Nonlinearity

Lecture 03 — CS 577 Deep Learning

Instructor: Yutong Wang

Computer Science
Illinois Institute of Technology

September 4, 2024

Administrative matter

- For the course project, you can form your own groups (2-4 people) or have the groups be assigned to you randomly.

Notations

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

- Training samples $\mathbf{x}^{(i)} \in \mathcal{X} \subseteq \mathbb{R}^d$ dimensional
- Labels $y^{(i)} \in \mathcal{Y} = \mathbb{R}$
- $f(\cdot; \boldsymbol{\theta}) : \mathcal{X} \to \mathcal{Y}$

Linearity

Definition. $f(\cdot;\boldsymbol{\theta})$ is *linear* if there exists some C such that

$$f(\mathbf{x}+\mathbf{x}';\boldsymbol{\theta}) - C = f(x;\theta) + f(x';\theta) - C$$

$$f(\lambda\mathbf{x};\boldsymbol{\theta}) - C = \lambda f(\lambda x;\theta) - \lambda C$$

Note: See [GBC16, p. 5.1.4] regarding "affine" vs "linear"

Linearity

Definition. $f(\cdot\,;\boldsymbol{\theta})$ is *linear* if there exists some C such that

$$f(\mathbf{x}+\mathbf{x}';\boldsymbol{\theta}) - C = w^{\top}(x+x')+b$$

$$f(\lambda\mathbf{x};\boldsymbol{\theta}) - C = w^{\top}x + w^{\top}x' + b + b - c$$

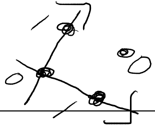
Example (linear regression).

$$f\,(\mathbf{x};\mathbf{w},b) = \mathbf{w}^{\top}\mathbf{x}+b \qquad f(x;\theta)+f(x';\theta) - C$$

Note: Limitations of linearity

Do: `lec03-in-class-ex1-xor.ipynb`

Discuss: does "learning" occur?



Set to
$C = b$

Linearity

Definition. $f(\cdot\,; \boldsymbol{\theta})$ is *linear* if there exists some C such that

It seems like $w \to 0$.

$$f(\mathbf{x} + \mathbf{x}'; \boldsymbol{\theta}) - C =$$

$$f(\lambda \mathbf{x}; \boldsymbol{\theta}) - C =$$

What does this mean?

Example (linear regression).

$$f\left(\mathbf{x}; \mathbf{w}, b\right) = \mathbf{w}^{\top}\mathbf{x} + b$$

y doesn't depend on x

> **Note:** Limitations of linearity
> Do: `lec03-in-class-ex1-xor.ipynb`
> Discuss: does "learning" occur?

Not at all

$f_{true}(x_1, x_2) = x_1$ exclusive x_2

Linearity

Definition. $f(\cdot\,;\boldsymbol{\theta})$ is *linear* if there exists some C such that

$$f(\mathbf{x}+\mathbf{x}';\boldsymbol{\theta}) - C =$$

$$f(\lambda\mathbf{x};\boldsymbol{\theta}) - C =$$

Example (linear regression).

$$f\left(\mathbf{x};\mathbf{w},b\right) = \mathbf{w}^{\top}\mathbf{x} + b$$

Note: Limitations of linearity
Do: `lec03-in-class-ex1-xor.ipynb`
Discuss: does "learning" occur?

! Empirical observation !

Beyong linearity: feature map

Definition. $f(\cdot\,;\boldsymbol{\theta})$ is *linear* if there exists some C such that

Before

$w^T \underline{x} + b$

$$f(\mathbf{x}+\mathbf{x}';\boldsymbol{\theta}) - C =$$

$$f(\lambda\mathbf{x};\boldsymbol{\theta}) - C =$$

Example (linear regression with a (fixed) feature map). $\phi : \mathbb{R}^1 \to \mathbb{R}^D$

$$f\,(x;\mathbf{w},b) = \mathbf{w}^\top \underline{\phi(x)} + b$$

$\tilde{x}$ = engineered features

degree d polynomial feature map

$\phi(x) = [1, x, x^2, \cdots, x^d]$

For 1-D data only

? Can we apply it to XOR?

Beyong linearity: feature map

inner func

```
def polynomial_feature_map_(x,degree):
  return [x**d for d in range(degree+1)]

def polynomial_feature_map(x_array,degree):
  return np.array([polynomial_feature_map_(x_array[i],degree) for i in range(len(x_array))])

deg = 37

# has a single HYPERPARAMETER -> deg
Xtilde = polynomial_feature_map(x, deg)
```

outer func

$= \text{degree} = 2$

$\{ ax^2 + bx + c :$ poly in 1 unknown

$a, b, c \in \mathbb{R} \}$

Beyong linearity: feature map

```
def polynomial_feature_map_(x,degree):
  return [x**d for d in range(degree+1)]

def polynomial_feature_map(x_array,degree):
  return np.array([polynomial_feature_map_(x_array[i],degree) for i in range(len(x_array))])

deg = 37

# has a single HYPERPARAMETER -> deg
Xtilde = polynomial_feature_map(x, deg)
```

