Deep Learning

Lecture 01 — CS 577 Deep Learning

Instructor: Yutong Wang

Computer Science
Illinois Institute of Technology

August 21, 2024

Discussion with your neighbors (5 minutes)

- What do you want to learn in this class?
- Be as specific or as broad as you like
- Be ready to share.

Build LLM

Rec Sys

Popular foundation Dino

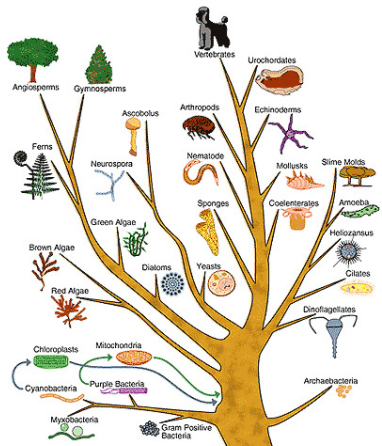
GANs and auto-encoder

Math of DL

RNNs

Scalability across Compute server

Tree of Life



Courtesy of BBC Wildlife Magazine

Humans vs algorithms

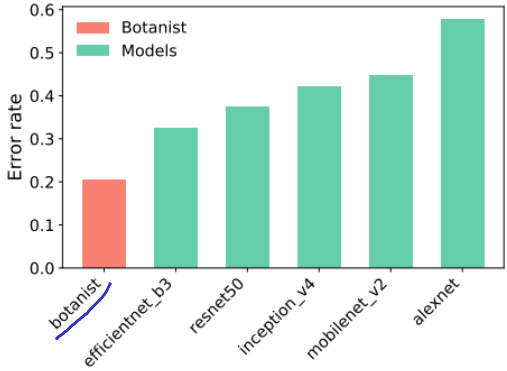


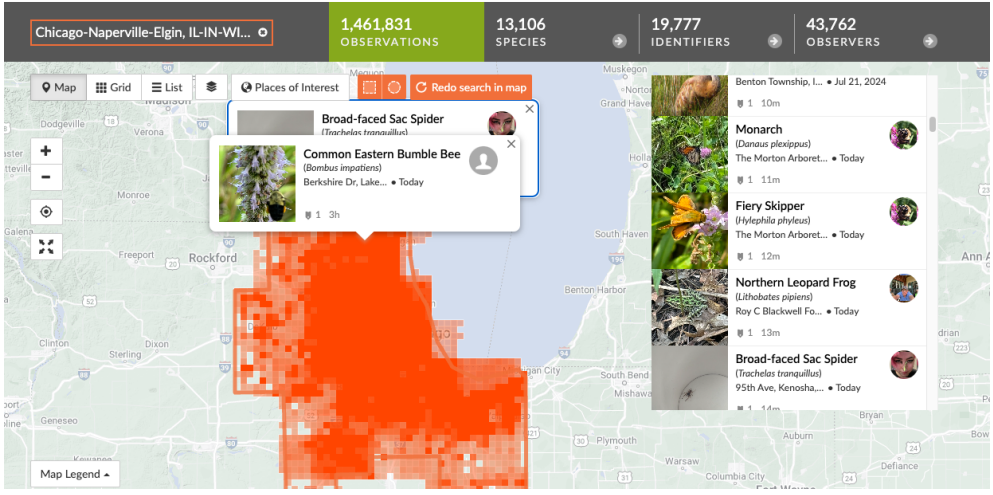
Figure 7: Error rate on the mini test set of an expert compared to several neural networks. All models and the expert return on average 4.1 species on the mini test set.

Garcin et al 2021 NeurIPS dataset

iNaturalist

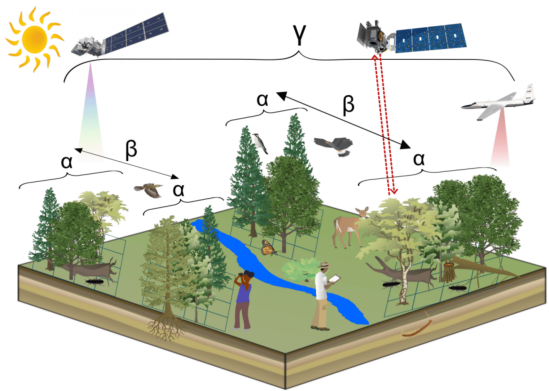
Courtesy of `inaturalist.org/`

iNaturalist



Courtesy of `inaturalist.org/`

Citizen Science for biodiversity monitoring



(Kacic and Kuenzer, 2022)

Deep learning is a continuously evolving field

- Frontier (state-of-the-art models) constantly moving

Deep learning is a continuously evolving field

- Frontier (state-of-the-art models) constantly moving
  - Support vector machines [BGV92]

Deep learning is a continuously evolving field

- Frontier (state-of-the-art models) constantly moving
  - Support vector machines [BGV92]
  - Convolutional nets [KSH12]

Deep learning is a continuously evolving field

- Frontier (state-of-the-art models) constantly moving
  - Support vector machines [BGV92]
  - Convolutional nets [KSH12]
  - Diffusion models [Soh+15; SE19]

Deep learning is a continuously evolving field

- Frontier (state-of-the-art models) constantly moving
  - Support vector machines [BGV92]
  - Convolutional nets [KSH12]
  - Diffusion models [Soh+15; SE19]
  - Transformers [Vas+17]

Deep learning is a continuously evolving field

- Frontier (state-of-the-art models) constantly moving
  - Support vector machines [BGV92]
  - Convolutional nets [KSH12]
  - Diffusion models [Soh+15; SE19]
  - Transformers [Vas+17]
- Models change, mathematical foundations don't

Why take this class?

- You want to understand deep learning "under the hood"
- You want to make your own contributions
- You want to be able to read cutting edge papers in DL

Backprop by hand

PyTorch by hand

"The Annotated Transformer"

OpenReview

Notations

Goodfellow et al Textbook

Let $i = 1, \ldots, N$ (the sample index)

- Samples $\mathbf{x}_i \in \mathcal{X} = \mathbb{R}^d$

CV d = width × height × channels

NLP d = context hello, how are you

Comp bio. d = # of genes 20,000 sparse vector

Notations

Let $i = 1, \ldots, N$ (the sample index)

- Samples $\mathbf{x}_i \in \mathcal{X}$
- Labels $y_i \in \mathcal{Y}$

Regression Classif LLM

$\mathbb{R}$ $\{\pm 1\}$ $\{1, \cdots, K\}$ next token pred

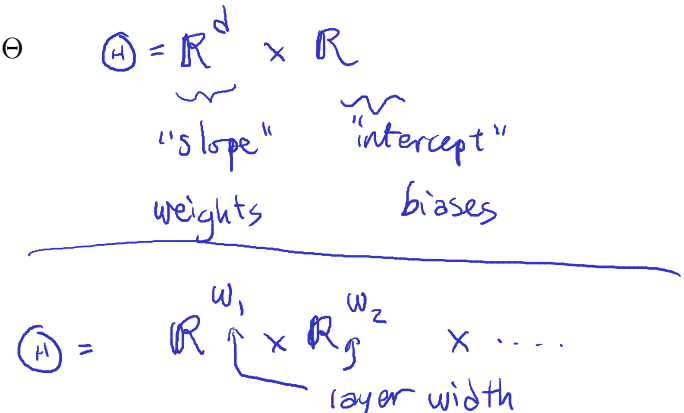
Structured label No label

Graphs Generation

Notations

Let $i = 1, \ldots, N$ (the sample index)

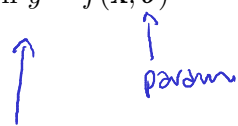
- Samples $\mathbf{x}_i \in \mathcal{X}$
- Labels $y_i \in \mathcal{Y}$
- Model params $\boldsymbol{\theta} \in \Theta$



Notations

Let $i = 1, \ldots, N$ (the sample index)

- Samples $\mathbf{x}_i \in \mathcal{X}$
- Labels $y_i \in \mathcal{Y}$
- Model params $\boldsymbol{\theta} \in \Theta$
- Prediction $\hat{y} = f(\mathbf{x}; \boldsymbol{\theta})$



`\hat{y}` is an estimate of y

Mathematical foundations

- Models — input-output relationship/function

Mathematical foundations

- Models — input-output relationship/function
- Optimization — ways to fit the models

Mathematical foundations

- Models — input-output relationship/function
- Optimization — ways to fit the models
- Calculus — gradients used for optimization

most methods so far are first-order only grad (not 2nd deriv)

Mathematical foundations

- Models — input-output relationship/function
- Optimization — ways to fit the models
- Calculus — gradients used for optimization
- Computer science — algorithms for calculating gradients

Linear algebra , parallelism

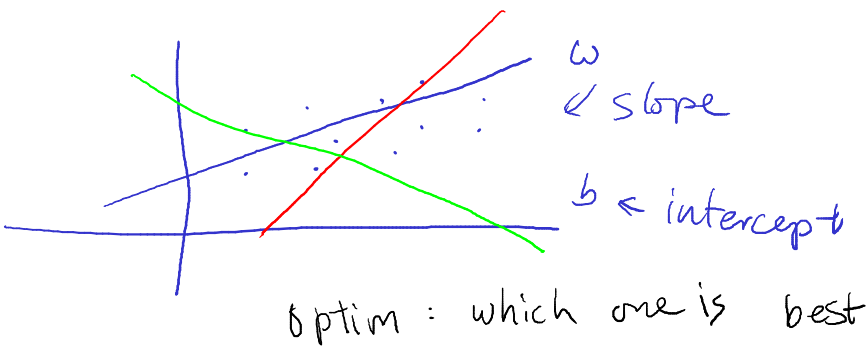
Mathematical foundations

- **Models**
- Optimization — ways to fit the models
- Calculus — gradients used for optimization
- Computer science — algorithms for calculating gradients

