✨

ch6: Logistic Regression

(Linear) Discriminant Functions

Discriminant Functions（判别函数）（P9）

- Function that characterize the degree of data belonging to a class, parameterized with a set of parameters θ_i.

$$g_i(x \mid \theta_i)$$

- Model the decision boundaries between classes **directly** and **simultaneously**.

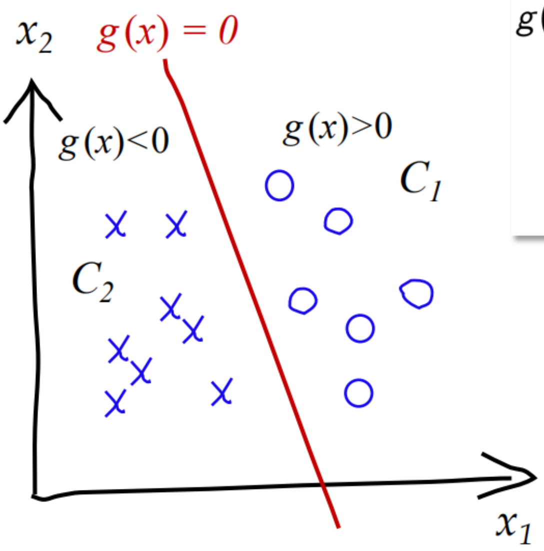
> Determinizing class boundaries (discriminants) is usually easier than estimating the class densities.

- consider a classifier with c classes
- define c discriminant functions $g_i(x)$
- **decision rule**: assign x to C_j if $g_j(x) > g_i(x)$, $\forall i \neq j$

> **Example – Bayes Decision(P10)**

1. Linear Discriminant Functions(P11-12)

- A linear discriminant function models a **linear decision boundary** of two classes.



$$
\begin{aligned}
g(\mathbf{x}) &= g_1(\mathbf{x}) - g_2(\mathbf{x}) \\
&= (\mathbf{w}_1^T \mathbf{x} + w_{10}) - (\mathbf{w}_2^T \mathbf{x} + w_{20}) \\
&= (\mathbf{w}_1 - \mathbf{w}_2)^T \mathbf{x} + (w_{10} - w_{20}) \\
&= \mathbf{w}^T \mathbf{x} + w_0
\end{aligned}
$$

- Decision rule:

$$
\text{Choose } \begin{cases} C_1 & \text{if } g(\mathbf{x}) > 0 \\ C_2 & \text{otherwise} \end{cases}
$$

Advantages:

- simplicity: $O(d)$ time and space complexity; 可解释性：最终的输出是属性的加权和
- understandability: final output is a weighted sum of attributes;
- accuracy: quite accurate if some assumptions are satisfied.

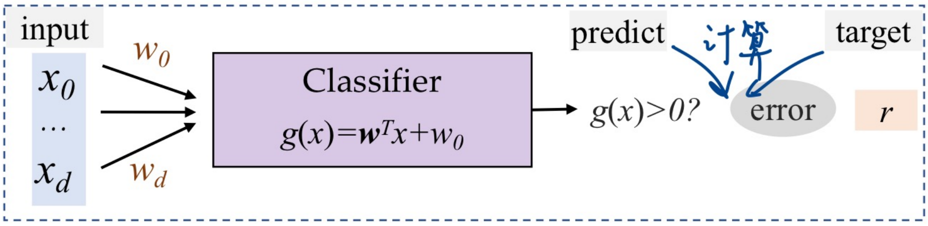
Geometry Interpretation: Hyperplane(P13)

The linear function $g(x)$ defifines a **hyperplane** that divides the input space into 2 half-spaces

Perceptron（感知机）

Definition(P16)

- The first, naïve linear classifier. (to introduce in future lectures)



- Train:

  根据误差评估出系数w

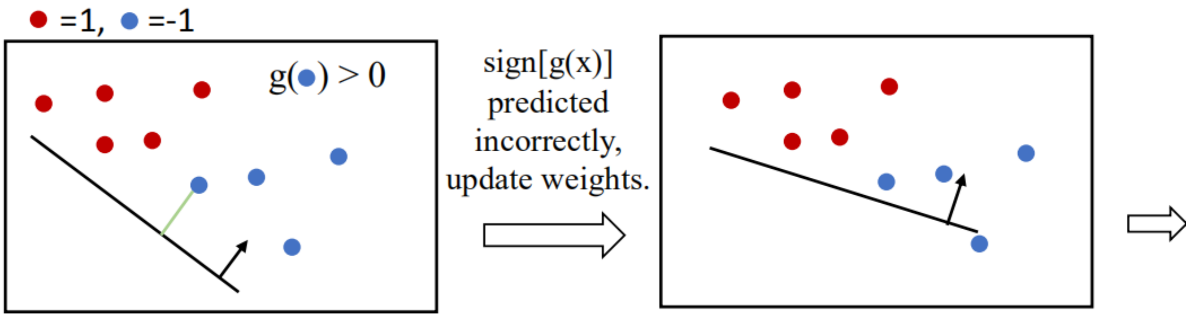
  - estimate the parameters $\boldsymbol{w}$ and w_0 from data
- Predict:
  - calculate $g(x) = \boldsymbol{w}^{\mathrm{T}}x+w_0$ and choose C_1 if $g(x) > 0$ or choose C_2 if $g(x)<0$.

perceptron classifies data based on which **side** of the plane the new point lies on.

Training(P17)

Let $\boldsymbol{x}$ denote the data input, and $r \in \{1,-1\}$ (means the sign of $g(x)$) denote the **label** of target classes.

Input: dataset $D = \{(x^{(1)}, r^{(1)}), (x^{(2)}, r^{(2)}), \ldots (x^{(N)}, r^{(N)})\}$
for each training instance $(x^{(\ell)}, r^{(\ell)}) \in D$:
if $r^{(\ell)} g_i(x^{(\ell)}) \leq 0$: // a misclassification occurs
$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta\, r^{(\ell)} x^{(\ell)}$ // move the hyperplane (defined by $\boldsymbol{w}$) towards the misclassified data point
repeat until the entire training set is classified correctly



Limitations of Perceptron(P18)

Hard Decision and Optimization:只有0-1二值，难以优化

Logistic Regression

Linear Classifification with Uncertainty（P19)

- What is the posterior probability of choosing C1 (or C2)?

Let

$$P(C_1 \mid \boldsymbol{x}) = y \qquad P(C_2 \mid \boldsymbol{x}) = 1 - y$$

Classification rule:

$$\text{Choose} \begin{cases} C_1 & \text{if } y > 0.5 \\ C_2 & \text{otherwise} \end{cases}$$

Equivalent Rule:

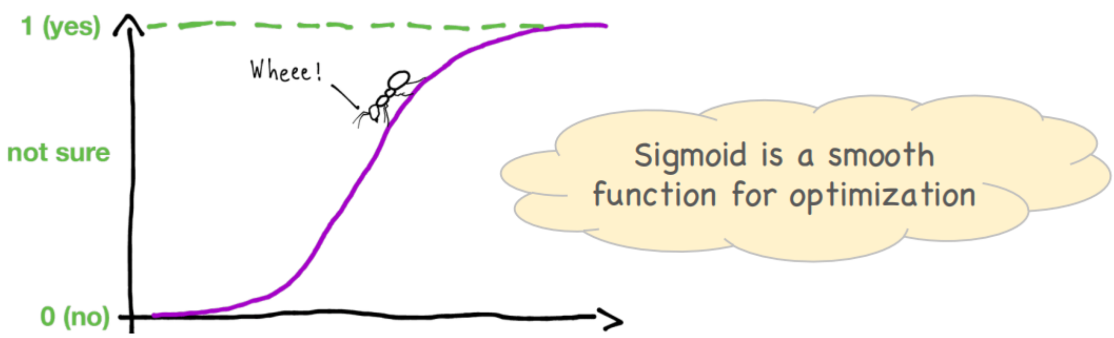
$$\underbrace{\frac{y}{1-y}}_{\text{odds of y}} > 1 \quad or \quad \log \underbrace{\frac{y}{1-y}}_{\text{Log odds (logit function) of y}} > 0$$

The Sigmoid Function(P21)

$$\boxed{\log \frac{y}{1-y} = \boldsymbol{w}^{\mathrm{T}} x + w_0}$$

$$\Rightarrow \qquad y = \text{sigmoid}(\mathbf{w}^T \mathbf{x} + w_0) = \frac{1}{1 + \exp[-(\mathbf{w}^T \mathbf{x} + w_0)]}$$