Poly deg 2 $= ax^2 + bx + c = \phi(x)$

$$= \underbrace{\begin{bmatrix} a & b & c \end{bmatrix}}_{w} \begin{bmatrix} x^2 \\ x \\ 1 \end{bmatrix}$$

Beyong linearity: feature map

```
def polynomial_feature_map_(x,degree):
  return [x**d for d in range(degree+1)]

def polynomial_feature_map(x_array,degree):
  return np.array([polynomial_feature_map_(x_array[i],degree) for i in range(len(x_array))])

deg = 37

# has a single HYPERPARAMETER -> deg
Xtilde = polynomial_feature_map(x, deg)
```

poly of deg 2 in 2 free variable

$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ $\quad a x_2^2 + b x_1^2 + c x_1 x_2 + d x_1 + e x_2 + f \cdot 1$

Beyong linearity: feature map

Can capture XOR with deg 2 poly feat map

Definition. $f(\cdot\,;\boldsymbol{\theta})$ is *linear* if there exists some C such that

$$f(\mathbf{x}+\mathbf{x}';\boldsymbol{\theta}) - C =$$

$$f(\lambda\mathbf{x};\boldsymbol{\theta}) - C =$$

Example (linear regression with a (fixed) feature map). $\phi : \mathbb{R}^d \to \mathbb{R}^D$

$$f\,(\mathbf{x};\mathbf{w},b) = \mathbf{w}^\top\phi(\mathbf{x}) + b$$

Note: Implement the feature map for degree 2 polynomials in 2 free variables.

$d = 2 \qquad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad D = ?$

all degree 2 polynomial feature

Linearity

Example (linear regression with a (varying) feature map). $\phi : \mathbb{R}^d \to \mathbb{R}^D$

$$f\left(\mathbf{x}; \mathbf{w}^{(2)}, b^{(2)}\right) = \mathbf{w}^{(2)\top}\phi(\mathbf{x}) + b^{(2)}$$

$$\phi(x; W^{(1)}, b^{(1)}) = \qquad \phi(\mathbf{x}) = \underbrace{W^{(1)\top}}_{} x + b^{(1)}$$

feature map with tunable parameter

$$\in \mathbb{R}^{d \times D} \qquad \Big| \qquad \in \mathbb{R}^{D}$$

Linearity

Example (linear regression with a (varying) feature map). $\phi : \mathbb{R}^d \to \mathbb{R}^D$

$$f(\mathbf{x}; \mathbf{w}, b) = \mathbf{w}^\top \phi(\mathbf{x}) + b$$

$$\phi(\mathbf{x}) = W^{(1)\top} x + b^{(1)}$$

Still linear

w in disguise

$$w^{(2)\top}\left(W^{(1)\top} x + b^{(1)}\right) + b^{(2)}$$

$$= \left(W^{(1)} w^{(2)}\right)^\top x + w^{(2)\top} b^{(1)} + b^{(2)}$$

Linearity

Example (linear regression with a (varying) feature map). $\phi : \mathbb{R}^d \to \mathbb{R}^D$

$$f(\mathbf{x}; \mathbf{w}\quad, b) = \mathbf{w}\quad^\top \phi(\mathbf{x}) + b$$

$$\phi(\mathbf{x}) =$$

Composition of matrix mul
+ bias

= mat mul + bias

Linearity

Mat Mul + bias

2-layer neural network with "linear" activation. $\mathbf{W}^{(1)} \in \mathbb{R}^{d\times n_1}$ ✓

$$f^{(1)}(\mathbf{x};\mathbf{W}^{(1)},\mathbf{b}^{(1)}) = \phi(x; W^{(1)}, b^{(1)})$$

Matmul + bias →

$$f^{(2)}(\mathbf{h};\mathbf{w}^{(2)},b^{(2)}) = w^{(2)T} h + b^{(2)}$$

hidden nodes

$$f\left(\mathbf{x};\mathbf{w}^{(2)},b^{(2)},\mathbf{W}^{(1)},\mathbf{b}^{(1)}\right) := f^{(2)}\left(f^{(1)}(\mathbf{x};\mathbf{W}^{(1)},\mathbf{b}^{(1)});\mathbf{w}^{(2)},b^{(2)}\right)$$

still linear

Note: Is this still linear?

Linearity

2-layer neural network with "linear" activation.

$$f\left(\mathbf{x}; \mathbf{w}^{(2)}, b^{(2)}, \mathbf{W}^{(1)}, \mathbf{b}^{(1)}\right) := \mathbf{w}^{(2)\top}(\mathbf{W}^{(1)\top}\mathbf{x} + \mathbf{b}^{(1)}) + b^{(2)}$$

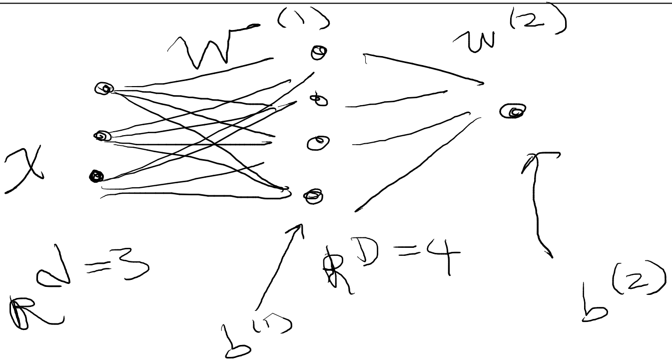
Note: Is this still linear?

