noise:

- $\beta^{t+1} = \beta^t - \alpha_t[\nabla_\beta f(\beta^t) + \omega_t]$
- $\mathbb{E}[\omega_t] = 0$ is sufficient for convergence.

Example (When the cost is an expectation)

In machine learning, the cost is frequently an expectation of some function ℓ,

$$f(\beta) = \int_X dP(x)\ell(x, \beta)$$

This can be approximated with a sample

$$f(\beta) \approx \frac{1}{T} \sum_t \ell(x_t, \beta)$$

The same holds for the gradient:

$$\nabla_\beta f(\beta) = \int_X dP(x)\nabla_\beta \ell(x, \beta) \approx \frac{1}{T} \sum_t \nabla_\beta \ell(x_t, \beta)$$

Perceptron algorithm as gradient descent

Target error function

$$\mathbb{E}_{P}^{\beta}[\ell] = \int_{\mathcal{X}} dP(x) \sum_{y} P(y|x)\ell(x, y, \beta)$$

Minimises the error on the true distribution.

Perceptron algorithm as gradient descent

Target error function

$$\mathbb{E}_{P}^{\beta}[\ell] = \int_{\mathcal{X}} dP(x) \sum_{y} P(y|x)\ell(x, y, \beta)$$

Minimises the error on the true distribution.

Empirical error function

$$\mathbb{E}_{D}^{\beta}[\ell] = \frac{1}{T} \sum_{t=1}^{T} \ell(x_t, y_t, \beta), \qquad D = (x_t, y_t)_{t=1}^{T}, \quad x_t, y_t \sim P.$$

Minimises the error on the empirical distribution.

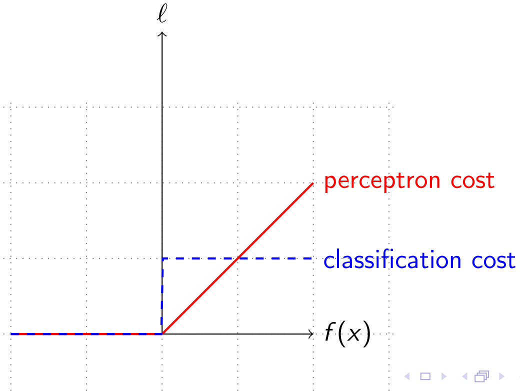
Cost functions and the chain rule

Perceptron cost function

The cost of each example

$$\ell(x, y, \beta) = \overbrace{\mathbb{I}\left\{y(x^\top \beta) < 0\right\}}^{\text{misclassified?}} \overbrace{\left[-y(x^\top \beta)\right]}^{\text{margin of error}} \tag{3}$$

where the indicator function $\mathbb{I}\{A\}$ is 1 when A is true and 0 otherwise.



Derivative of the perceptron cost function

The total cost over the data is defined as

$$L(D, \beta) = \sum_{(x,y)\in D} \ell(x, y, \beta)$$

Taking the derivative, we have

$$\nabla_\beta L(D, \beta) = \nabla_\beta \sum_{(x,y)\in D} \ell(x, y, \beta) = \sum_{(x,y)\in D} \nabla_\beta \ell(x, y, \beta)$$

Reminder: The chain rule

Let $z = g(y)$, $y = f(x)$ so that $z = g(f(x))$. Then $\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$

Derivative of the perceptron cost function

The total cost over the data is defined as

$$L(D, \beta) = \sum_{(x,y)\in D} \ell(x, y, \beta)$$

Taking the derivative, we have

$$\nabla_\beta L(D, \beta) = \nabla_\beta \sum_{(x,y)\in D} \ell(x, y, \beta) = \sum_{(x,y)\in D} \nabla_\beta \ell(x, y, \beta)$$

Reminder: The chain rule

Let $z = g(y)$, $y = f(x)$ so that $z = g(f(x))$. Then $\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$

Applying the chain rule to calculate the gradient

- $\nabla_\beta \ell(x, y, \beta) = -\mathbb{I}\left\{y(x^\top \beta) < 0\right\} \nabla_\beta [y(x^\top \beta)]$.

Derivative of the perceptron cost function

The total cost over the data is defined as

$$L(D, \beta) = \sum_{(x,y) \in D} \ell(x, y, \beta)$$

Taking the derivative, we have

$$\nabla_\beta L(D, \beta) = \nabla_\beta \sum_{(x,y) \in D} \ell(x, y, \beta) = \sum_{(x,y) \in D} \nabla_\beta \ell(x, y, \beta)$$

Reminder: The chain rule

Let $z = g(y)$, $y = f(x)$ so that $z = g(f(x))$. Then $\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$

Applying the chain rule to calculate the gradient

- $\nabla_\beta \ell(x, y, \beta) = -\mathbb{I}\left\{y(x^\top \beta) < 0\right\} \nabla_\beta [y(x^\top \beta)]$.
- $\frac{\partial \beta}{\partial \beta_i}[y(x_t^\top \beta)] = y x_{t,i}$ (gradient of Perceptron's output)

Derivative of the perceptron cost function

The total cost over the data is defined as

$$L(D,\beta) = \sum_{(x,y)\in D} \ell(x,y,\beta)$$

Taking the derivative, we have

$$\nabla_\beta L(D,\beta) = \nabla_\beta \sum_{(x,y)\in D} \ell(x,y,\beta) = \sum_{(x,y)\in D} \nabla_\beta \ell(x,y,\beta)$$

