人工智慧專題

劉瓊如

September 30, 2022

1. Notice from previous class
2. Recommendation by graph
   - Pinterest
3. Machine Learning
   - Perceptron
   - Backpropagation
4. Word 2 Vector

Last time

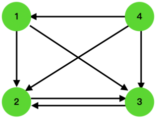


Figure: matrix operation

Let the adjacent matrix $W = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$ and the degree of each nodes: $d_1 = 2,\ d_2 = 1,\ d_3 = 1,\ d_4 = 3$. Let $D = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 3 \end{pmatrix}$. Then

$$A = WD^{-1}$$

左乘

Define

$$W_{ij} = W_{i \leftarrow j} = \begin{cases} 1 & \text{if } 1 \leftarrow j \\ 0 & \text{otherwise} \end{cases}$$

$$W = \begin{pmatrix} 0 & 0 & 0 & 1 \leftarrow 4 \\ 2 \leftarrow 1 & 0 & 2 \leftarrow 3 & 2 \leftarrow 4 \\ 3 \leftarrow 1 & 3 \leftarrow 2 & 0 & 3 \leftarrow 4 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$AR = WD^{-1}R = \begin{pmatrix} 0 & 0 & 0 & \frac{1}{3} \\ \frac{1}{2} & 0 & 1 & \frac{1}{3} \\ \frac{1}{2} & 0 & 0 & \frac{1}{3} \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \end{pmatrix}$$

Row operation and column operation

$$D^{-1} = \begin{pmatrix} \frac{1}{2} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \frac{1}{3} \end{pmatrix}$$

- D^{-1} 對 W 做 column operation $W_c = WD^{-1} = \begin{pmatrix} 0 & 0 & 0 & \frac{1}{3} \\ \frac{1}{2} & 0 & 1 & \frac{1}{3} \\ \frac{1}{2} & 0 & 0 & \frac{1}{3} \\ 0 & 0 & 0 & 0 \end{pmatrix}$
- D^{-1} 對 W 做 row operation $W_r = D^{-1}W = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1/3 & 0 & 1/3 & 1/3 \\ 1/2 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 \end{pmatrix}$

Okapi BM25

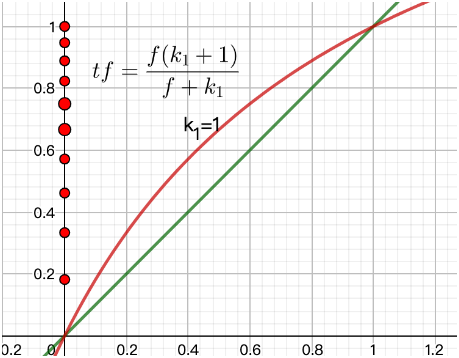
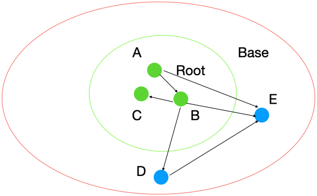
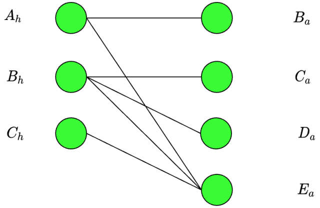


Figure: BM25 $b = 0,\ k_1 = 1$

SALSA 與隨機遊走間的關係

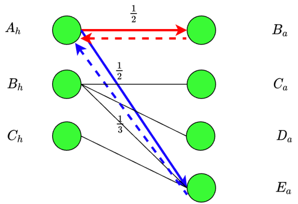




Let the adjacent matrix $W = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

$$\tilde{H} = W_r W_c^T = \begin{pmatrix} 0 & 1/2 & 0 & 1/2 \\ 0 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1/2 & 0 & 1/3 \\ 0 & 1/2 & 1 & 1/3 \\ 0 & 0 & 0 & 1/3 \end{pmatrix}^T$$

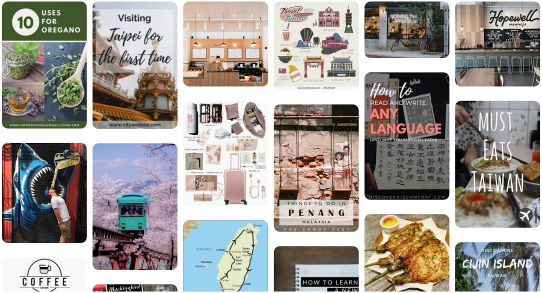
$$= \begin{pmatrix} 4/6 & 1/6 & 1/6 \\ 1/9 & 7/9 & 1/9 \\ 1/3 & 1/3 & 1/3 \end{pmatrix}$$