Linearity

2-layer neural network with "linear" activation.

$$f\left(\mathbf{x}; \mathbf{w}^{(2)}, b^{(2)}, \mathbf{W}^{(1)}, \mathbf{b}^{(1)}\right) := \mathbf{w}^{(2)\top}(\mathbf{W}^{(1)\top}\mathbf{x} + \mathbf{b}^{(1)}) + b^{(2)}$$

Note: Draw the network architecture diagram



Linearity

mat mul g(matmul + bias) + bias

2-layer neural network with "non-linear" activation $g : \mathbb{R} \to \mathbb{R}$.

$$f^{(1)}(\mathbf{x}; \mathbf{W}^{(1)}, \mathbf{b}^{(1)}) = g\left(\mathbf{W}^{(1)\top}\mathbf{x} + \mathbf{b}^{(1)}\right)$$

$$f^{(2)}(\mathbf{h}; \mathbf{w}^{(2)}, b^{(2)}) = \mathbf{w}^{(2)\top}\mathbf{x} + b^{(2)}$$

$$f\left(\mathbf{x}; \mathbf{w}^{(2)}, b^{(2)}, \mathbf{W}^{(1)}, \mathbf{b}^{(1)}\right) := f^{(2)}\left(f^{(1)}(\mathbf{x}; \mathbf{W}^{(1)}, \mathbf{b}^{(1)})\,;\, \mathbf{w}^{(2)}, b^{(2)}\right)$$

Activation function

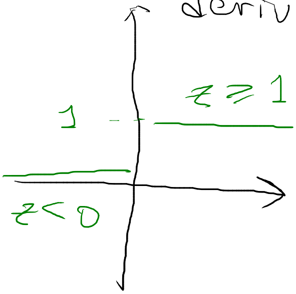
$$\mathrm{relu}'(z) = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases}$$

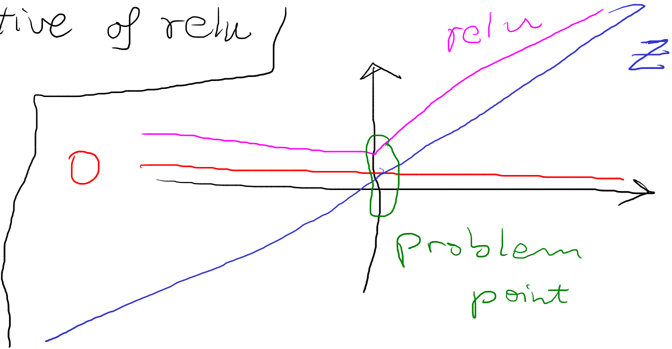
Rectified linear unit or "relu" $\quad g(z): \mathbb{R} \to \mathbb{R}$

$$\mathrm{relu}(z) := \max\{0, z\}$$

Note: Plot "relu" and its derivative

derivative of relu





Linearity

2-layer neural network with "non-linear" activation $g : \mathbb{R} \to \mathbb{R}$.

$$f\left(\mathbf{x}; \mathbf{w}^{(2)}, b^{(2)}, \mathbf{W}^{(1)}, \mathbf{b}^{(1)}\right) := \mathbf{w}^{(2)\top} g(\mathbf{W}^{(1)\top}\mathbf{x} + \mathbf{b}^{(1)}) + b^{(2)}$$

Note: Is this still linear?

No?

6:21 pm

We will see through example

Calculation of the gradient

2-layer NN
with hidden width
n_1

$$f\left(\mathbf{x}; \mathbf{w}^{(2)}, b^{(2)}, \mathbf{W}^{(1)}, \mathbf{b}^{(1)}\right) := \mathbf{w}^{(2)\top} g(\mathbf{W}^{(1)\top}\mathbf{x} + \mathbf{b}^{(1)}) + b^{(2)}$$