Linear estimators (1-dimensional samples)

$(x_i, y_i) \in \mathbb{R} \times \mathbb{R}$ and $\theta = (w, b) \in \mathbb{R} \times \mathbb{R}$

$$f(x; (w, b)) = w \cdot x + b$$



Mathematical foundations

- Models
- **Optimization — ways to fit the models**
- Calculus — gradients used for optimization
- Computer science — algorithms for calculating gradients

Loss and risk

Let $i = 1, \ldots, N$

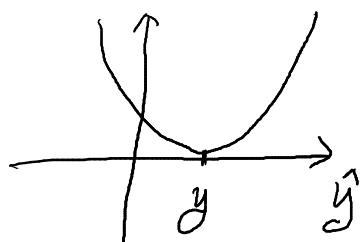
- Samples $\mathbf{x}_i \in \mathcal{X}$
- Labels $y_i \in \mathcal{Y}$
- Model params $\boldsymbol{\theta} \in \Theta$
- Prediction $\hat{y} = f(\mathbf{x}; \boldsymbol{\theta})$
- Loss $L(\hat{y}, y)$

Loss

estimator

ground truth

$$L(\hat{y}, y) = (\hat{y} - y)^2$$



Loss and risk

Let $i = 1, \ldots, N$

- Samples $\mathbf{x}_i \in \mathcal{X}$
- Labels $y_i \in \mathcal{Y}$
- Model params $\boldsymbol{\theta} \in \Theta$
- Prediction $\hat{y} = f(\mathbf{x}; \boldsymbol{\theta})$
- Loss $L(\hat{y}, y)$
- Empirical risk minimization (ERM)

"final" optimization

$\hat{y}_i$ y_i truth

versus

"population"

not Σ but $\mathbb{E}$

$$\min_{\boldsymbol{\theta} \in \Theta} \quad J(\boldsymbol{\theta}) := \frac{1}{N} \sum_{i=1}^{N} L(f(x_i; \boldsymbol{\theta}), y_i)$$

average

Mathematical foundations

- Models
- Optimization — ways to fit the models
- **Calculus — gradients used for optimization**
- Computer science — algorithms for calculating gradients

Gradients

given $f : \mathbb{R}^d \to \mathbb{R}$ differentiable

Let $\mathbf{x} = [x_1, \ldots, x_d]^\top$, the gradient is

↑ space

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \left[\frac{\partial f}{\partial x_1} f(\mathbf{x}) \quad \cdots \quad \frac{\partial f}{\partial x_1} f(\mathbf{x})\right]^\top$$

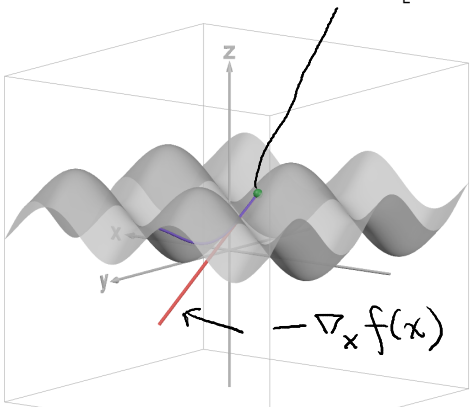
← partial deriv as a col vector

Gradients

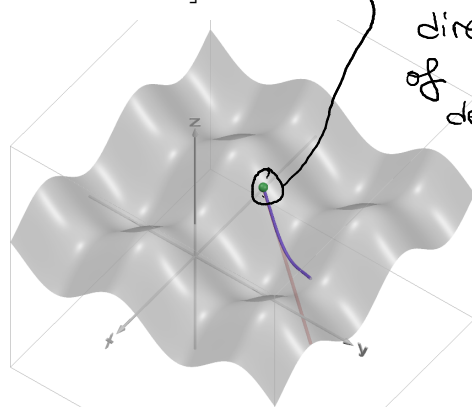
$f(x,y) = \sin(x)^2 + \cos(y)^2$

Let $\mathbf{x} = [x_1, \ldots, x_d]^\top$, the gradient is

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \left[\frac{\partial f}{\partial x_1} f(\mathbf{x}) \quad \cdots \quad \frac{\partial f}{\partial x_1} f(\mathbf{x})\right]^\top$$



locally direction of most decrease $-\nabla_x f(x)$



~~Sufficient~~ Necessary condition for local minimality

want to min

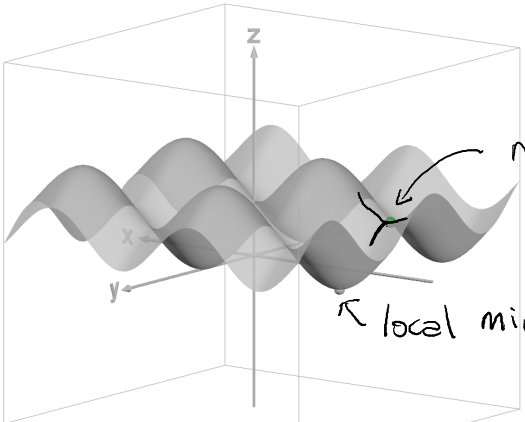
If $\mathbf{x}^*$ is a local minimum of f, then

$$\nabla_{\mathbf{x}} f(\mathbf{x}^*) = \mathbf{0}$$

$J(\theta)$

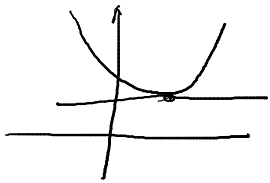
ERM

first-order vanishing



not local min but saddle

local min



Linear regression (1-dimensional samples)

Empirical risk minimization [GBC16, §8.1.1]

$$\min_{w\in\mathbb{R},b\in\mathbb{R}} \quad J(\boldsymbol{\theta}) := \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i;\boldsymbol{\theta}))^2$$

Linear regression (1-dimensional samples)

Empirical risk minimization [GBC16, §8.1.1]

$$\min_{w\in\mathbb{R},b\in\mathbb{R}} \quad J(\boldsymbol{\theta}) := \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i;\boldsymbol{\theta}))^2$$

Linear regression (1-dimensional samples)

Example: $N = 3$, $(x_1 = 1, y_1 = 2)$, $(x_2 = 2, y_2 = 3)$, $(x_3 = 3, y_3 = 1)$

$$N \cdot J(\theta) = \sum_{i=1}^{N} (y_i - (w x_i + b))^2$$

Step 1: rewrite as norm

find w & b minimizing $J(\theta)$, $\theta = \begin{bmatrix} w \\ b \end{bmatrix}$

$$= \left\| \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} - \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_N & 1 \end{bmatrix} \begin{bmatrix} w \\ b \end{bmatrix} \right\|^2$$

$\vec{y}$, $\tilde{X}$, θ

norm

$$\left\| \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_N \end{bmatrix} \right\|^2 := \sum_{i=1}^{N} z_i^2$$

$$= \|\vec{y} - \tilde{X}\theta\|^2$$

$(a-b)^2 = a^2 - 2ab + b^2$

$$= \vec{y}^T\vec{y} - 2\vec{y}^T\tilde{X}\theta + \theta^T\tilde{X}^T\tilde{X}\theta$$

Linear regression (1-dimensional samples)

Example: $N = 3$, $(x_1 = 1, y_1 = 2)$, $(x_2 = 2, y_2 = 3)$, $(x_3 = 3, y_3 = 1)$

$$J(\theta) = \vec{y}^T\vec{y} - 2\vec{y}^T\tilde{X}\theta + \theta^T\tilde{X}^T\tilde{X}\theta \quad \text{(copied)}$$

$$= \vec{y}^T\vec{y} - 2\theta^T\tilde{X}^T\vec{y} + \theta^T\tilde{X}^T\tilde{X}\theta \qquad a^Tb = b^Ta$$

$$\nabla_\theta J(\theta) = 0 \qquad -2\tilde{X}^T\vec{y} \qquad +2\tilde{X}^T\tilde{X}\theta$$

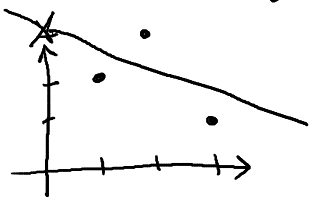
$$\nabla_a a^Tb = b \qquad \nabla_a a^TMa = 2Ma$$

"$\frac{d}{dx}ma^2 = 2ma$" vector version

Linear regression (1-dimensional samples)

Example: $N = 3$, $(x_1 = 1, y_1 = 2)$, $(x_2 = 2, y_2 = 3)$, $(x_3 = 3, y_3 = 1)$

$$0 = \nabla_\theta J(\theta) = -2\tilde{X}^T\vec{y} + 2\tilde{X}^T\tilde{X}\theta$$



↑ solve this equation

weight
↙ slope

$$\tilde{X} = \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_N & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{bmatrix}$$

$$\theta = (\underbrace{\tilde{X}^T\tilde{X}}_{2\times 2})^{-1}\tilde{X}^T\underbrace{\vec{y}}_{\begin{bmatrix}2\\3\\1\end{bmatrix}} = \begin{bmatrix} -0.5 \\ 3 \end{bmatrix}$$

↖ bias

Linear regression (1-dimensional samples)

Example: $N = 3$, $(x_1 = 1, y_1 = 2)$, $(x_2 = 2, y_2 = 3)$, $(x_3 = 3, y_3 = 1)$

$\|a - b\|^2 = ?$

$\|a\|^2 = \sum_{i=1}^{N} a_i^2 = a^T a = [a_1 \cdots a_N] \begin{bmatrix} a_1 \\ \vdots \\ a_N \end{bmatrix} = a_1^2 + \cdots + a_N^2$

why?