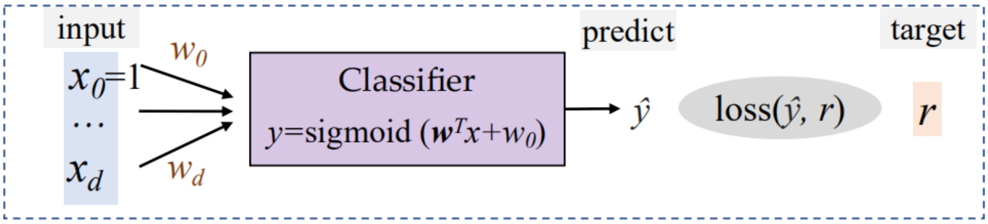
- The sigmoid function (or logistic function) is the **inverse function** of logit, which directly computes the **posterior** class probability $P(C_1|x)$.



Logistic Regression(P22)

- A classifier that estimates the decision boundary as a logistic function:

$$y = \text{sigmoid}\,(\boldsymbol{w}^{\mathrm{T}}x + w_0) = \frac{1}{1+\exp[-(\boldsymbol{w}^{T}x+w_0)]}$$



- Train:
  - estimate the parameters $\boldsymbol{w}$ and w_0 from data
- Test:
  - calculate y = sigmoid ($\boldsymbol{w}^{\mathrm{T}}x+w_0$) and choose C_1 if $y > 0.5$ (y can be interpreted as a posterior probability).

Loss Function(P23)

- For a given input x, the model outputs a probability y of $x \in C_1$. Let $r \in \{0, 1\}$ be the label of the real class (r =1: $x \in C_1$, $r = 0$: $x \in C_2$):
  - if $r = 1$: we aim to maximize $\log p(C_1|x) = \log y$, cost is $-\log y$
  - if $r = 0$: we aim to maximize $\log p(C_2|x) = \log (1-y)$, cost is $-\log (1-y)$
- Can writhe this succinctly as a cross-entropy loss:

$$\ell\,(\boldsymbol{w}, w_0\,|\,x, r) = -\underbrace{r \log y}_{\text{nonzero only if r=1}} - \underbrace{(1-r) \log (1-y)}_{\text{nonzero only if r=0}}$$

$$\mathrm{CE}(\mathrm{p}, \mathrm{q}) = \mathbb{E}_{x\sim p(x)}\left\{\log \frac{1}{q(x)}\right\} = -\sum_{x\in X} p(x)\log_2 q(x)$$

- Equivalent to maximize the likelihood:

$$r\,|\,x \sim \text{Bernoulli}(y)$$

$$p(r|x) = y^{r}\,(1\text{-}y)^{(1\text{-}r)} = \begin{cases} y & \text{if } r{=}1 \\ 1-y & \text{if } r{=}0 \end{cases}$$

Training(P24)

Given: $D = \{(x^{(1)}, r^{(1)}), \ldots, (x^{(\mathrm{N})}, r^{(\mathrm{N})})\}$

minimize the loss function using gradient descend:

- Goal:

$$\min_w L(\boldsymbol{w})$$

- Iteration:

$$\boldsymbol{w}_{\mathrm{t+1}} = \boldsymbol{w}_{\mathrm{t}} - \eta_{\mathrm{t}} \frac{\partial L}{\partial w}$$

Optimization – Gradient Descend(P25-27)

> 比较复杂的数学推导，回头只需要掌握结论

$$\ell\,(\boldsymbol{w}, w_0 \,|\, x, r) = - r \log y - (1-r) \log (1-y)$$

$$L(\boldsymbol{w}, w_0 \,|\, D) = - \textstyle\sum_{\ell=1}^{N} r^{(\ell)} \log y^{(\ell)} + (1-r^{(\ell)}) \log (1-y^{(\ell)})$$

What is $\frac{\partial L}{\partial w}$?

