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 $\mathbf{x}^{(i)} \in \mathcal{X} \subseteq \mathbb{R}^d$
- 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

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

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

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

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

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

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?

Beyong linearity: feature 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). $\phi : \mathbb{R}^1 \to \mathbb{R}^D$

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

Beyong linearity: feature map

```python
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)
```

Beyong linearity: feature 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). $\phi : \mathbb{R}^d \to \mathbb{R}^D$

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

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

Linearity

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

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

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

Linearity

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

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

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

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

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

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

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

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

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

Activation function

Rectified linear unit or "relu"

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

Note: Plot "relu" and its derivative

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?

Calculation of the gradient

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

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

Derivative with respect to a generic point

Note: Calculate the derivative when $d = 1$

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

where

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Binary linear classifier

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

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

Perceptron

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

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

Perceptron update

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

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

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

Output: $\mathbf{w}$

Perceptron: an example

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

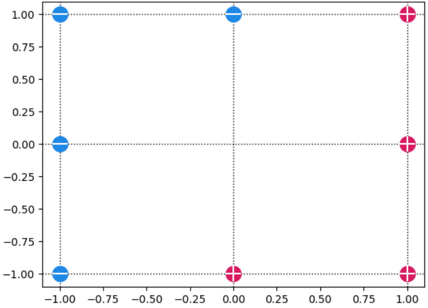
$$f(\mathbf{x}; \mathbf{w}) := \operatorname{sign}(\mathbf{w}^\top \mathbf{x}) \in \{\pm 1\}$$

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

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

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

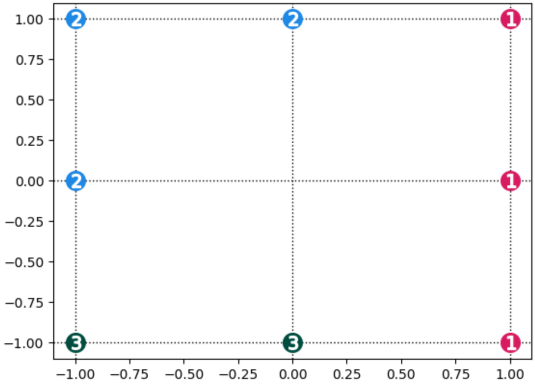
$$\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta}} \frac{1}{m} \sum_{i=1}^{m} L(f(\mathbf{x}^{(i)}, \boldsymbol{\theta}), y^{(i)})$$

- 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

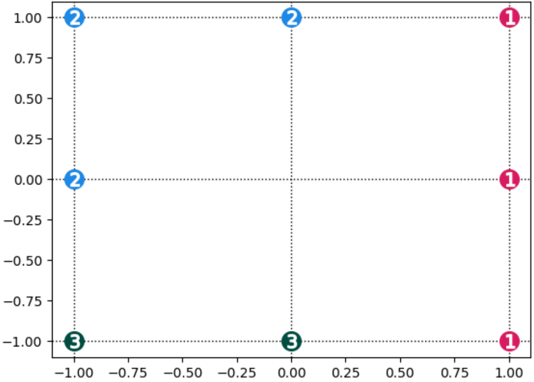
$$f(\mathbf{x}; \mathbf{W}) := \operatorname{argmax}_{\hat{y}=1,\ldots,K} \quad \mathbf{w}_{\hat{y}}^{\top} \mathbf{x} \quad \in \{1, \ldots, K\} \tag{2}$$