$a = \begin{bmatrix} a_1 \\ \vdots \\ a_N \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ \vdots \\ b_N \end{bmatrix}$

"expand the square of diff but for vectors"

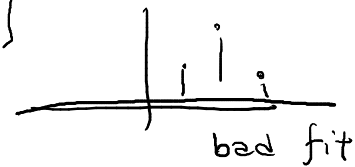
$\|a-b\|^2 = (a-b)^T(a-b) = a^Ta - a^Tb - b^Ta + b^Tb$

$= a^Ta - 2a^Tb + b^Tb$

Gradient descent (GD)

Let $\epsilon_k > 0$ be learning rates, $k = 1, 2, \ldots$

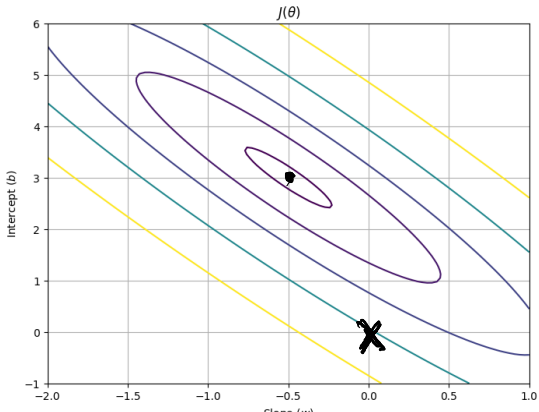
- Initialize $\boldsymbol{\theta}$ for example $\theta = \begin{bmatrix} w \\ b \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$
- While not converged (k = iteration counter):
  - Compute gradient $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
  - Compute update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon_k \mathbf{g}$



step size

(Goodfellow et al)

GD on our example



$J(\theta) = \frac{1}{3} \sum_i \cdots$

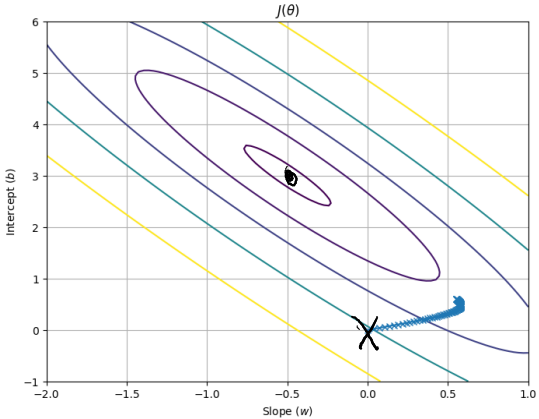
$\theta^* = \begin{bmatrix} -0.5 \\ 3 \end{bmatrix}$

"optimal θ"

Convention "*" means optimal

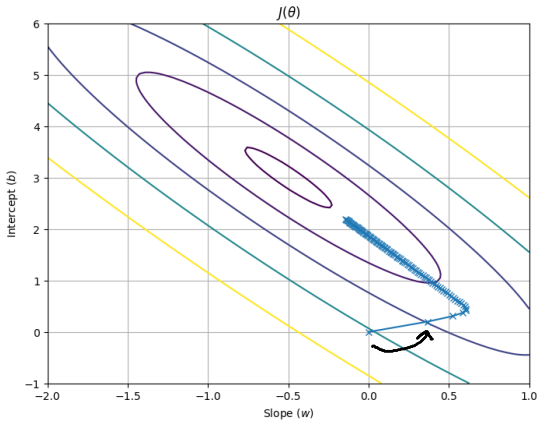
parameter space

GD on our example $\epsilon_k \equiv 0.005$ (100 iterations)



fixed

GD on our example $\epsilon_k \equiv 0.05$ (100 iterations)



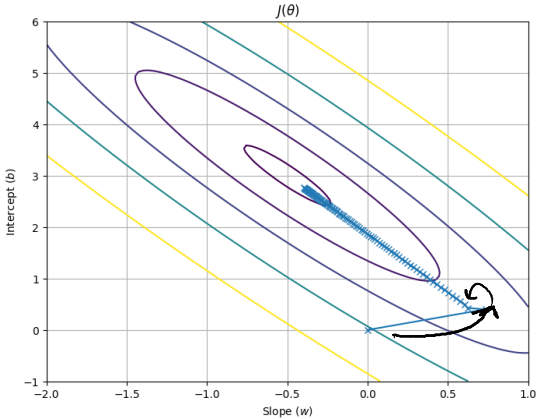
fixed < overshoot oscillation

IN THIS EXAMPLE

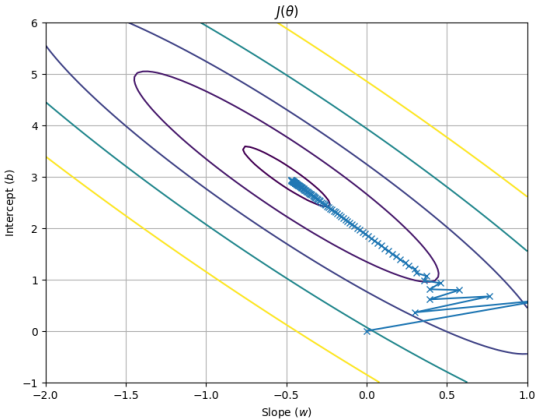
larger step size

better convergence speed

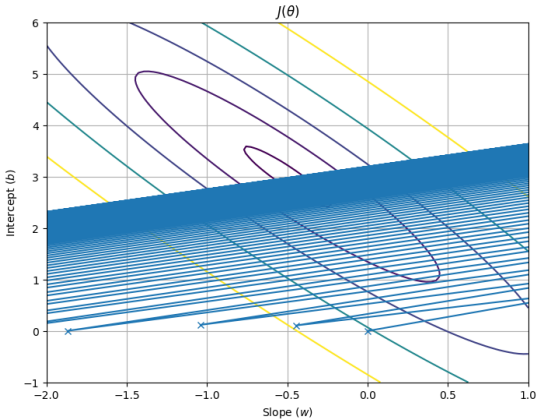
GD on our example $\epsilon_k \equiv 0.1$ (100 iterations)



GD on our example $\epsilon_k \equiv 0.15$ (100 iterations)



GD on our example $\epsilon_k \equiv 0.2$ (100 iterations)



Choice of ϵ_k is very sensitive

Stochastic gradient descent (SGD)

Let $\epsilon_k > 0$ be learning rates, $k = 1, 2, \ldots$

- Initialize $\boldsymbol{\theta}$
- While not converged (k = iteration counter):
  - Select m samples $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\}$ and matching labels $\{y^{(1)}, \ldots, y^{(m)}\}$
  - Compute gradient $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta}} \frac{1}{m} \sum_{i=1}^{m} L(f(\mathbf{x}^{(i)}, \boldsymbol{\theta}), y^{(i)})$
  - Compute update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon_k \mathbf{g}$

"minibatch"

e.g. 64 128

Previously took $g = \nabla_\theta J(\theta)$ "Full batch"

Too long to compute a single iteration

versus

$\frac{1}{N} \sum_{i=1}^{N} L(f(x^{(i)}, \theta), y^{(i)})$

estimating ↑ "training loss"

SGD on our example $\epsilon_k \equiv 0.1$ (100 iterations)

worked okay for GD

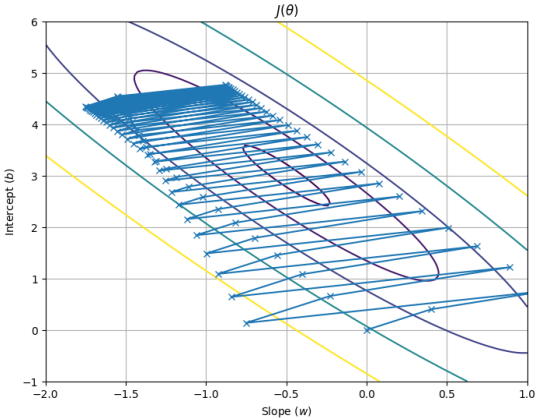
"minibatch" → m = 1

{ x1, x2, x3 | x1, x2, x3 | ... }

no shuffled

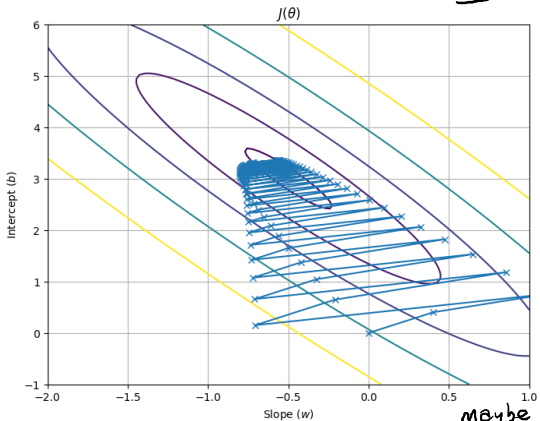
full batch = 3

minibatch = 1



SGD $\epsilon_k = (0.99)^k 0.1$ (first iter $k = 0$)

decay the learning rate



Why GD is actually better

Previous approach use matrix inversion

expensive for large data dimension (exact)

maybe good enough

GD inexact, but cheaper

Classification

Let $i = 1, \ldots, N$

- Samples $\mathbf{x}_i \in \mathcal{X}$
- Labels $y_i \in \mathcal{Y}$
- Model params $\boldsymbol{\theta} \in \Theta$
- Prediction $\hat{y} = f(\mathbf{x}; \boldsymbol{\theta})$