Hint: if y=sigmoid(a) = 1/[1+exp(-a)], its derivative is $\frac{dy}{da} = y(1-y)$

For each $\boldsymbol{w}_{\mathrm{j}}$ (j =1,…,d):

$$\frac{\partial L}{\partial w_j} = -\sum_{\ell} \underbrace{\left(\frac{\partial L}{\partial y^{(\ell)}} \frac{\partial y^{(\ell)}}{\partial a^{(\ell)}} \frac{\partial a^{(\ell)}}{\partial w_j} \right)}_{\text{Chain rule}} = -\sum_{\ell} \left(\frac{r^{(\ell)}}{y^{(\ell)}} - \frac{1-r^{(\ell)}}{1-y^{(\ell)}} \right) \frac{\partial y^{(\ell)}}{\partial a^{(\ell)}} \frac{\partial a^{(\ell)}}{\partial w_j}$$

For each $\boldsymbol{w}_{\mathrm{j}}$ (j =1,…,d):

$$\frac{\partial L}{\partial w_j} = -\sum_{\ell} \left(\frac{r^{(\ell)}}{y^{(\ell)}} - \frac{1-r^{(\ell)}}{1-y^{(\ell)}} \right) \frac{\partial y^{(\ell)}}{\partial a^{(\ell)}} \frac{\partial a^{(\ell)}}{\partial w_j}$$

Since $a^{(l)} = \boldsymbol{w}^{\mathrm{T}} x^{(l)} + w_0$, we have $\frac{\partial a^{(\ell)}}{\partial w_j} = x_j^{(\ell)}$

So,

$$\frac{\partial L}{\partial w_j} = -\sum_{\ell} \left(\frac{r^{(\ell)}}{y^{(\ell)}} - \frac{1-r^{(\ell)}}{1-y^{(\ell)}} \right) y^{(\ell)} \left(1 - y^{(\ell)}\right) x_j^{(\ell)} = -\sum_{\ell} (r^{(\ell)} - y^{(\ell)}) x_j^{(\ell)}$$

- An interesting point

$$\frac{\partial L}{\partial w_j} = -\sum_l \underbrace{(r^{(l)} - y^{(l)})}_{\text{error}} \underbrace{x_j^{(l)}}_{\text{input}}$$

The update to each weight is the product of error and input (signal)

Algorithm（P28）（没细看，感觉应该不会考）

Gradient Descend for LR

Input: $D = \{(\mathrm{x}^{(l)}, \mathrm{r}^{(l)})\}\ (l = 1{:}N)$
for $j = 0, \ldots, d$
  $w_j \leftarrow rand\,(-0.01, 0.01)$
repeat
  for $j = 0, \ldots, d$
    $\Delta w_j \leftarrow 0$
  for $l = 1, \ldots, N$
    $a \leftarrow 0$
    for $j = 0, \ldots, d$
      $a \leftarrow a + w_j x_j^{(l)}$
    $y \leftarrow \mathrm{sigmoid}\,(a)$
    $\Delta w_j \leftarrow \Delta w_j + (r^{(l)} - y) x_j^{(l)}$
  for $j = 0, \ldots, d$
    $w_j \leftarrow w_j + \eta \Delta w_j$
until convergence

Multiple Classes

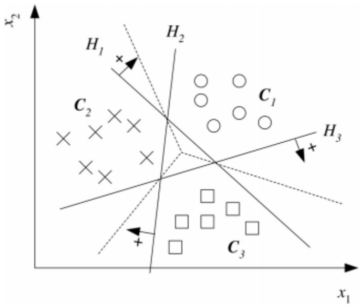
- **Linear Classififier for Multiclass（P30）**

- K discriminant functions:

$$g_i(\mathbf{x} \mid \mathbf{w}_i, w_{i0}) = \mathbf{w}_i^T \mathbf{x} + w_{i0}$$

- Linearly separable classes:

$$g_i(\mathbf{x} \mid \mathbf{w}_i, \mathbf{w}_{i0}) = \begin{cases} > 0 & \text{if } \mathbf{x} \in C_i \\ \leq 0 & \text{otherwise} \end{cases}$$



For each class C_i, there exists a hyperplane H_i such that all $\mathbf{x} \in C_i$ lie on the positive side and all other $\mathbf{x} \in C_j, j \neq i$ lie on the negative side.

- decision rule for any test case x:

$$\boxed{\text{Choose } C_i \text{ if } g_i(\mathbf{x}) = \max_{j=1}^{K} g_j(\mathbf{x})}$$

- geometrically a linear classifier partitions the feature space into K convex decision regions $\mathcal{R}_i$.

What is the posterior probability of choosing C_i (i=1,...,K)?