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

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



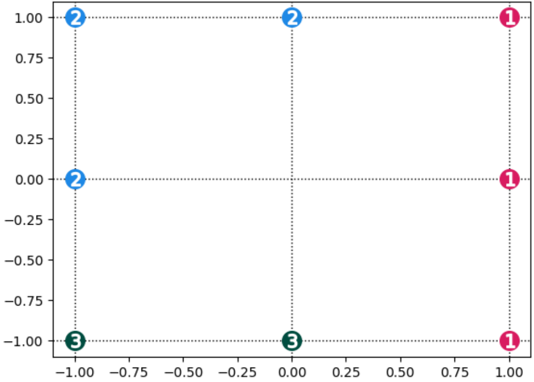
Multiclass perceptron

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: $\mathbf{W}$



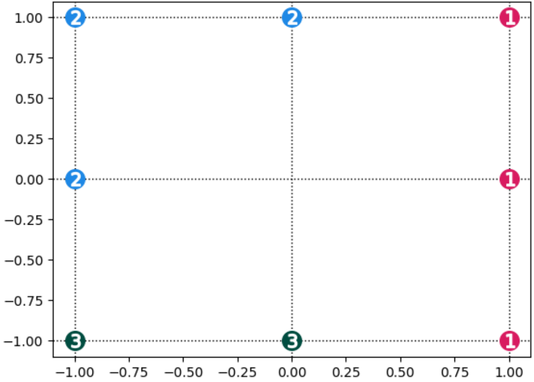
Multiclass perceptron

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: $\mathbf{W}$



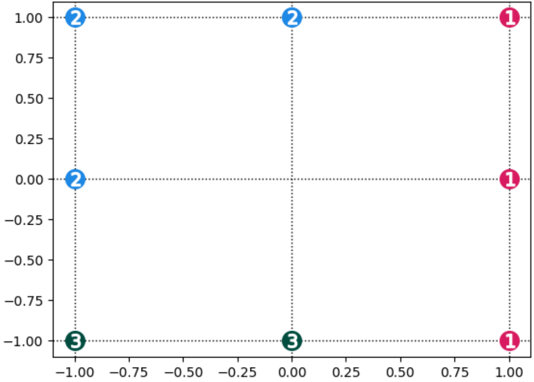
Multiclass perceptron

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

Output: $\mathbf{W}$



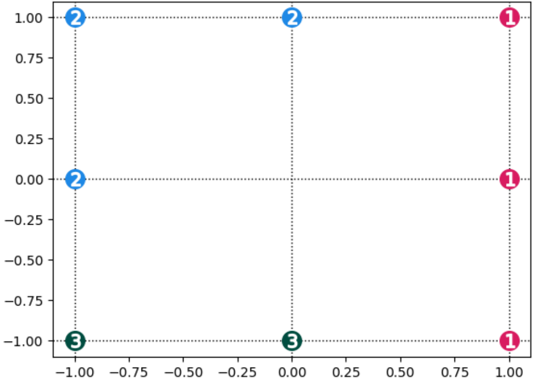
Multiclass perceptron

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

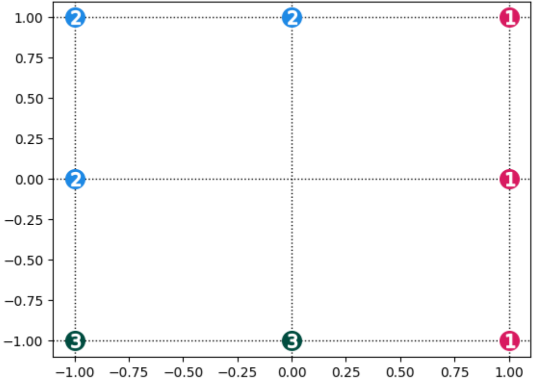
Perceptron update

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

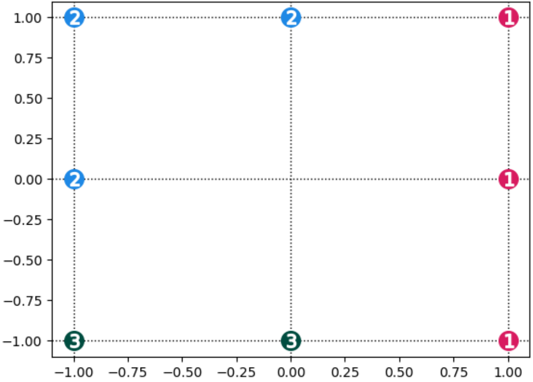
Perceptron update

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)

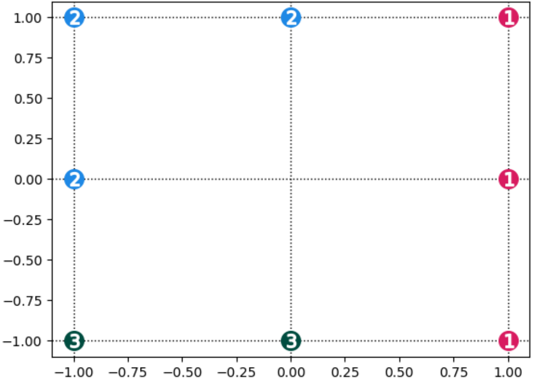
Perceptron update

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

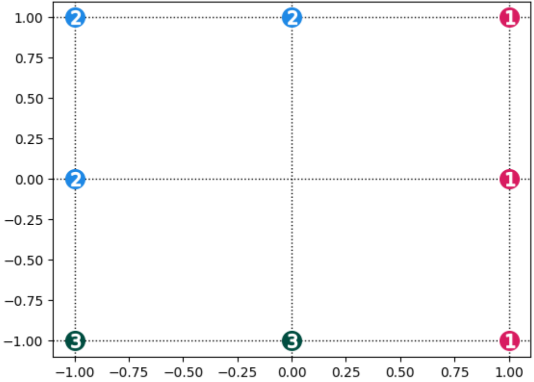
Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots,$ time T