$\{\pm 1\}$ binary

$\{1, \cdots, K\}$ multiclass

example

next-token-prediction in LLM

Classification

Let $i = 1, \ldots, N$

- Samples $\mathbf{x}_i \in \mathcal{X}$
- Labels $y_i \in \mathcal{Y}$
- Model params $\boldsymbol{\theta} \in \Theta$
- Prediction $\hat{y} = f(\mathbf{x}; \boldsymbol{\theta})$
- Loss $L(\hat{y}, y)$

Classification

Let $i = 1, \ldots, N$

- Samples $\mathbf{x}_i \in \mathcal{X}$
- Labels $y_i \in \mathcal{Y}$
- Model params $\boldsymbol{\theta} \in \Theta$
- Prediction $\hat{y} = f(\mathbf{x}; \boldsymbol{\theta})$
- Loss $L(\hat{y}, y)$
- Empirical risk minimization (ERM)

$$\min_{\boldsymbol{\theta} \in \Theta} \quad J(\boldsymbol{\theta}) := \frac{1}{N} \sum_{i=1}^{N} L(f(x_i; \boldsymbol{\theta}), y_i)$$

Mathematical foundations

- **Models**
- Optimization — ways to fit the models
- Calculus — gradients used for optimization
- Computer science — algorithms for calculating gradients

(2-dimensional data)

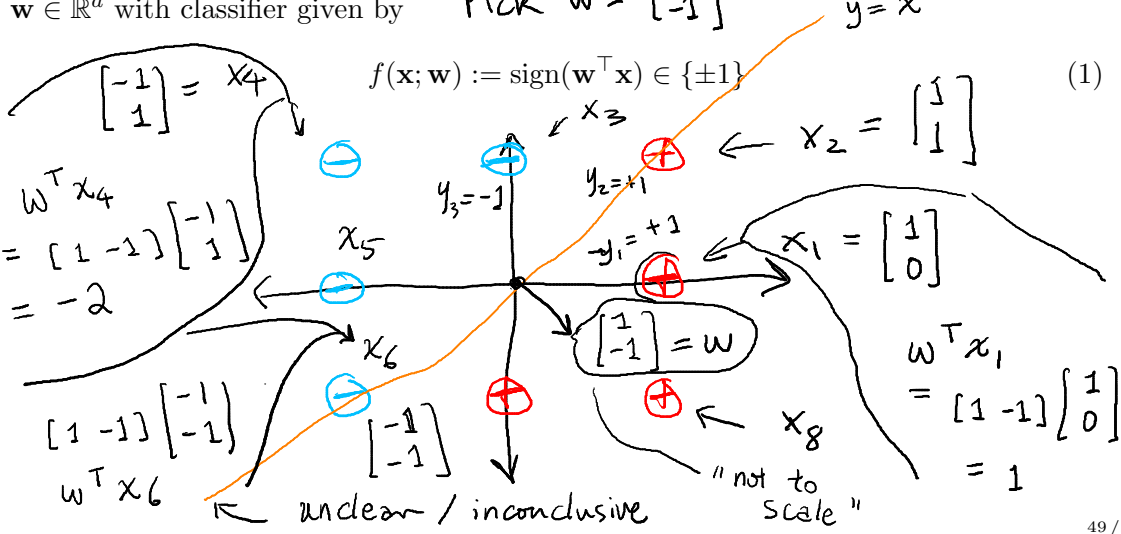
Binary linear classifier

(forget about the bias for now)

$\mathbf{w} \in \mathbb{R}^d$ with classifier given by

Pick $W = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$

$$f(\mathbf{x}; \mathbf{w}) := \operatorname{sign}(\mathbf{w}^\top \mathbf{x}) \in \{\pm 1\} \tag{1}$$



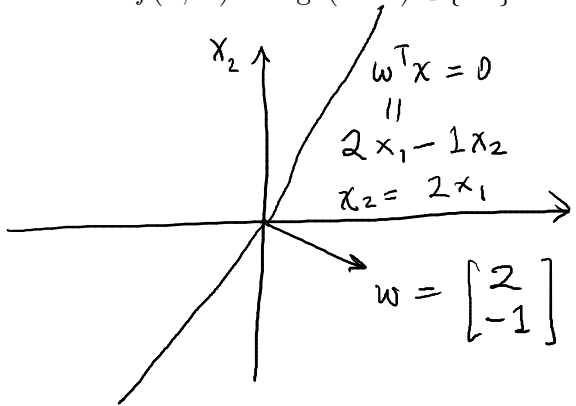
Binary linear classifier

Starting from this slide

$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

$\mathbf{w} \in \mathbb{R}^d$ with classifier given by

$$f(\mathbf{x}; \mathbf{w}) := \text{sign}(\mathbf{w}^\top \mathbf{x}) \in \{\pm 1\} \tag{1}$$

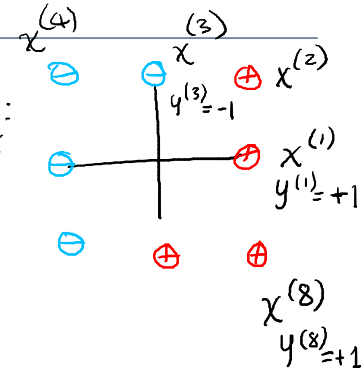


what is the set of $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \mathbb{R}^2$ s.t. $w^T \vec{x} = 0$

Perceptron ≠ "Multilayer perceptron"

$\mathbf{w} \in \mathbb{R}^d$ with classifier given by

$$f(\mathbf{x}; \mathbf{w}) := \text{sign}(\mathbf{w}^\top \mathbf{x}) \in \{\pm 1\}$$



Perceptron update

Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots$, time T

1. Initialize $\mathbf{w} = \mathbf{0}$.

Output: $\mathbf{w}$

16
loop back
if reaches
over the limit

Perceptron

$\mathbf{w} \in \mathbb{R}^d$ with classifier given by

$$f(\mathbf{x}; \mathbf{w}) := \text{sign}(\mathbf{w}^\top \mathbf{x}) \in \{\pm 1\}$$

Perceptron update

Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots$, time T

1. Initialize $\mathbf{w} = \mathbf{0}$.
2. For $t = 1, 2, \ldots, T$ =16

Output: $\mathbf{w}$

Perceptron

$\mathbf{w} \in \mathbb{R}^d$ with classifier given by

$$f(\mathbf{x}; \mathbf{w}) := \text{sign}(\mathbf{w}^\top \mathbf{x}) \in \{\pm 1\}$$

"Good classifier keep it as is"

Perceptron update

Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots,$ time T

1. Initialize $\mathbf{w} = \mathbf{0}$.
2. For $t = 1, 2, \ldots, T$
   - 2.1 If $y^{(t)} \mathbf{w}^\top \mathbf{x}^{(t)} > 0$, then $\mathbf{w} \leftarrow \mathbf{w}$,

Output: $\mathbf{w}$

Correct

prediction of model w at time t

ground truth of t-th sample

$w^\top x^{(t)}$ has sign $=$ $y^{(t)}$

Perceptron

$\mathbf{w} \in \mathbb{R}^d$ with classifier given by

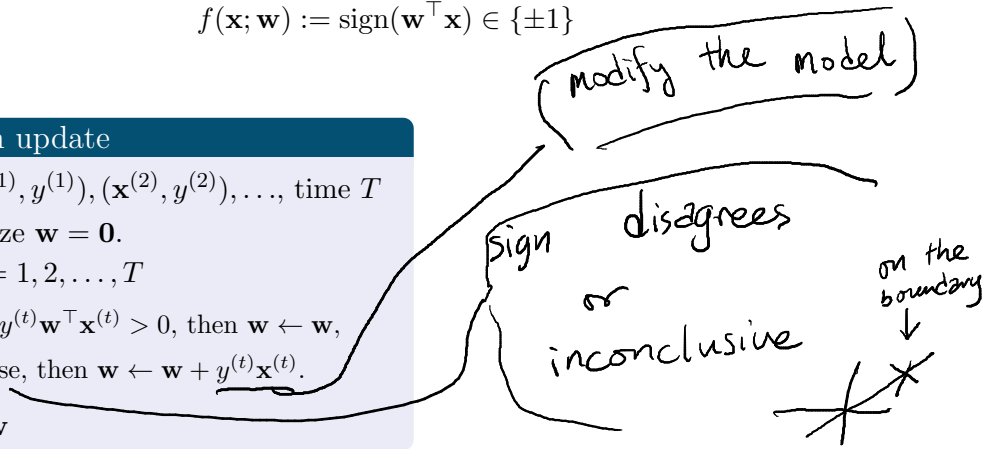
$$f(\mathbf{x}; \mathbf{w}) := \text{sign}(\mathbf{w}^\top \mathbf{x}) \in \{\pm 1\}$$

Perceptron update

Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots,$ time T

1. Initialize $\mathbf{w} = \mathbf{0}$.
2. For $t = 1, 2, \ldots, T$
   - 2.1 If $y^{(t)} \mathbf{w}^\top \mathbf{x}^{(t)} > 0$, then $\mathbf{w} \leftarrow \mathbf{w}$,
   - 2.2 Else, then $\mathbf{w} \leftarrow \mathbf{w} + y^{(t)} \mathbf{x}^{(t)}$.