Reminder: The chain rule

Let $z = g(y)$, $y = f(x)$ so that $z = g(f(x))$. Then $\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$

Applying the chain rule to calculate the gradient

- $\nabla_\beta \ell(x,y,\beta) = -\mathbb{I}\left\{y(x^\top\beta) < 0\right\} \nabla_\beta[y(x^\top\beta)]$.
- $\frac{\partial\beta}{\partial\beta_i}[y(x_t^\top\beta)] = yx_{t,i}$ (gradient of Perceptron's output)
- Gradient update: $\beta^{t+1} = \beta^t - \nabla_\beta \ell(x,y,\beta) = \beta^t + yx_t$

Derivative of the perceptron cost function

The total cost over the data is defined as

$$L(D, \beta) = \sum_{(x,y) \in D} \ell(x, y, \beta)$$

Taking the derivative, we have

$$\nabla_\beta L(D, \beta) = \nabla_\beta \sum_{(x,y) \in D} \ell(x, y, \beta) = \sum_{(x,y) \in D} \nabla_\beta \ell(x, y, \beta)$$

Reminder: The chain rule

Let $z = g(y)$, $y = f(x)$ so that $z = g(f(x))$. Then $\frac{dz}{dx} = \frac{dz}{dy} \frac{dy}{dx}$

Applying the chain rule to calculate the gradient

- $\nabla_\beta \ell(x, y, \beta) = -\mathbb{I}\left\{y(x^\top \beta) < 0\right\} \nabla_\beta [y(x^\top \beta)]$.
- $\frac{\partial \beta}{\partial \beta_i}[y(x_t^\top \beta)] = y x_{t,i}$ (gradient of Perceptron's output)
- Gradient update: $\beta^{t+1} = \beta^t - \nabla_\beta \ell(x, y, \beta) = \beta^t + y x_t$

Derivative of the perceptron cost function

The total cost over the data is defined as

$$L(D, \beta) = \sum_{(x,y) \in D} \ell(x, y, \beta)$$

Taking the derivative, we have

$$\nabla_\beta L(D, \beta) = \nabla_\beta \sum_{(x,y) \in D} \ell(x, y, \beta) = \sum_{(x,y) \in D} \nabla_\beta \ell(x, y, \beta)$$

Reminder: The chain rule

Let $z = g(y)$, $y = f(x)$ so that $z = g(f(x))$. Then $\frac{dz}{dx} = \frac{dz}{dy} \frac{dy}{dx}$

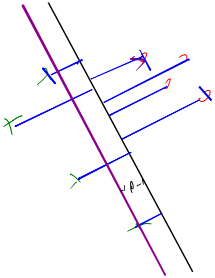
Applying the chain rule to calculate the gradient

- $\nabla_\beta \ell(x, y, \beta) = -\mathbb{I}\left\{y(x^\top \beta) < 0\right\} \nabla_\beta [y(x^\top \beta)]$.
- $\frac{\partial \beta}{\partial \beta_i}[y(x_t^\top \beta)] = y x_{t,i}$ (gradient of Perceptron's output)
- Gradient update: $\beta^{t+1} = \beta^t - \nabla_\beta \ell(x, y, \beta) = \beta^t + y x_t$

The classification error cost function is not differentiable :(

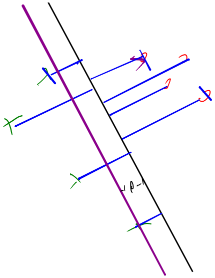
Margins and confidences

We can think of the output of the network as a measure of confidence



Margins and confidences

We can think of the output of the network as a measure of confidence



By applying the logit function, we can bound a real number x to $[0, 1]$:

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

Logistic regression

Output as a measure of confidence, given the parameter β

$$P_\beta(y = 1|x) = \frac{1}{1 + \exp(-x_t^\top \beta)}$$

The original output $x_t^\top \beta$ is now passed through the logit function.

Logistic regression

Output as a measure of confidence, given the parameter β

$$P_\beta(y=1|x) = \frac{1}{1+\exp(-x_t^\top \beta)}$$

The original output $x_t^\top \beta$ is now passed through the logit function.

Negative Log likelihood

$\ell(x_t, y_t, \beta) = -\ln P_\beta(y_t|x_t) = \ln(1+\exp(-y_t x_t^\top \beta))$

$$\begin{aligned}
\nabla_\beta \ell(x_t, y_t, \beta) &= \frac{1}{1+\exp(-yx_t^\top \beta)} \nabla_\beta [1+\exp(-yx_t^\top \beta)] \\
&= \frac{1}{1+\exp(-yx_t^\top \beta)} \exp(-yx_t^\top \beta)[\nabla_\beta(-y_t x_t^\top \beta)] \\
&= -\frac{1}{1+\exp(x_t^\top \beta)} (x_{t,i})_{i=1}^n e
\end{aligned}$$

- $\mathbb{E}_P(\ell) = \int_X dP(x) \sum_{y \in Y} P(y|x) P_\beta(y_t + x_t)$

The Perceptron and Gradients

./src/Perceptron/Perceptron_gd.ipynb

- Perceptron implemenation to fill in
- Gradient descent implementation
- Experiment on the learning rate with sklearn