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)

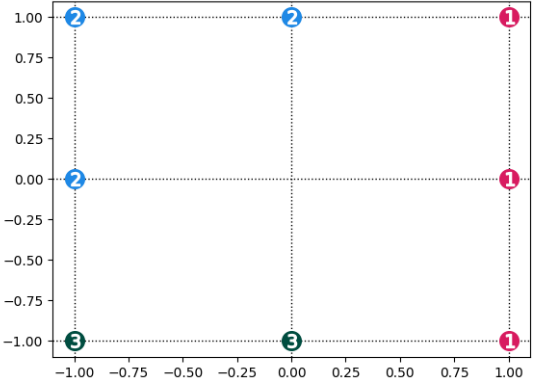
Perceptron update

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

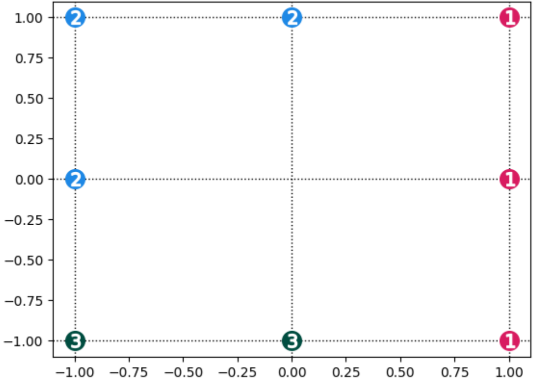
   2.1 $\hat{y}^{(t)} \leftarrow \mathrm{argmax}_{\hat{y}} \mathbf{w}_{\hat{y}}^\top \mathbf{x}^{(t)}$

   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)

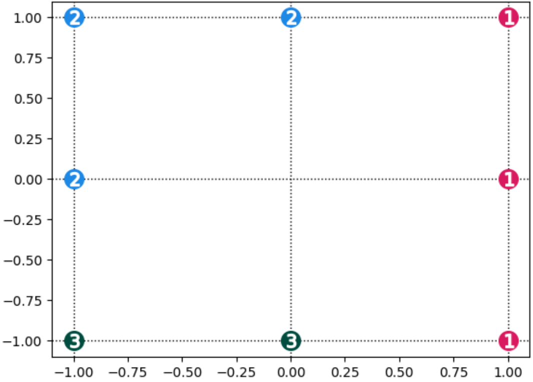
Perceptron update

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

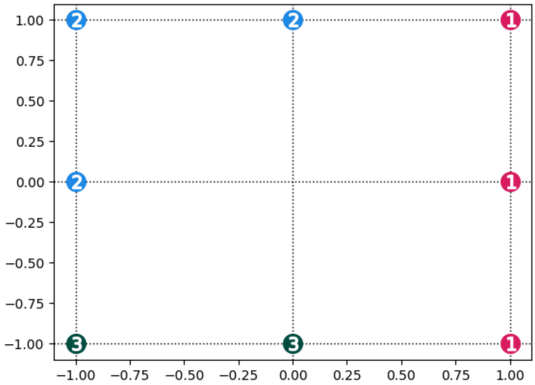
   2.1 $\hat{y}^{(t)} \leftarrow \operatorname{argmax}_{\hat{y}} \mathbf{w}_{\hat{y}}^\top \mathbf{x}^{(t)}$

   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)

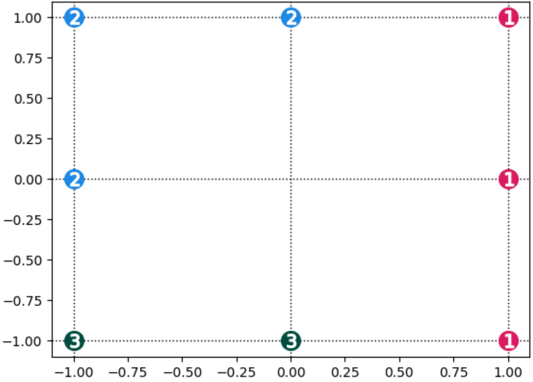
Perceptron update

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



References I

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