Output: $\mathbf{w}$



Perceptron: an example

run to $T = 8$

$\mathbf{w} \in \mathbb{R}^d$ with classifier given by

$$f(\mathbf{x}; \mathbf{w}) := \text{sign}(\mathbf{w}^\top \mathbf{x}) \in \{\pm 1\}$$

Perceptron update

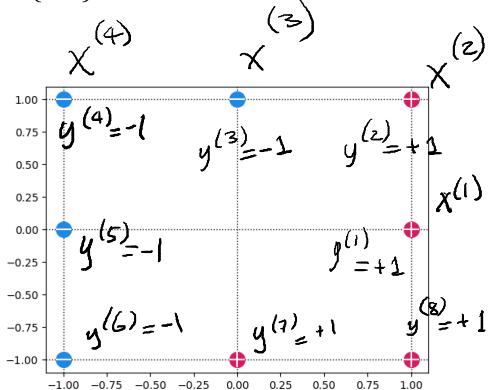
Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots,$ time T

1. Initialize $\mathbf{w} = \mathbf{0} \in \mathbb{R}^2$
2. For $t = 1, 2, \ldots, T$
   - 2.1 If $y^{(t)} \mathbf{w}^\top \mathbf{x}^{(t)} > 0$, then $\mathbf{w} \leftarrow \mathbf{w}$,
   - 2.2 Else, then $\mathbf{w} \leftarrow \mathbf{w} + y^{(t)} \mathbf{x}^{(t)}$.

Output: $\mathbf{w}$

scalar vector

entrywise multiplication



Perceptron: an example

$\mathbf{w} \in \mathbb{R}^d$ with classifier given by

$$f(\mathbf{x}; \mathbf{w}) := \text{sign}(\mathbf{w}^\top \mathbf{x}) \in \{\pm 1\}$$

$t = 1$

$x^{(1)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

$y^{(1)} = +1$

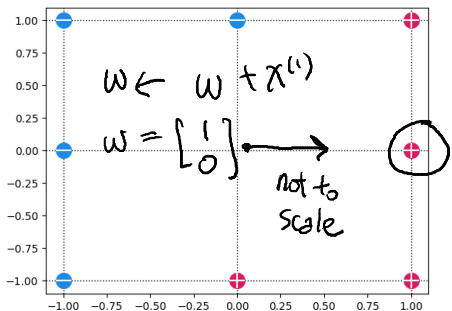
$w = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

Perceptron update

Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots,$ time T

1. Initialize $\mathbf{w} = \mathbf{0}$.
2. For $t = 1, 2, \ldots, T$
   - 2.1 If $y^{(t)} \mathbf{w}^\top \mathbf{x}^{(t)} > 0$, then $\mathbf{w} \leftarrow \mathbf{w}$,
   - 2.2 Else, then $\mathbf{w} \leftarrow \mathbf{w} + y^{(t)} \mathbf{x}^{(t)}$.

Output: $\mathbf{w}$



Perceptron: an example

$\mathbf{w} \in \mathbb{R}^d$ with classifier given by

$$f(\mathbf{x}; \mathbf{w}) := \text{sign}(\mathbf{w}^\top \mathbf{x}) \in \{\pm 1\}$$

$y^{(2)} = +1$

$x^{(2)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

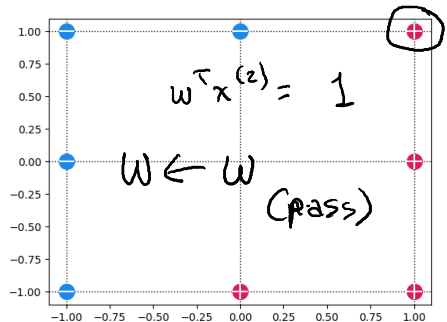
$w = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

Perceptron update

Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots,$ time T

1. Initialize $\mathbf{w} = \mathbf{0}$.
2. For $t = 1, 2, \ldots, T$
   - 2.1 If $y^{(t)} \mathbf{w}^\top \mathbf{x}^{(t)} > 0$, then $\mathbf{w} \leftarrow \mathbf{w}$,
   - 2.2 Else, then $\mathbf{w} \leftarrow \mathbf{w} + y^{(t)} \mathbf{x}^{(t)}$.

Output: $\mathbf{w}$



Perceptron: an example

$t=3$ inconclusive

$\mathbf{w} \in \mathbb{R}^d$ with classifier given by

$$f(\mathbf{x}; \mathbf{w}) := \text{sign}(\mathbf{w}^\top \mathbf{x}) \in \{\pm 1\}$$

$w = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

$y^{(3)} = -1$

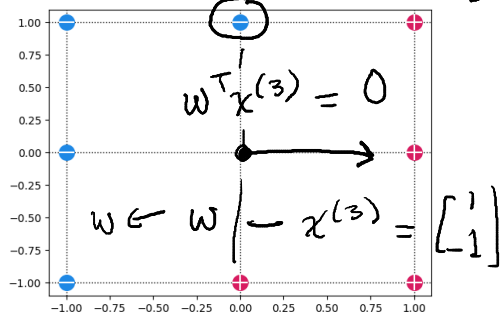
$x^{(3)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

Perceptron update

Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots,$ time T

1. Initialize $\mathbf{w} = \mathbf{0}$.
2. For $t = 1, 2, \ldots, T$
   - 2.1 If $y^{(t)} \mathbf{w}^\top \mathbf{x}^{(t)} > 0$, then $\mathbf{w} \leftarrow \mathbf{w}$,
   - 2.2 Else, then $\mathbf{w} \leftarrow \mathbf{w} + y^{(t)} \mathbf{x}^{(t)}$.

Output: $\mathbf{w}$



Perceptron: an example

$t = 4$

$w = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$

$\mathbf{w} \in \mathbb{R}^d$ with classifier given by

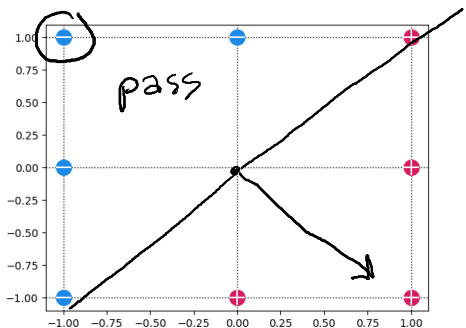
$$f(\mathbf{x}; \mathbf{w}) := \text{sign}(\mathbf{w}^\top \mathbf{x}) \in \{\pm 1\}$$

Perceptron update

Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots,$ time T

1. Initialize $\mathbf{w} = \mathbf{0}$.
2. For $t = 1, 2, \ldots, T$
   - 2.1 If $y^{(t)} \mathbf{w}^\top \mathbf{x}^{(t)} > 0$, then $\mathbf{w} \leftarrow \mathbf{w}$,
   - 2.2 Else, then $\mathbf{w} \leftarrow \mathbf{w} + y^{(t)} \mathbf{x}^{(t)}$.

Output: $\mathbf{w}$



Perceptron: an example

$t = 5$

$\mathbf{w} \in \mathbb{R}^d$ with classifier given by

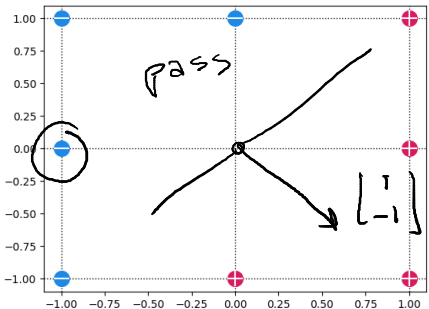
$$f(\mathbf{x}; \mathbf{w}) := \text{sign}(\mathbf{w}^\top \mathbf{x}) \in \{\pm 1\}$$

Perceptron update

Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots,$ time T

1. Initialize $\mathbf{w} = \mathbf{0}$.
2. For $t = 1, 2, \ldots, T$
   - 2.1 If $y^{(t)} \mathbf{w}^\top \mathbf{x}^{(t)} > 0$, then $\mathbf{w} \leftarrow \mathbf{w}$,
   - 2.2 Else, then $\mathbf{w} \leftarrow \mathbf{w} + y^{(t)} \mathbf{x}^{(t)}$.

Output: $\mathbf{w}$



Perceptron: an example

$t = 6$

$y^{(6)} = -1$

$x^{(6)} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$

$w \leftarrow w - \begin{bmatrix} -1 \\ -1 \end{bmatrix}$

$\mathbf{w} \in \mathbb{R}^d$ with classifier given by

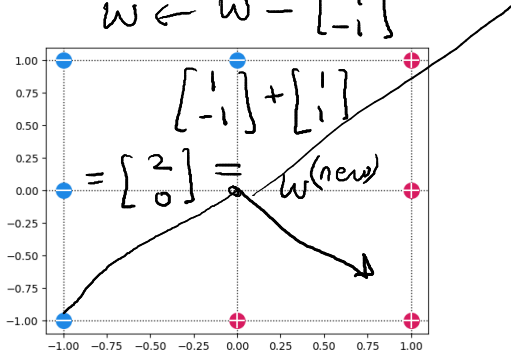
$$f(\mathbf{x}; \mathbf{w}) := \text{sign}(\mathbf{w}^\top \mathbf{x}) \in \{\pm 1\}$$

Perceptron update

Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots,$ time T

1. Initialize $\mathbf{w} = \mathbf{0}$.
2. For $t = 1, 2, \ldots, T$
   - 2.1 If $y^{(t)} \mathbf{w}^\top \mathbf{x}^{(t)} > 0$, then $\mathbf{w} \leftarrow \mathbf{w}$,
   - 2.2 Else, then $\mathbf{w} \leftarrow \mathbf{w} + y^{(t)} \mathbf{x}^{(t)}$.