- ~~1-dimensional data $d = 1$~~
- ~~$\mathbf{x}^{(i)} = x^{(i)}$ (no bold)~~
- ~~$\mathbf{W}^{(1)} \in \mathbb{R}^{d \times n_1}$~~

$$W^{(1)} = [W_1^{(1)}, \ldots, W_{n_1}^{(1)}] \in \mathbb{R}^{d \times n_1}$$

$$\underbrace{w^{(2)T} g(\underbrace{W^{(1)T}x + b^{(1)}}_{n_1})}_{\text{scalar}} \underbrace{+ b^{(2)}}_{\text{still scalar}}$$

entrywise

Calculation of the gradient

$w^{(2)}, \quad w^{(1)} \in \mathbb{R}^{n_1}$

$$f\left(\mathbf{x}; \mathbf{w}^{(2)}, b^{(2)}, \mathbf{W}^{(1)}, \mathbf{b}^{(1)}\right) := \mathbf{w}^{(2)\top} g(\underline{\mathbf{W}^{(1)\top}\mathbf{x}} + \mathbf{b}^{(1)}) + b^{(2)}$$

lower case $w^{(1)}$ col vec

- 1-dimensional data $d = 1$
- $\mathbf{x}^{(i)} = x^{(i)}$ (no bold) $\quad x$ just a number
- $\mathbf{W}^{(1)} \in \mathbb{R}^{d \times n_1}$

rename

$$W^{(1)} = [w_1^{(1)}, \ldots, w_1^{(n_1)}]$$

$$\begin{bmatrix} w_1^{(1)} \\ \vdots \\ w_1^{(n_1)} \end{bmatrix} x$$

Derivative with respect to a generic point

Note: Calculate the derivative when $d = 1$

MSE

$$J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} J_i(\boldsymbol{\theta})$$

Sum of sample-wise MSE

where

$$J_i(\boldsymbol{\theta}) := (y^{(i)} - \mathbf{w}^{(2)\top} g(\mathbf{w}^{(1)} x^{(i)} + \mathbf{b}^{(1)}) + b^{(2)})^2$$

$$\nabla_\theta J(\theta) = \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta J_i(\theta)$$

$$\begin{bmatrix} w_1^{(1)} \\ \vdots \\ w_{n_1}^{(1)} \end{bmatrix}$$

$$\nabla_\theta J(\theta) = \{\nabla_{w^{(2)}} J(\theta), \nabla_{b^{(2)}} J(\theta), \dots\}$$

Focus on a single i.

Derivative with respect to a generic point

$$J_i(\boldsymbol{\theta}) := (y^{(i)} - \underbrace{\boxed{\mathbf{w}^{(2)\top} g(\mathbf{w}^{(1)} x^{(i)} + \mathbf{b}^{(1)}) + b^{(2)}}}_{=: z^{(i)}})^2 = (y^{(i)} - z^{(i)})^2$$

"model output"

Note: Calculate the derivative

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{w}^{(2)}} = 2(y^{(i)} - z^{(i)})(-1)\frac{\partial z^{(i)}}{\partial w^{(2)}}$$

b.c it's $-z_i$

$$\boxed{\text{Fact}}\quad \boxed{\frac{\partial}{\partial a}(a^\top b) = b}$$

Requires b not dependent on a

Derivative with respect to a generic point

$$J_i(\boldsymbol{\theta}) := (y^{(i)} - \underbrace{\mathbf{w}^{(2)\top} g(\mathbf{w}^{(1)} x^{(i)} + \mathbf{b}^{(1)}) + b^{(2)}}_{z_i})^2$$

Note: Calculate the derivative

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{w}^{(2)}} =$$

$$\frac{\partial z_i}{\partial w^{(2)}} = g(w^{(1)} x^{(i)} + b^{(i)})$$

Derivative with respect to a generic point

$$J_i(\boldsymbol{\theta}) := (y^{(i)} - \mathbf{w}^{(2)\top} g(\mathbf{w}^{(1)} x^{(i)} + \mathbf{b}^{(1)}) + b^{(2)})^2$$

Note: Calculate the derivative

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial b^{(2)}} = -2\left(y^{(i)} - z^{(i)}\right) \underbrace{\frac{\partial z^{(i)}}{\partial b^{(2)}}}_{=1}$$

Derivative with respect to a generic point

$$J_i(\boldsymbol{\theta}) := (y^{(i)} - \overbrace{\mathbf{w}^{(2)\top} g(\mathbf{w}^{(1)} x^{(i)} + \mathbf{b}^{(1)}) + b^{(2)}}^{z^{(i)}})^2$$

Note: Calculate the derivative

Hint: $\odot$ denotes element-wise product between vectors

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{w}^{(1)}} =$$

$$w^{(1)} = \begin{bmatrix} w_1^{(1)} \\ \vdots \\ w_{n_1}^{(1)} \end{bmatrix}$$

$$\frac{\partial}{\partial w_i^{(1)}} J_i(\theta) = -2(y^{(i)} - z^{(i)}) \frac{\partial z^{(i)}}{\partial w_i^{(1)}}$$

Derivative with respect to a generic point

entrywise

$$J_i(\boldsymbol{\theta}) := (y^{(i)} - \mathbf{w}^{(2)\top} g(\mathbf{w}^{(1)} x^{(i)} + \mathbf{b}^{(1)}) + b^{(2)})^2$$

z_i

> **Note:** Calculate the derivative
>
> Hint: $\odot$ denotes element-wise product between vectors