- One of the K classes, e.g., C_K, is taken as the reference class.
- Assume that

$$\log \frac{p(\mathbf{x}|C_i)}{p(\mathbf{x} \mid C_K)} = \mathbf{w}_i^T \mathbf{x} + w_{i0}^0, \quad i = 1, \ldots, K-1$$

So we have

$$\begin{aligned} \frac{P(C_i \mid \mathbf{x})}{P(C_K \mid \mathbf{x})} &= \frac{p(\mathbf{x} \mid C_i)P(C_i)}{p(\mathbf{x} \mid C_K)P(C_K)} \\ &= \exp\left(\mathbf{w}_i^T \mathbf{x} + w_{i0}^0\right) \cdot \exp\left(\log \frac{P(C_i)}{P(C_K)}\right) \\ &= \exp\left(\mathbf{w}_i^T \mathbf{x} + w_{i0}\right) \qquad (1) \end{aligned}$$

where $w_{i0} = w_{i0}^0 + \log[p(C_i)/P(C_K)]$.

Softmax Regression

Softmax Function（P33-34）

If we want to treat all classes uniformly without having to choose a reference class, we can use the softmax function instead for the posterior class probabilities:

$$\boxed{y_i = \hat{P}(C_i \mid \mathbf{x}) = \frac{\exp\left(\mathbf{w}_i^T \mathbf{x} + w_{i0}\right)}{\sum_{j=1}^{K} \exp\left(\mathbf{w}_j^T \mathbf{x} + w_{j0}\right)}, \quad i = 1, \ldots, K}$$

- In general, the **softmax function** is defined as

$$y_{\mathrm{i}} = \mathrm{softmax}\,(a_1, \ldots, a_{\mathrm{K}})_{\mathrm{i}} = \frac{e^{a_{\mathrm{i}}}}{\sum_{j=1}^{K} e^{a_{\mathrm{j}}}}$$

where the inputs $a_i = \boldsymbol{W}_{\mathrm{i}} x + w_{\mathrm{i}0}$ are called the **logits**. $\boldsymbol{W}_{\mathrm{i}}$ and $w_{\mathrm{i}0}$ are the trainable parameters.

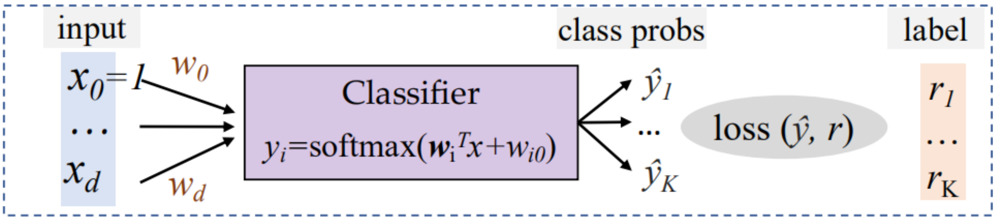
if $a_k \gg a_j, \forall j \neq k$, then $p(\mathcal{C}_k|\mathbf{x}) \approx 1,\ p(\mathcal{C}_j|\mathbf{x}) \approx 0.$

- If one of the a_k's is much larger than the others, softmax $(a_1, \ldots a_K)$ is a smoothed approximation of argmax. (so really it's more like "soft-argmax".)
  - "max" because it amplifies probability of the largest logit a_{i}.
  - "soft" because still assigns some probability to smaller a_{j}.
- The softmax function behaves like taking a maximum, but it has the advantage of being differentiable.

Softmax Regression（P35）

- A classifier that estimates the decision boundaries as **Softmax functions**:

$$y_i = \text{softmax}\,(\boldsymbol{w}_i^T x + w_{i0}) = \frac{\exp\,[\boldsymbol{w}_i^T x + w_{i0}]}{\sum_{j=1}^{K} \exp\,[\boldsymbol{w}_j^T x + w_{j0}]} \quad (i = 1,..,K)$$



- Train:
  - estimate the parameters $\boldsymbol{w}_i$ and w_{i0} (i=1:K) from data
- Test:
  - calculate y_i = softmax ($\boldsymbol{w}_i^T x + w_0$) and choose C_i if $y_i = \max\{y_{1:K}\}$ (y can be interpreted as a posterior probability).