Output: $\mathbf{w}$



Perceptron: an example

$\mathbf{w} \in \mathbb{R}^d$ with classifier given by

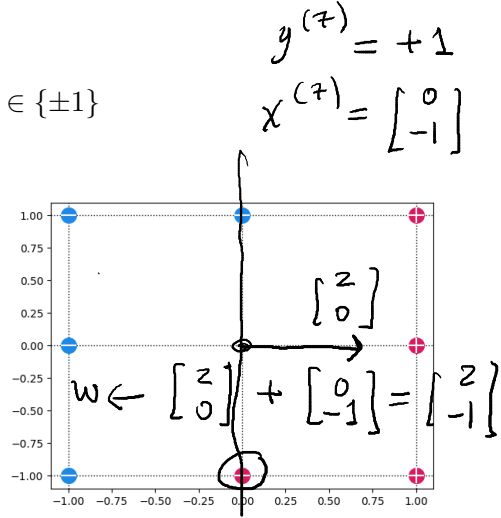
$$f(\mathbf{x}; \mathbf{w}) := \text{sign}(\mathbf{w}^\top \mathbf{x}) \in \{\pm 1\}$$

Perceptron update

Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots,$ time T

1. Initialize $\mathbf{w} = \mathbf{0}$.
2. For $t = 1, 2, \ldots, T$
   - 2.1 If $y^{(t)} \mathbf{w}^\top \mathbf{x}^{(t)} > 0$, then $\mathbf{w} \leftarrow \mathbf{w}$,
   - 2.2 Else, then $\mathbf{w} \leftarrow \mathbf{w} + y^{(t)} \mathbf{x}^{(t)}$.

Output: $\mathbf{w}$



Perceptron: an example

Converged

$\mathbf{w} \in \mathbb{R}^d$ with classifier given by

$$f(\mathbf{x}; \mathbf{w}) := \text{sign}(\mathbf{w}^\top \mathbf{x}) \in \{\pm 1\}$$

$w = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$

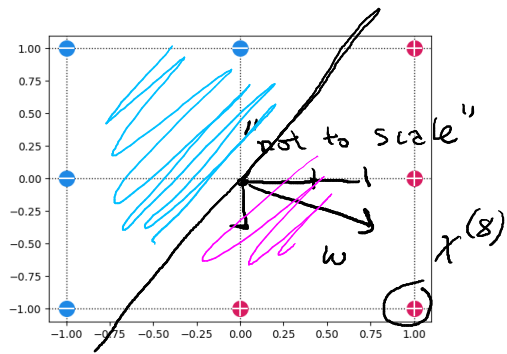
$x_2 = 2x_1$

Perceptron update

Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots,$ time T

1. Initialize $\mathbf{w} = \mathbf{0}$.
2. For $t = 1, 2, \ldots, T$
   - 2.1 If $y^{(t)} \mathbf{w}^\top \mathbf{x}^{(t)} > 0$, then $\mathbf{w} \leftarrow \mathbf{w}$,
   - 2.2 Else, then $\mathbf{w} \leftarrow \mathbf{w} + y^{(t)} \mathbf{x}^{(t)}$.

Output: $\mathbf{w}$



Perceptron: an example

$\mathbf{w} \in \mathbb{R}^d$ with classifier given by

$$f(\mathbf{x}; \mathbf{w}) := \text{sign}(\mathbf{w}^\top \mathbf{x}) \in \{\pm 1\}$$

$t = 1$

$y^{(1)} = +1$

$w = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$x^{(1)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

Perceptron update

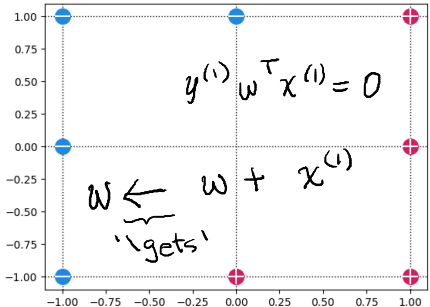
Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots,$ time T

1. Initialize $\mathbf{w} = \mathbf{0}$.
2. For $t = 1, 2, \ldots, T$

   2.1 If $y^{(t)} \mathbf{w}^\top \mathbf{x}^{(t)} > 0$, then $\mathbf{w} \leftarrow \mathbf{w}$,

   2.2 Else, then $\mathbf{w} \leftarrow \mathbf{w} + y^{(t)} \mathbf{x}^{(t)}$.

Output: $\mathbf{w}$



Perceptron: is this SGD?

$\mathbf{w} \in \mathbb{R}^d$ with classifier given by

$$f(\mathbf{x}; \mathbf{w}) := \text{sign}(\mathbf{w}^\top \mathbf{x}) \in \{\pm 1\}$$

Perceptron update

Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots,$ time T

1. Initialize $\mathbf{w} = \mathbf{0}$.
2. For $t = 1, 2, \ldots, T$
   - 2.1 If $y^{(t)} \mathbf{w}^\top \mathbf{x}^{(t)} > 0$, then $\mathbf{w} \leftarrow \mathbf{w}$,
   - 2.2 Else, then $\mathbf{w} \leftarrow \mathbf{w} + y^{(t)} \mathbf{x}^{(t)}$.

Output: $\mathbf{w}$

- Compute gradient

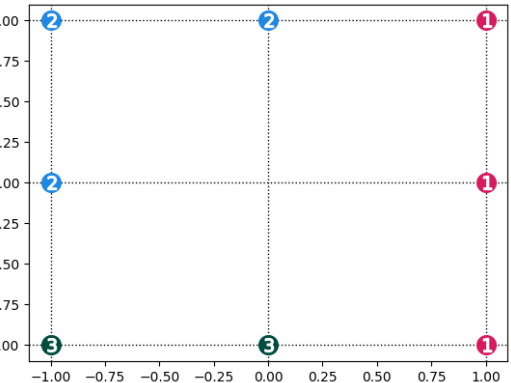
$$\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta}} \frac{1}{m} \sum_{i=1}^{m} L(f(\mathbf{x}^{(i)}, \boldsymbol{\theta}), y^{(i)})$$

- Compute update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon_k \mathbf{g}$

Multiclass linear classifier (first attempt)

$\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 & \cdots & \mathbf{w}_K \end{bmatrix} \in \mathbb{R}^{d \times K}$ with classifier given by

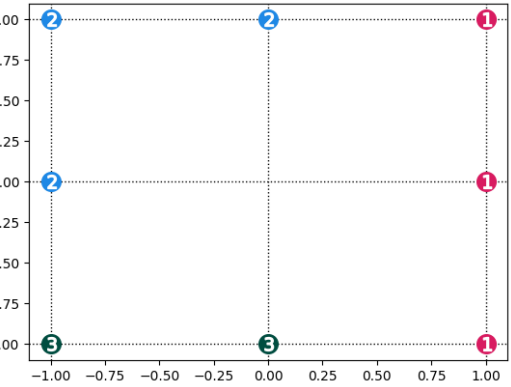
$$f(\mathbf{x}; \mathbf{W}) := \operatorname{argmax}_{\hat{y}=1,\ldots,K} \quad \mathbf{w}_{\hat{y}}^\top \mathbf{x} \quad \in \{1, \ldots, K\} \tag{2}$$



Multiclass linear classifier (second attempt)

$\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 & \cdots & \mathbf{w}_K \end{bmatrix} \in \mathbb{R}^{d \times K}$ with classifier given by

$$f(\mathbf{x}; \mathbf{W}) := \operatorname{argmax}_{\hat{y}=1,\ldots,K} \quad \mathbf{w}_{\hat{y}}^\top \mathbf{x} \quad \in \{1, \ldots, K\} \tag{3}$$



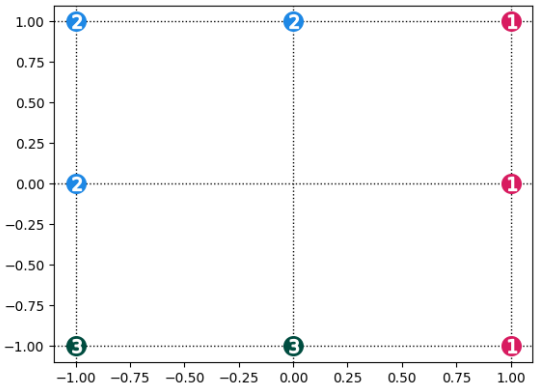
Multiclass perceptron

Perceptron update

Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots,$ time T

1. Initialize $\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 & \cdots & \mathbf{w}_K \end{bmatrix} = \mathbf{0}$.