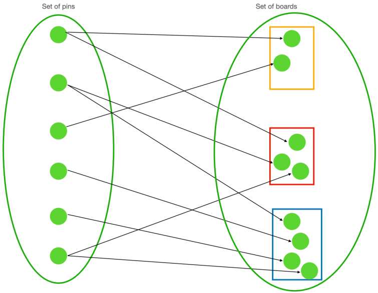
$4/6 = 1/2 \times 1 + 1/2 \times 1/3$

Related Pins at Pinterest

論文：Related Pins at Pinterest: The Evolution of a Real-World Recommender System
整體架構：建立 Pin-Borad Graph、Memboost、Ranking 的流程及算法
效益：Related Pins 對 Pinterest 的推薦增加 40% 使用者選用 pins



- Pinterest 網站入口有許多圖片選釘 (pin)，代表不同主題，在使用者首次登入，就會詢問你的興趣，與這些主題相關的 pin，就會出現在使用者自己的主畫面上．使用者可以將這些 pin 儲存成不同的命名 (topic) 在自己的 board 上。
- 每個 pin 的組成：image, link, description 每個 pin 被存在不同的 board，視為不同的 pin，相同的照片存成不同的 pin 在不同的 board．這樣的對應形成一個 bipartite graph



- 算法是以儲存率為目標：相關推薦 pins 存入自己的 board(or 看過的推薦 pins)
- 模型流程分成：
  - Candidate Generation
  - Ranking
  - Memboost

Candidate Generation

一個 Pin：healthy chocolate strawberry milk shake 可能被不同的使用者以不同命名的 board 儲存：smoothies (奶昔)，strawberry(草莓)，yummm (好吃)

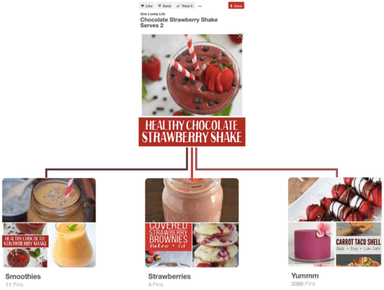


Figure: Pin-Board Graph

Pin-Board Graph

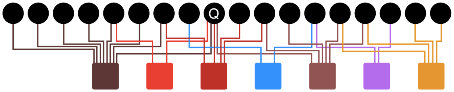


Figure: Pin-Board Graph

- $G = (P, B, E)$
- $P = \{p : \text{pins}\}$
- $B = \{b : \text{boards}\}$
- $E = \{e : \text{link between pins and boards}\}$

Pixie

參考論文：論文：Pixie: A System for Recommending 3+ Billion Items to 200+ Million Users in Real-Time

Algorithm 1 Basic Random Walk; q is the query pin; E denotes the edges of graph G; α determines the length of walks; N is the total number of steps of the walk; V stores pin visit counts.

```
BasicRandomWalk(q: Query pin, E: Set of edges, α: Real, N: Int)
totSteps = 0, V = 0⃗
repeat
    currPin = q
    currSteps = SampleWalkLength(α)
    for i = [1 : currSteps] do
        currBoard = E(currPin)[rand()]
        currPin = E(currBoard)[randNeighbor()]
        V[currPin]++
    totSteps += currSteps
until totSteps ≥ N
return V
```

Figure: Randomwalk

Random walk

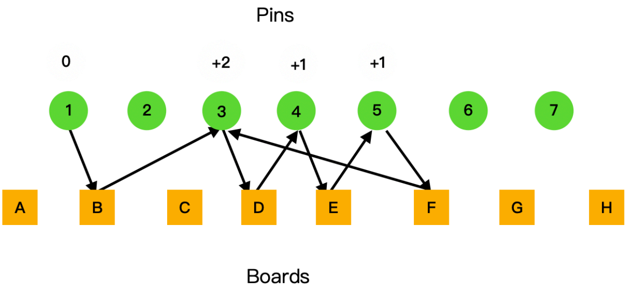
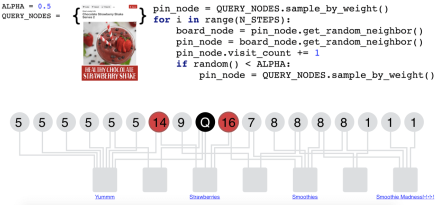