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{w}^{(1)}} =$$

$$\frac{\partial z^{(i)}}{\partial w_1^{(1)}} = \frac{\partial}{\partial w_1^{(1)}} \Big(\underbrace{w_1^{(2)} g(w_1^{(1)} x^{(i)} + b_1^{(1)})}_{\text{only part that depends } w_1^{(1)}} + \cdots + w_{n_1}^{(2)} g(w_{n_1}^{(1)} x^{(i)} + b_{n_1}^{(1)}) + b^{(2)} \Big)$$

Derivative with respect to a generic point

$$J_i(\boldsymbol{\theta}) := (y^{(i)} - \mathbf{w}^{(2)\top} g(\mathbf{w}^{(1)} x^{(i)} + \mathbf{b}^{(1)}) + b^{(2)})^2$$

Note: Calculate the derivative

Hint: $\odot$ denotes element-wise product between vectors

$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{w}^{(1)}} =$

$$\frac{\partial z^{(i)}}{\partial w_1^{(1)}} = \frac{\partial}{\partial w_1^{(1)}} \left(\underbrace{w_1^{(2)} g(w_1^{(1)} x^{(i)} + b_1^{(1)})}_{\text{only part}} + \text{const wrt } w_1^{(1)} \right)$$

$$= w_1^{(2)} g'(w_1^{(1)} x^{(i)} + b_1^{(1)}) x^{(i)}$$

Derivative with respect to a generic point

$$J_i(\boldsymbol{\theta}) := (y^{(i)} - \mathbf{w}^{(2)\top} g(\mathbf{w}^{(1)} x^{(i)} + \mathbf{b}^{(1)}) + b^{(2)})^2$$

Note: Calculate the derivative

Hint: $\odot$ denotes element-wise product between vectors

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{w}^{(1)}} = \frac{\partial z^{(i)}}{\partial w_j^{(1)}} = \frac{\partial}{\partial w_j^{(1)}} \left(w_j^{(2)} g(w_j^{(1)} x^{(i)} + b_j^{(1)}) \right)$$

only part

$$- w_j^{(2)} g'(w_j^{(1)} x^{(i)} + b_j^{(1)}) x^{(i)}$$

Derivative with respect to a generic point

$$J_i(\boldsymbol{\theta}) := (y^{(i)} - \mathbf{w}^{(2)\top} g(\mathbf{w}^{(1)} x^{(i)} + \mathbf{b}^{(1)}) + b^{(2)})^2$$

Note: Calculate the derivative

Hint: $\odot$ denotes element-wise product between vectors

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{w}^{(1)}} =$$

$$\frac{\partial}{\partial w^{(1)}} J_i(\theta) = w^{(2)} \odot \underbrace{g'(\underset{n_1}{w^{(1)}} x^{(i)} + b^{(1)})}_{n_1} x^{(i)}$$

$$\dim(g) = n_1$$

Sanity check

$$w^{(1)} + \frac{\partial J_i(\theta)}{\partial w^{(1)}}$$

Derivative with respect to a generic point

$$J_i(\boldsymbol{\theta}) := (y^{(i)} - \mathbf{w}^{(2)\top} g(\mathbf{w}^{(1)} x^{(i)} + \mathbf{b}^{(1)}) + b^{(2)})^2$$

Note: Calculate the derivative

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{b}^{(1)}} =$$

$$-2\left(y^{(i)} - z^{(i)}\right) \underbrace{\frac{\partial z_i}{\partial b^{(1)}}}_{w^{(2)} \odot g'\left(w^{(1)} x^{(i)} + b^{(1)}\right)}$$

1-layer neural network

$$J_i(\boldsymbol{\theta}) := (y^{(i)} - (y^{(i)} - z^{(i)}))^2 \quad \text{where} \quad z_i = \mathbf{w}^{(2)\top} g(\mathbf{w}^{(1)} x^{(i)} + \mathbf{b}^{(1)}) + b^{(2)}$$

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{w}^{(2)}} = -2(y^{(i)} - z^{(i)}) g(\mathbf{w}^{(1)} x^{(i)} + \mathbf{b}^{(1)})$$

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial b^{(2)}} = -2(y^{(i)} - z^{(i)})$$

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{w}^{(1)}} = -2(y^{(i)} - z^{(i)})(\mathbf{w}^{(2)} \odot g'(\mathbf{w}^{(1)} x^{(i)} + \mathbf{b}^{(1)})) x^{(i)}$$

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{b}^{(1)}} = -2(y^{(i)} - z^{(i)})(\mathbf{w}^{(2)} \odot g'(\mathbf{w}^{(1)} x^{(i)} + \mathbf{b}^{(1)}))$$

1-layer neural network

Given params
calculate grad

1

$$J_i(\boldsymbol{\theta}) := (y^{(i)} - (y^{(i)} - z^{(i)}))^2 \quad \text{where} \quad z_i = \mathbf{w}^{(2)\top} g(\mathbf{h}^{(i)}) + b^{(2)} \quad \text{and} \quad \mathbf{h}^{(i)} = \mathbf{w}^{(1)} x^{(i)} + \mathbf{b}^{(1)}$$