Output: $\mathbf{W}$



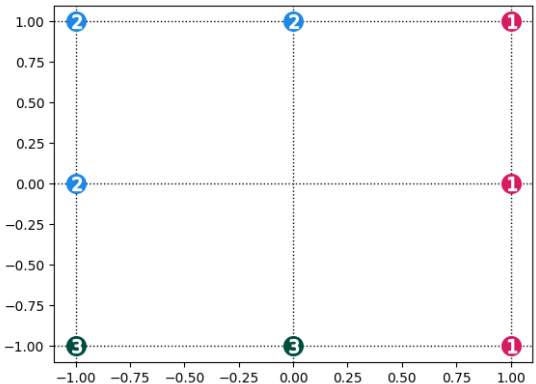
Multiclass perceptron

Perceptron update

Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots,$ time T

1. Initialize $\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 & \cdots & \mathbf{w}_K \end{bmatrix} = \mathbf{0}$.
2. For $t = 1, 2, \ldots, T$

Output: $\mathbf{W}$



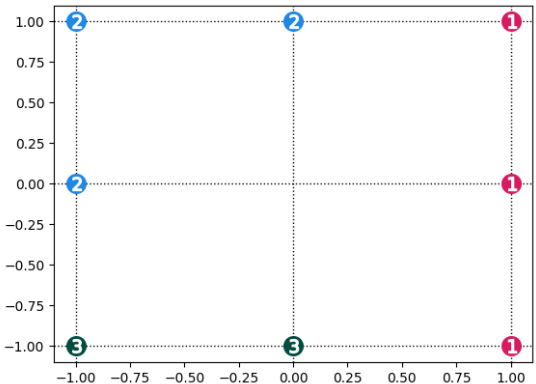
Multiclass perceptron

Perceptron update

Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots,$ time T

1. Initialize $\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 & \cdots & \mathbf{w}_K \end{bmatrix} = \mathbf{0}$.
2. For $t = 1, 2, \ldots, T$
   2.1 $\hat{y}^{(t)} \leftarrow \text{argmax}_{\hat{y}} \mathbf{w}_{\hat{y}}^\top \mathbf{x}^{(t)}$

Output: $\mathbf{W}$



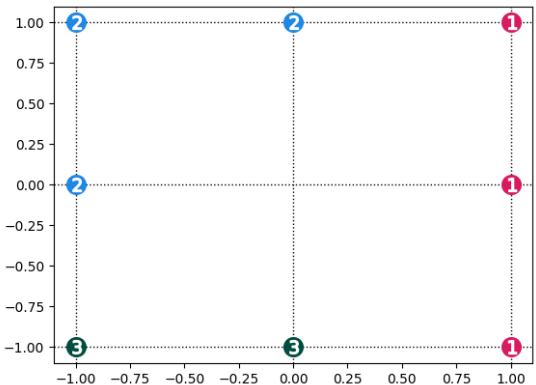
Multiclass perceptron

Perceptron update

Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots,$ time T

1. Initialize $\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 & \cdots & \mathbf{w}_K \end{bmatrix} = \mathbf{0}$.
2. For $t = 1, 2, \ldots, T$
   - 2.1 $\hat{y}^{(t)} \leftarrow \text{argmax}_{\hat{y}} \mathbf{w}_{\hat{y}}^\top \mathbf{x}^{(t)}$
   - 2.2 If $\hat{y}^{(t)} = y^{(t)}$, then $\mathbf{W} \leftarrow \mathbf{W}$,

Output: $\mathbf{W}$



Multiclass perceptron

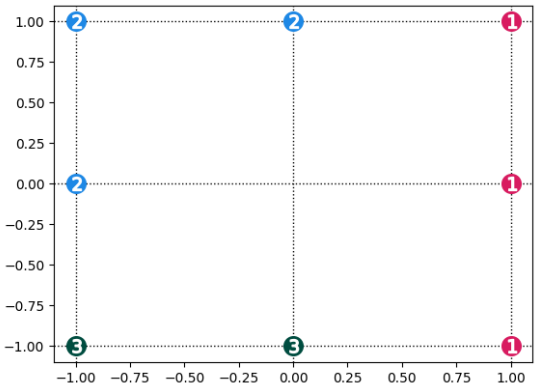
Perceptron update

Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots,$ time T

1. Initialize $\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 & \cdots & \mathbf{w}_K \end{bmatrix} = \mathbf{0}$.
2. For $t = 1, 2, \ldots, T$
   - 2.1 $\hat{y}^{(t)} \leftarrow \operatorname{argmax}_{\hat{y}} \mathbf{w}_{\hat{y}}^\top \mathbf{x}^{(t)}$
   - 2.2 If $\hat{y}^{(t)} = y^{(t)}$, then $\mathbf{W} \leftarrow \mathbf{W}$,
   - 2.3 Else, then

$$\mathbf{w}_{\hat{y}^{(t)}} \leftarrow \mathbf{w}_{\hat{y}^{(t)}} - \mathbf{x}^{(t)}$$
$$\mathbf{w}_{y^{(t)}} \leftarrow \mathbf{w}_{y^{(t)}} + \mathbf{x}^{(t)}$$

Output: $\mathbf{W}$



Multiclass perceptron (calculations)

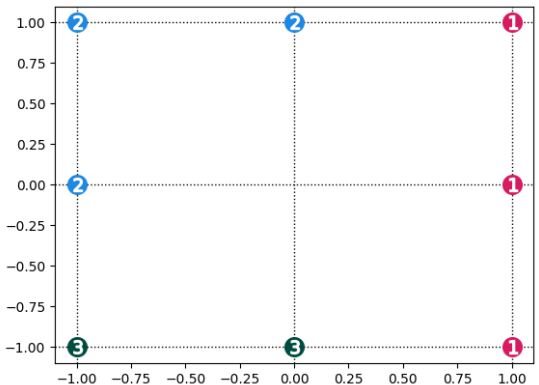
Perceptron update

Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots,$ time T

1. Initialize $\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 & \cdots & \mathbf{w}_K \end{bmatrix} = \mathbf{0}$.
2. For $t = 1, 2, \ldots, T$
   - 2.1 $\hat{y}^{(t)} \leftarrow \text{argmax}_{\hat{y}} \mathbf{w}_{\hat{y}}^\top \mathbf{x}^{(t)}$
   - 2.2 If $\hat{y}^{(t)} = y^{(t)}$, then $\mathbf{W} \leftarrow \mathbf{W}$,
   - 2.3 Else, then

$$\mathbf{w}_{\hat{y}^{(t)}} \leftarrow \mathbf{w}_{\hat{y}^{(t)}} - \mathbf{x}^{(t)}$$
$$\mathbf{w}_{y^{(t)}} \leftarrow \mathbf{w}_{y^{(t)}} + \mathbf{x}^{(t)}$$

Output: $\mathbf{W}$



Multiclass perceptron (calculations)

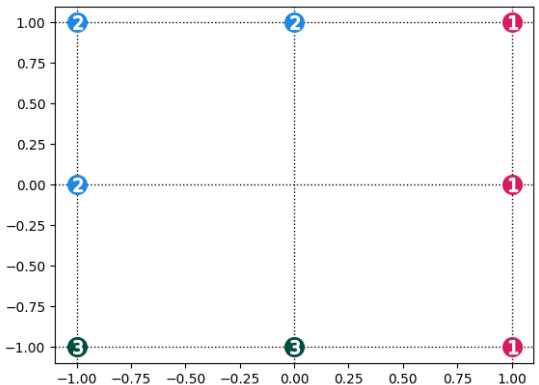
Perceptron update

Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots,$ time T

1. Initialize $\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 & \cdots & \mathbf{w}_K \end{bmatrix} = \mathbf{0}$.
2. For $t = 1, 2, \ldots, T$
   - 2.1 $\hat{y}^{(t)} \leftarrow \text{argmax}_{\hat{y}} \mathbf{w}_{\hat{y}}^\top \mathbf{x}^{(t)}$
   - 2.2 If $\hat{y}^{(t)} = y^{(t)}$, then $\mathbf{W} \leftarrow \mathbf{W}$,
   - 2.3 Else, then

$$\mathbf{w}_{\hat{y}^{(t)}} \leftarrow \mathbf{w}_{\hat{y}^{(t)}} - \mathbf{x}^{(t)}$$
$$\mathbf{w}_{y^{(t)}} \leftarrow \mathbf{w}_{y^{(t)}} + \mathbf{x}^{(t)}$$

Output: $\mathbf{W}$



Multiclass perceptron (calculations)

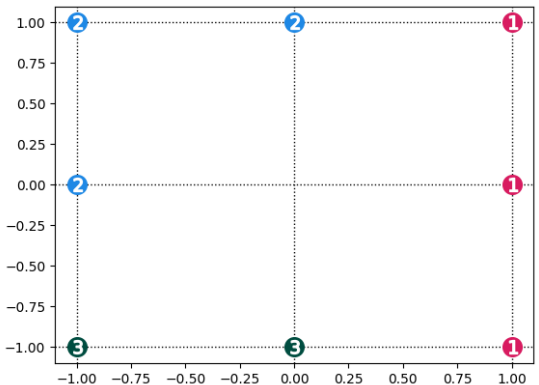
Perceptron update

Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots,$ time T

1. Initialize $\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 & \cdots & \mathbf{w}_K \end{bmatrix} = \mathbf{0}$.
2. For $t = 1, 2, \ldots, T$
   - 2.1 $\hat{y}^{(t)} \leftarrow \text{argmax}_{\hat{y}} \mathbf{w}_{\hat{y}}^\top \mathbf{x}^{(t)}$
   - 2.2 If $\hat{y}^{(t)} = y^{(t)}$, then $\mathbf{W} \leftarrow \mathbf{W}$,
   - 2.3 Else, then

$$\mathbf{w}_{\hat{y}^{(t)}} \leftarrow \mathbf{w}_{\hat{y}^{(t)}} - \mathbf{x}^{(t)}$$
$$\mathbf{w}_{y^{(t)}} \leftarrow \mathbf{w}_{y^{(t)}} + \mathbf{x}^{(t)}$$