Figure: Randomwalk

Top counts of random walk

Extract the top visited pins：選取 count 數前幾名的 pin 來當成 candidate，例如下圖 count 數 14 與 16 的 pin：



原理是利用 pin 儲存在 board 的 co-occurrence 的特性（因為使用者比較會把同性質的 pin 存在同一個 board）

進階 random walk

- Biasing the random walk towards user-specific pins
  為了推薦可以更符合使用者的需求，pin 與 board 針對不同的語言與 topics 有不同的設定．解決這個問題：在 random walk 時，對 user U，沿著同語言或 topics 邊（帶權重的邊）：Personalized Neighbor (E,U)
  而不做對不同 user 用不同的 graph
- Multiple query pins with weights
  推薦應與使用者的歷史（“購買紀錄”，已建立的 board）相關，與其考慮一個 query pin，我們考慮多個 query．有一個 query set $Q = \{(q, w_q)\}$ 有多個 pin，推薦 pin 若跟較多個 query 有關係，它的重要性應該比較高．分別計算每個 query $q \in Q$ 的 counter $V_q[p]$．因為 degree 較高的 q，被很多 board 加，需要較長的 step 才走得出來．所以，對每個 query 有不同 random walk 的長度有不同的 scaling factor

$$s_q = |E(q)| \cdot (C - \log |E(q)|)$$

where $C = \max_{p \in P} |E(p)|$ maximal pin degree．每個 random walk starting from pin q 的長度：

$$N_q = w_q N \frac{s_q}{\sum_{r \in Q} s_r}$$

- Multi-hit Booster
  推薦 pin 若跟較多個 query 有關係，它的重要性應該比較高，計算與所有 query set 的互動加權分數

$$V[p] = \left(\sum_{q \in Q} \sqrt{V_q[p]} \right)^2$$

- Early Stopping
  是否要對每個 query pin 都維持 random walk N_q 的長度，若 walk 的至少有 n_p 個 candidate pins，每個至少拜訪 n_v 次，就先結束此 random walk

Algorithm 2：Random Walk with early stopping

Algorithm 2 Pixie Random Walk algorithm with early stopping.

PIXIERANDOMWALK(q: Query pin, E: Set of edges, U: User personalization features, α: Real, N: Int, n_p: Int, n_v: Int)

totSteps = 0, $V = \vec{0}$
nHighVisited = 0
repeat
    currPin = q
    currSteps = SampleWalkLength(α)
    for i = [1 : currSteps] **do**
        currBoard = E(currPin)[PersonalizedNeighbor(E,U)]
        currPin = E(currBoard)[PersonalizedNeighbor(E,U)]
        V[currPin]++
        if V[currPin] == n_v **then**
            nHighVisited++
    totSteps += currSteps
until totSteps $\geq N$ or nHighVisited $> n_p$
return V

Figure: Randomwalk

Session Co-occurrence: Pin2Vec

雖然 pin-board graph 有很好的 recall，但把固定的 pin 綁在同一個 board，有他的局限性：有些 board 存在很久，但使用者的興趣可能已經轉移．另外：board 有可能太窄：一種威士忌 x 和一杯以 x 做成的雞尾酒可能會被儲存成不同 board．若可以更善用即時的使用者行為：瀏覽及點擊行為．

使用 learning 的模型生成 related pins：從使用者的點擊行為和 w2v 的學習方式

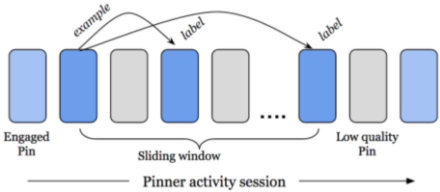


Figure 3. Extract Pin2Vec training pairs from Pinner activity history.