$x \to h \to z$

2

3

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{w}^{(2)}} = -2(y^{(i)} - z^{(i)}) g(\mathbf{h}^{(i)})$$

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial b^{(2)}} = -2(y^{(i)} - z^{(i)})$$

4

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{w}^{(1)}} = -2(y^{(i)} - z^{(i)})(\mathbf{w}^{(2)} \odot g'(\mathbf{h}^{(i)})) x^{(i)}$$

$$\frac{\partial J_i(\boldsymbol{\theta})}{\partial \mathbf{b}^{(1)}} = -2(y^{(i)} - z^{(i)})(\mathbf{w}^{(2)} \odot g'(\mathbf{h}^{(i)}))$$

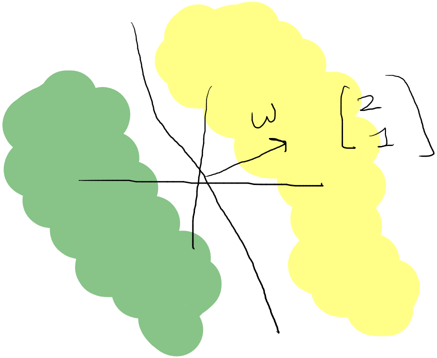
$$\nabla J = \frac{\sum_{i=1}^{N} \nabla J_i}{N}$$

Note: Do: `lec03-in-class-ex2-relu-net.ipynb`

Binary linear classifier

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

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



Perceptron

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

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

Perceptron update

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

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

Output: $\mathbf{w}$

Perceptron

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

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

Perceptron update

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

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

Output: $\mathbf{w}$

Perceptron

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

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

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 $\mathbf{w} \leftarrow \mathbf{w}$,

Output: $\mathbf{w}$

Perceptron

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

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

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 $\mathbf{w} \leftarrow \mathbf{w} \underline{+ y^{(t)} \mathbf{x}^{(t)}}$.

Output: $\mathbf{w}$

Perceptron: an example

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

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

Perceptron update

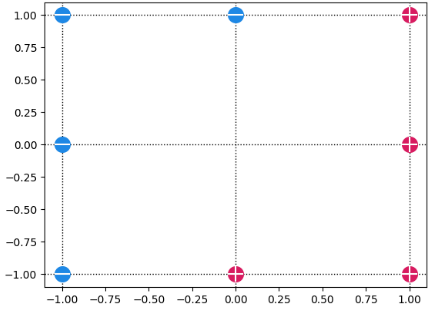
Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots$, time T

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: is this SGD?

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

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

$L(f(x;\theta), y^{(i)})$

$= \max(0, -\underbrace{y^{(i)} f(x;\theta)}_{z^{(i)}})$

$\boxed{\text{choice}}$

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: $\mathbf{w}$

$w - g^{(t)}$ $y^{(t)}$ $x^{(t)}$

- Compute gradient

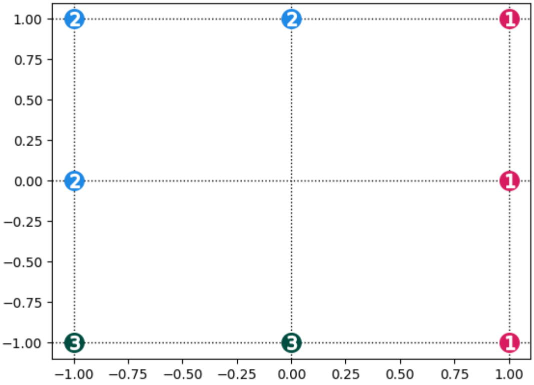
$$\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta}} \frac{1}{m} \sum_{i=1}^{m} L(f(\mathbf{x}^{(i)}, \boldsymbol{\theta}), y^{(i)})$$

- 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

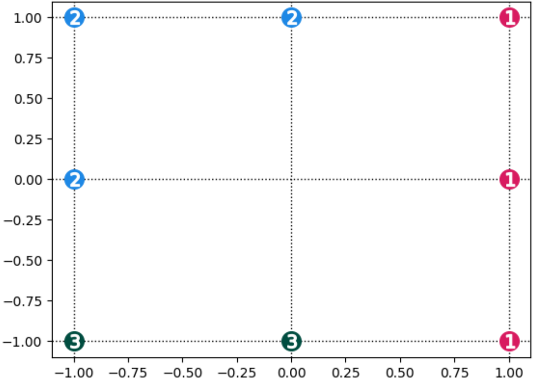
$$f(\mathbf{x}; \mathbf{W}) := \operatorname{argmax}_{\hat{y}=1,\ldots,K} \quad \mathbf{w}_{\hat{y}}^\top \mathbf{x} \quad \in \{1, \ldots, K\} \tag{2}$$