Output: $\mathbf{W}$



Multiclass perceptron (calculations)

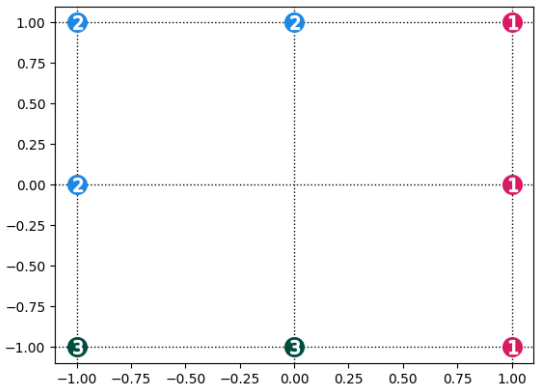
Perceptron update

Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots,$ time T

1. Initialize $\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 & \cdots & \mathbf{w}_K \end{bmatrix} = \mathbf{0}$.
2. For $t = 1, 2, \ldots, T$
   - 2.1 $\hat{y}^{(t)} \leftarrow \text{argmax}_{\hat{y}} \mathbf{w}_{\hat{y}}^\top \mathbf{x}^{(t)}$
   - 2.2 If $\hat{y}^{(t)} = y^{(t)}$, then $\mathbf{W} \leftarrow \mathbf{W}$,
   - 2.3 Else, then

$$\mathbf{w}_{\hat{y}^{(t)}} \leftarrow \mathbf{w}_{\hat{y}^{(t)}} - \mathbf{x}^{(t)}$$
$$\mathbf{w}_{y^{(t)}} \leftarrow \mathbf{w}_{y^{(t)}} + \mathbf{x}^{(t)}$$

Output: $\mathbf{W}$



Multiclass perceptron (calculations)

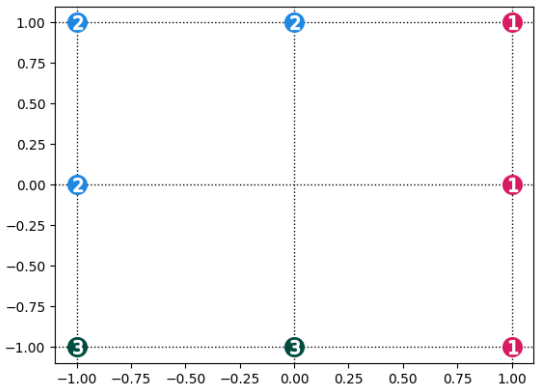
Perceptron update

Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots,$ time T

1. Initialize $\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 & \cdots & \mathbf{w}_K \end{bmatrix} = \mathbf{0}$.
2. For $t = 1, 2, \ldots, T$
   - 2.1 $\hat{y}^{(t)} \leftarrow \text{argmax}_{\hat{y}} \mathbf{w}_{\hat{y}}^\top \mathbf{x}^{(t)}$
   - 2.2 If $\hat{y}^{(t)} = y^{(t)}$, then $\mathbf{W} \leftarrow \mathbf{W}$,
   - 2.3 Else, then

$$\mathbf{w}_{\hat{y}^{(t)}} \leftarrow \mathbf{w}_{\hat{y}^{(t)}} - \mathbf{x}^{(t)}$$
$$\mathbf{w}_{y^{(t)}} \leftarrow \mathbf{w}_{y^{(t)}} + \mathbf{x}^{(t)}$$

Output: $\mathbf{W}$



Multiclass perceptron (calculations)

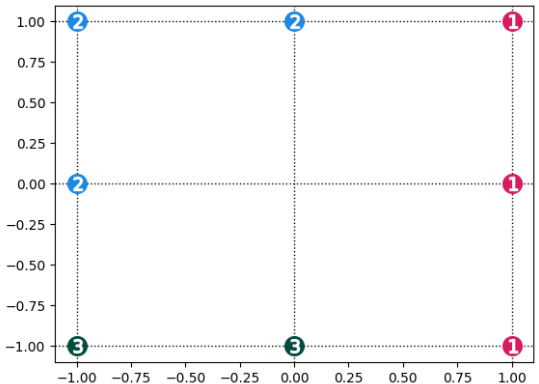
Perceptron update

Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots,$ time T

1. Initialize $\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 & \cdots & \mathbf{w}_K \end{bmatrix} = \mathbf{0}$.
2. For $t = 1, 2, \ldots, T$
   - 2.1 $\hat{y}^{(t)} \leftarrow \text{argmax}_{\hat{y}} \mathbf{w}_{\hat{y}}^\top \mathbf{x}^{(t)}$
   - 2.2 If $\hat{y}^{(t)} = y^{(t)}$, then $\mathbf{W} \leftarrow \mathbf{W}$,
   - 2.3 Else, then

$$\mathbf{w}_{\hat{y}^{(t)}} \leftarrow \mathbf{w}_{\hat{y}^{(t)}} - \mathbf{x}^{(t)}$$
$$\mathbf{w}_{y^{(t)}} \leftarrow \mathbf{w}_{y^{(t)}} + \mathbf{x}^{(t)}$$

Output: $\mathbf{W}$



Multiclass perceptron (calculations)

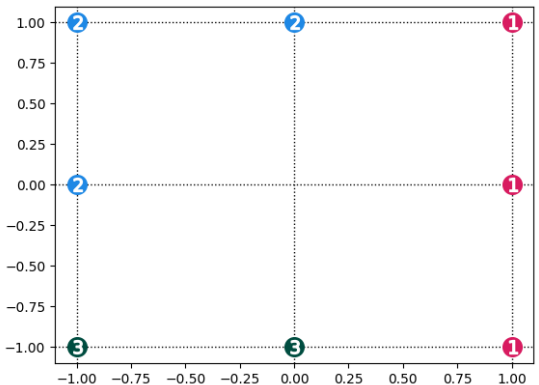
Perceptron update

Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots,$ time T

1. Initialize $\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 & \cdots & \mathbf{w}_K \end{bmatrix} = \mathbf{0}$.
2. For $t = 1, 2, \ldots, T$
   - 2.1 $\hat{y}^{(t)} \leftarrow \text{argmax}_{\hat{y}} \mathbf{w}_{\hat{y}}^\top \mathbf{x}^{(t)}$
   - 2.2 If $\hat{y}^{(t)} = y^{(t)}$, then $\mathbf{W} \leftarrow \mathbf{W}$,
   - 2.3 Else, then

$$\mathbf{w}_{\hat{y}^{(t)}} \leftarrow \mathbf{w}_{\hat{y}^{(t)}} - \mathbf{x}^{(t)}$$
$$\mathbf{w}_{y^{(t)}} \leftarrow \mathbf{w}_{y^{(t)}} + \mathbf{x}^{(t)}$$

Output: $\mathbf{W}$



Multiclass perceptron (calculations)

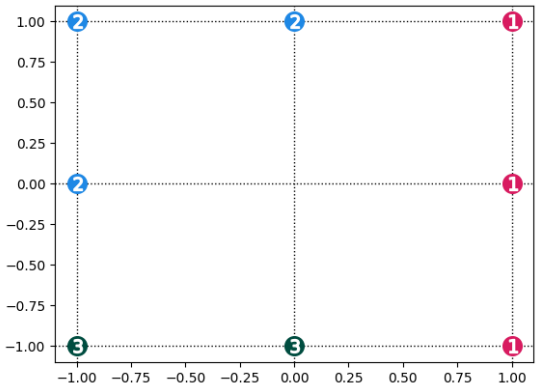
Perceptron update

Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots,$ time T

1. Initialize $\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 & \cdots & \mathbf{w}_K \end{bmatrix} = \mathbf{0}$.
2. For $t = 1, 2, \ldots, T$
   - 2.1 $\hat{y}^{(t)} \leftarrow \text{argmax}_{\hat{y}} \mathbf{w}_{\hat{y}}^\top \mathbf{x}^{(t)}$
   - 2.2 If $\hat{y}^{(t)} = y^{(t)}$, then $\mathbf{W} \leftarrow \mathbf{W}$,
   - 2.3 Else, then

$$\mathbf{w}_{\hat{y}^{(t)}} \leftarrow \mathbf{w}_{\hat{y}^{(t)}} - \mathbf{x}^{(t)}$$
$$\mathbf{w}_{y^{(t)}} \leftarrow \mathbf{w}_{y^{(t)}} + \mathbf{x}^{(t)}$$

Output: $\mathbf{W}$



References I

[BGV92] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. "A training algorithm for optimal margin classifiers". In: *Proceedings of the fifth annual workshop on Computational learning theory*. 1992, pp. 144–152.

[Gar+21] Camille Garcin, Alexis Joly, Pierre Bonnet, Jean-Christophe Lombardo, Antoine Affouard, Mathias Chouet, Maximilien Servajean, Titouan Lorieul, and Joseph Salmon. "Pl@ ntNet-300K: a plant image dataset with high label ambiguity and a long-tailed distribution". In: *NeurIPS 2021-35th Conference on Neural Information Processing Systems*. 2021.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT press, 2016.

References II

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012).

[SE19] Yang Song and Stefano Ermon. "Generative modeling by estimating gradients of the data distribution". In: *Advances in Neural Information Processing Systems* 32 (2019).

[Soh+15] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. "Deep unsupervised learning using nonequilibrium thermodynamics". In: *International conference on machine learning*. PMLR. 2015, pp. 2256–2265.

References III

[Vas+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: *Advances in Neural Information Processing Systems* 30 (2017).