Figure: Randomwalk

window 只開單邊：與之後的點擊 pin 拉近

Pin2Vec

使用 64 個 negative sampling。

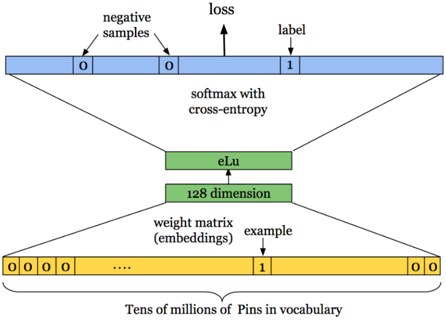


Figure: Randomwalk

假設 window size c，loss function：

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=i+1}^{i+c}\log p(w_{i+j}|w_i)$$

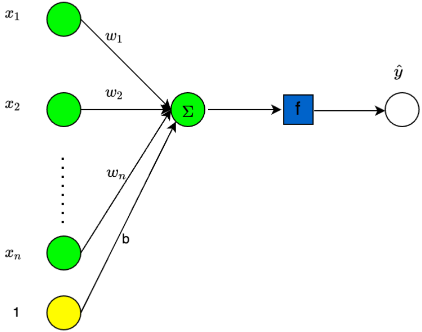
Pin2Vec Comparison

在 Pinterest 的實驗裡，Pin-board Graph 的效果比 Pin2Vec 好：

Figure 7. The Related Pins generated using board co-occurrence and Pin2Vec.

Perceptron

感知器（英語：Perceptron）是 Frank Rosenblatt 在 1957 年就職於康奈爾航空實驗室（Cornell Aeronautical Laboratory）時所發明的一種人工神經網路。它可以被視為一種最簡單形式的前饋神經網路，是一種二元線性分類器。



Perceptron

設有 n 維輸入的單個感知機 (單顆神經元)，為 $x_1,x_2,\cdots,x_n$ 的 n 維輸入向量，w_1, w_2, $\cdots$, w_n 為各個輸入向量連接到感知機的權量（或稱權值），b 為偏置 (bias)，f 為激活函數（activation function），$\hat{y}$ 為輸出

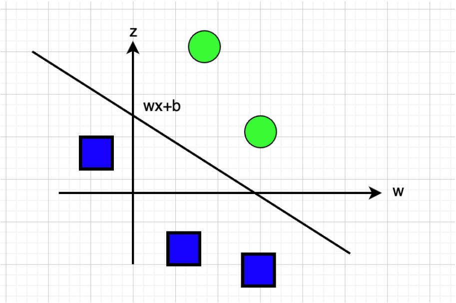
$$\hat{y} = f(\sum_{i=1}^{n} w_i x_i + b) = f(\mathbf{w}^T \mathbf{x} + b),$$

where $\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$, $\mathbf{w} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix}$, $f(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$.

Single-layer Percetron

Forward

Percetron 是一種線性分類器



缺點：不能處理非線性不可分問題

BackPropagation-Supervised learning

源自論文：Learning representations by back-propagating errors by Rumelhart, Hinton, Williams (1986)
Data set $D = \{(x_1, y_1), \cdots, (x_N, y_N)\}$, $\hat{y} = f(z)$.
Loss function

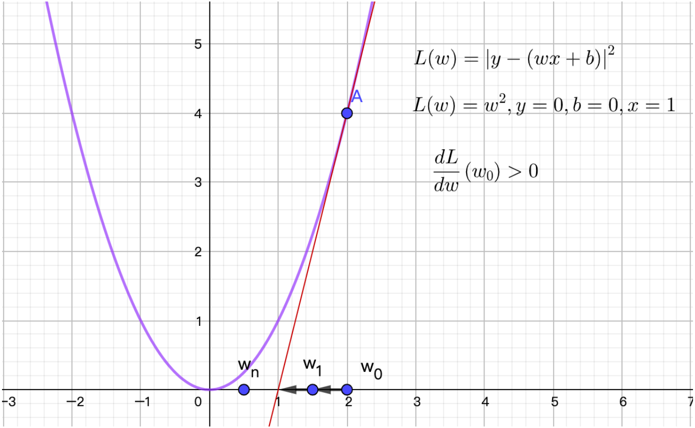
$$L = \frac{1}{N} \sum_{i=1}^{N} |\hat{y}_i - y_i|^2$$

Learning 的目標是求 L 的極小值
參數更新的方式：

$$w_i(t+1) = w_i(t) - \lambda \frac{\partial L}{\partial w_i}(w(t))$$

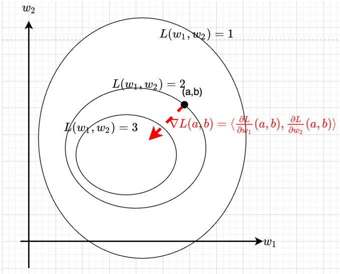
λ: learning rate (參數更新的步伐大小, λ 大，loss 震盪大，λ 小，loss 下降穩定
為何負號？