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

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



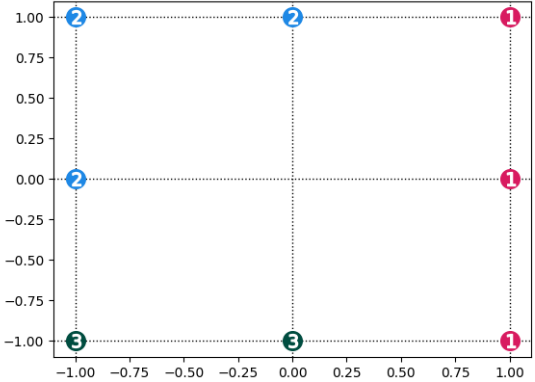
Multiclass perceptron

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: $\mathbf{W}$



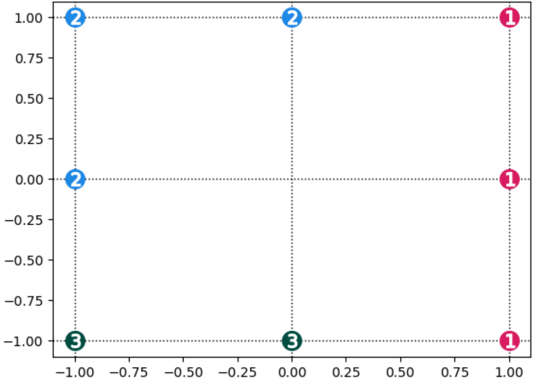
Multiclass perceptron

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

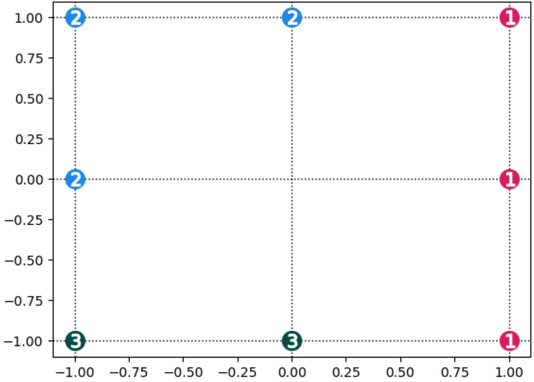
Perceptron update

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

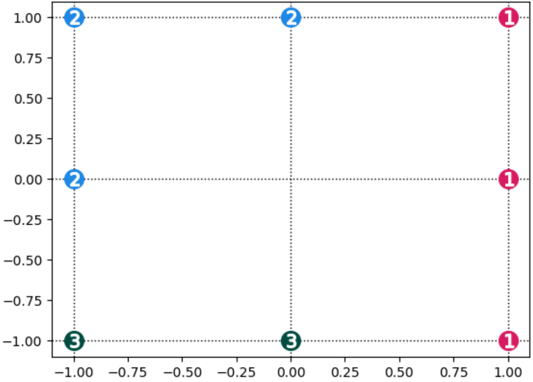
Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots$, time T

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 $\hat{y}^{(t)} \leftarrow \operatorname{argmax}_{\hat{y}} \mathbf{w}_{\hat{y}}^{\top} \mathbf{x}^{(t)}$

   2.2 If $\hat{y}^{(t)} = y^{(t)}$, then $\mathbf{W} \leftarrow \mathbf{W}$,

Output: $\mathbf{W}$



Multiclass perceptron

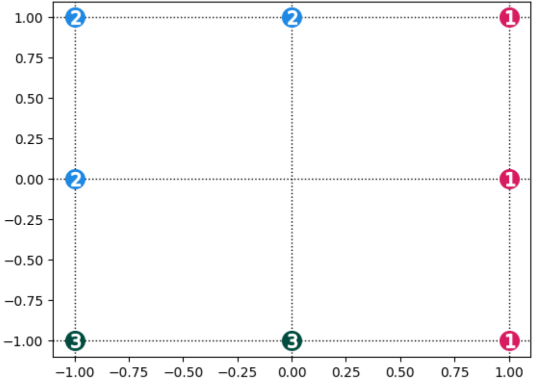
Perceptron update

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)

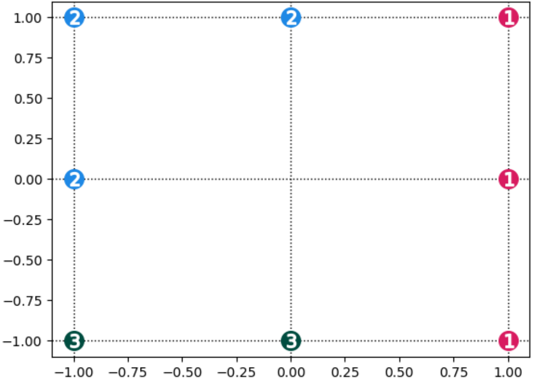
Perceptron update

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)