Loss Function（P36）

$$\boldsymbol{r} = \underbrace{(0, \ldots, 0, 1, 0, \ldots, 0)}_{\text{entry } k \text{ is } 1}$$

- For a given input $\boldsymbol{x}$, the model outputs a vector of class probabilities $\boldsymbol{y} = (y_1, \ldots\, y_K)$, and the label of target class is a one-hot vector $\boldsymbol{r} = (r_1, \ldots, r_K)$ (r_i=1:$x \in C_i$, r_i=0: $x \notin C_i$)
  - if $r_1 = 1$: we aim to maximize $\log p(C_1|x) = \log y_1$, cost is $-\log y_1$
  - if $r_2 = 1$: we aim to maximize $\log p(C_2|x) = \log y_2$, cost is $-\log y_2$
  - ……
- We can write this succinctly as a cross-entropy loss function:

$$L_{CE}(\boldsymbol{y}, \boldsymbol{r}) = -\sum_{i=1}^{K} r_i \log y_i = -\,\boldsymbol{r}^T(\log \boldsymbol{y})$$

where the *log* is applied elementwise.

Training & Optimization – Gradient Descend（P37-38）

- Given: $D = \{(x^{(1)}, r^{(1)}), \ldots, (x^{(N)}, r^{(N)})\}$
- minimize the loss function using **gradient descend**:

- Goal:

$$\min_w L(\boldsymbol{w})$$

- Iteration:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \frac{\partial L}{\partial w}$$

$$L(\boldsymbol{w}|\, \boldsymbol{x}, \boldsymbol{r}) = -\Sigma_{i=1}^{K} r_i \log y_i$$

$$L(\boldsymbol{w}|\, D) = -\Sigma_{\ell=1}^{N}[\Sigma_{i=1}^{K} r_i^{(\ell)} \log y_i^{(\ell)}]$$

What is $\frac{\partial L}{\partial w}$?

Hint: if $y_i = \exp(a_i)/\Sigma_j \exp(a_j)$, its derivative is $\frac{\partial y}{\partial a} = y_i(\delta_{ij} - y_j)$ where $\delta_{ij} = \begin{cases} 0 & \text{if } i \neq j, \\ 1 & \text{if } i = j. \end{cases}$

For each w_j and w_{j0} (j=1,…,K), given $\Sigma_i r_i^{(\ell)} = 1$:

$$-\frac{\partial L}{\partial w_j} = \sum_l \sum_i \frac{\partial L}{\partial y_i^{(\ell)}} \frac{\partial y_i^{(\ell)}}{\partial a^{(\ell)}} \frac{\partial a^{(\ell)}}{\partial w_j} = \sum_l \sum_i r_i^{(\ell)} (\delta_{ij} - y_j^{(l)}) \boldsymbol{x}^{(\ell)}$$

$$= \sum_l \left[\sum_i r_i^{(\ell)} \delta_{ij} - y_j^{(\ell)} \sum_i r_i^{(\ell)} \right] x^{(\ell)} = \sum_l (r_j^{(\ell)} - y_j^{(\ell)}) \boldsymbol{x}^{(\ell)}$$

The Algorithm（P39）

Gradient Descend for Softmax Regression

for $i = 1, \ldots, K$, **for** $j = 0,\ldots,d$,
    $w_{ij} \leftarrow$ rand (-0.01, 0.01) // initialization
repeat
    for $i = 1,\ldots,K$, **for** $j = 0,\ldots, d$, $\Delta w_{ij} \leftarrow 0$
    for $l = 1,\ldots,N$
        for $i = 1,\ldots,K$
            $a_i \leftarrow 0$
            for $j = 0,\ldots, d$
                $a_i \leftarrow a_i + w_{ij} x_j^{(l)}$
        for $i = 1,\ldots, K$
            $y_i \leftarrow \exp(a_i)/\boldsymbol{\Sigma}_j \exp(a_j)$
        for $i = 1,\ldots,K$
            for $j = 0,\ldots, d$
                $\Delta w_{ij} \leftarrow \Delta w_{ij} + (r_i^{(l)} - y_i) x_j^{(l)}$
    for $i = 1,\ldots,K$
        for $j = 0,\ldots, d$
            $w_{ij} \leftarrow w_{ij} + \boldsymbol{\eta} \Delta w_{ij}$
until convergence