Minimizing problem



Gradient and Gradient Descent

- 求 L 極大值：參數更新往 $\nabla L(w)$ 遞增的方向
- 求 L 極小值：參數更新往 $-\nabla L(w)$: 遞減的方向



Parameters are updated by gradient descent (梯度下降法)

$$w_i(t+1) = w_i(t) - \lambda \frac{\partial L}{\partial w_i}(w(t))$$

Supervise Learning 監督式學習

- Labels
- Models (萃取特徵＋ Classifying layer）
- Objective functions

Chain Rule

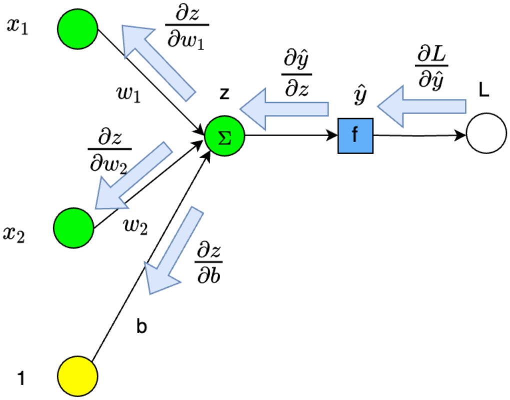
$$\frac{\partial L}{\partial w_i} = \frac{2}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)(-\frac{\partial \hat{y}}{\partial w_i}),$$

where

$$\frac{\partial \hat{y}}{\partial w_i} = \frac{\partial f}{\partial z}\cdot\frac{\partial z}{\partial w_i} = \frac{\partial f}{\partial z}\cdot x_i$$

for $z = \sum_{i=1}^{n} w_i x_i + b$. However f in the above is not differentiable.

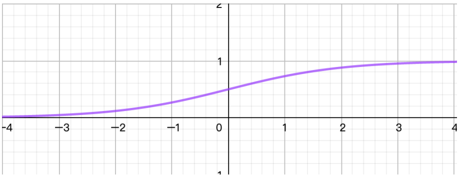
Backprapagation



Activation functions

但是上述 f 其實不是一個好的分類器，且不連續，後來較常用的是 sigmoid function

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



常用的 activation functions: $\tanh(x)$, ReLU, Leaky ReLU, $\cdots$

Updating parameters

$$
\begin{aligned}
\frac{\partial L}{\partial w_i} &= \frac{2}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)(-\frac{\partial \hat{y}}{\partial w_i}) \\
&= -\frac{2}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i) \cdot \frac{\partial f}{\partial z} \cdot x_i \quad \text{if } f(z) = \frac{1}{1+e^{-z}} \\
&= -\frac{2}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i) \cdot \left(\frac{e^{-z}}{(1+e^{-z})^2}\right) \cdot x_i \quad z = \sum_{i=1}^{n} w_i xi + b
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial L}{\partial b} &= \frac{2}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)(-\frac{\partial \hat{y}}{\partial b}) \\
&= -\frac{2}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i) \cdot \frac{\partial f}{\partial z} \cdot 1 \quad \text{if } f(z) = \frac{1}{1+e^{-z}} \\
&= -\frac{2}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i) \cdot \left(\frac{e^{-z}}{(1+e^{-z})^2}\right) 1 \quad z = \sum_{i=1}^{n} w_i x_i i + b
\end{aligned}
$$

Example

Given a simple neural network as following: Input data $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2)\}$, *where* $\mathbf{x}_1 = (2, 0, 1)^T$, $y_1 = 1$, $\mathbf{x}_2 = (0, 1, 0)^T$, $y_2 = 0$. *The hidden layer is a 1-dimension space*

$$z = h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b.$$

The output is

$$\hat{y} = \sigma(h(\mathbf{x})), \qquad \text{where } \sigma(x) = \frac{1}{1+e^{-x}} \text{ is the sigmoid function.}$$

This network is supervised by minimizing the L_2*-norm (MSE=Mean Square Error)*

$$L = \frac{1}{2}\sum_{i=1}^{2} |y_i - \hat{y}_i|^2$$

Compute the first update of the weight $\mathbf{w}^T = (w_1, w_2, w_3)$ *and* b *through backpropagation with learning rate 1. Suppose the initial weights are* $\mathbf{w}(0) = (0.5, 0.3, 0.2)$, $b_0 = 1$.