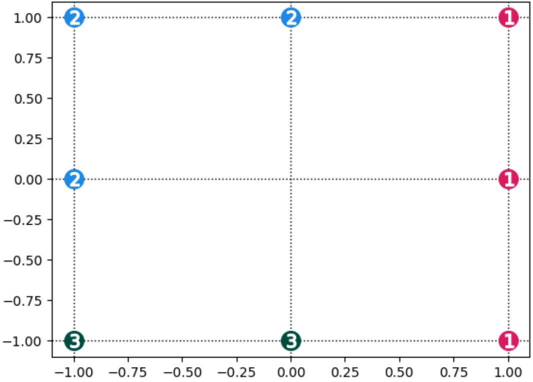
Perceptron update

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

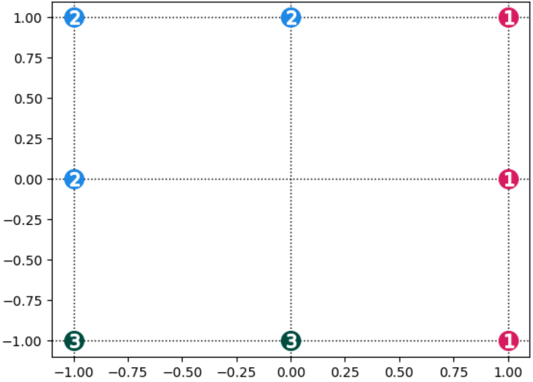
Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots,$ time T

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 $t = 1, 2, \ldots, T$

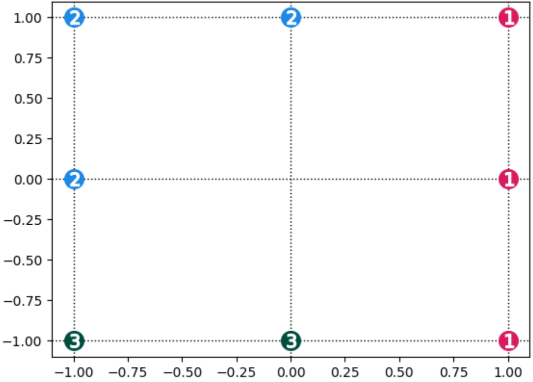
   2.1 $\hat{y}^{(t)} \leftarrow \operatorname{argmax}_{\hat{y}} \mathbf{w}_{\hat{y}}^\top \mathbf{x}^{(t)}$

   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)

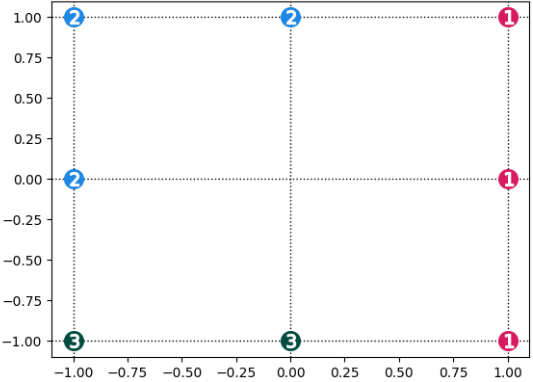
Perceptron update

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

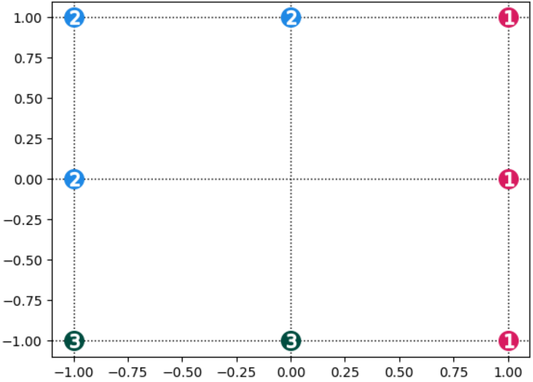
Perceptron update

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)

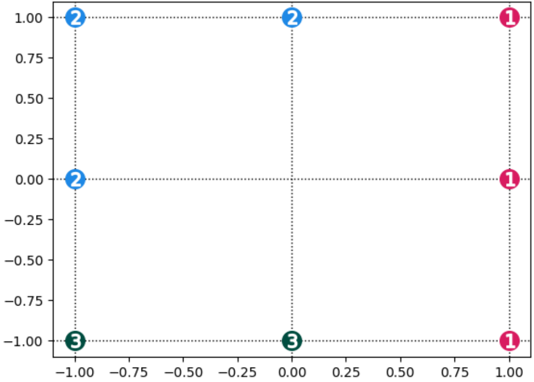
Perceptron update

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)

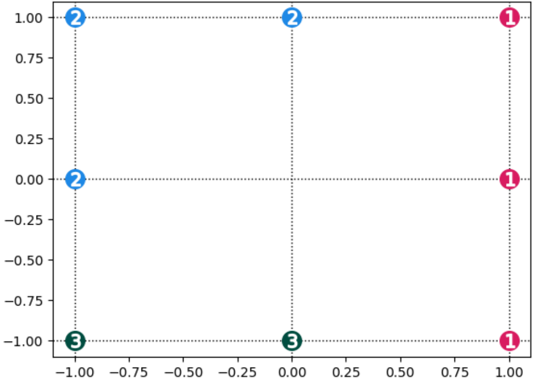
Perceptron update

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

.

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