Solution

$z_1 = \mathbf{w}(0)^T\mathbf{x}_1 + b = (0.5, 0.3, 0.2) \cdot (2, 0, 1)^T + 1 = 2.2,$
$\hat{y}_1 = \sigma(z_1) = \frac{1}{1+e^{-2.2}} = 0.9$
$z_2 = \mathbf{w}(0)^T\mathbf{x}_2 + b = (0.5, 0.3, 0.2) \cdot (0, 1, 0)^T + 1 = 1.3,$
$\hat{y}_2 = \sigma(z_2) = \frac{1}{1+e^{-1.2}} = 0.786$

$$\frac{\partial L}{\partial w_1} = -\sum_{i=1}^{2}(y_i - \hat{y}_i) \cdot \left(\frac{e^{-z_i}}{(1+e^{-z_i})^2}\right) \cdot x_1$$

$$\frac{\partial L}{\partial w_1}(\mathbf{w}(0), b(0)) = -(1-0.9)0.9(1-0.9)2 - (0-0.786)0.786(1-0.786) \cdot 0 = -0.0$$

$$\frac{\partial L}{\partial w_2} = -\sum_{i=1}^{2}(y_i - \hat{y}_i) \cdot \left(\frac{e^{-z_i}}{(1+e^{-z_i})^2}\right) \cdot x_2$$

$$\frac{\partial L}{\partial w_2}(\mathbf{w}(0), b(0)) = -(1-0.9)0.9(1-0.9)0 - (0-0.786)0.786(1-0.786) \cdot 1 = 0.13$$

Solution

$$\frac{\partial L}{\partial w_3} = -\sum_{i=1}^{2}(y_i - \hat{y}_i) \cdot \left(\frac{e^{-z_i}}{(1+e^{-z_i})^2}\right) \cdot x_3$$

$$\frac{\partial L}{\partial w_3}(\mathbf{w}(0), b(0)) = -(1-0.9)0.9(1-0.9)1 - (0-0.786)0.786(1-0.786) \cdot 0 = -0.0$$

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{2}(y_i - \hat{y}_i) \cdot \left(\frac{e^{-z_i}}{(1+e^{-z_i})^2}\right) \cdot 1$$

$$\frac{\partial L}{\partial b}(\mathbf{w}(0), b(0)) = -(1-0.9)0.9(1-0.9)1 - (0-0.786)0.786(1-0.786) \cdot 1 = -0.00$$

$$w_1(1) = w_1(0) - \lambda\tfrac{\partial L}{\partial w_1}(w_1(0)) = 0.5 - (-0.018) = 0.518$$
$$w_2(1) = w_2(0) - \lambda\tfrac{\partial L}{\partial w_2}(w_2(0)) = 0.3 - (0.132) = 0.168$$
$$w_3(1) = w_3(0) - \lambda\tfrac{\partial L}{\partial w_3}(w_3(0)) = 0.2 - (-0.009) = 0.209$$
$$b = b(0) - \lambda\tfrac{\partial L}{\partial w_1}(b(0)) = 0.2 - (-0.009) = 0.209$$

作業四

Given a simple neural network as following: Input data $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2)\}$, where $\mathbf{x}_1 = (2, 0, 1)^T$, $y_1 = 1$, $\mathbf{x}_2 = (0, 1, 0)^T$, $y_2 = 0$. The hidden layer is a 1-dimension space

$$z = h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b.$$

The output is

$$\hat{y} = \sigma(h(\mathbf{x})), \qquad \text{where } \sigma(x) = \frac{1}{1 + e^{-x}} \text{ is the sigmoid function.}$$

This network is supervised by minimizing the Binary Cross-Entropy

$$L = \frac{1}{2} \sum_{i=1}^{2} (y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i))$$

Compute the first update of the weight $\mathbf{w}^T = (w_1, w_2, w_3)$ and b through backpropagation with learning rate 1. Suppose the initial weights are $\mathbf{w}(0) = (0.5, 0.3, 0.2)$, $b(0) = 1$.

Word2vec

論文：Distributed Representations of Words and Phrases and their Compositionality 2013

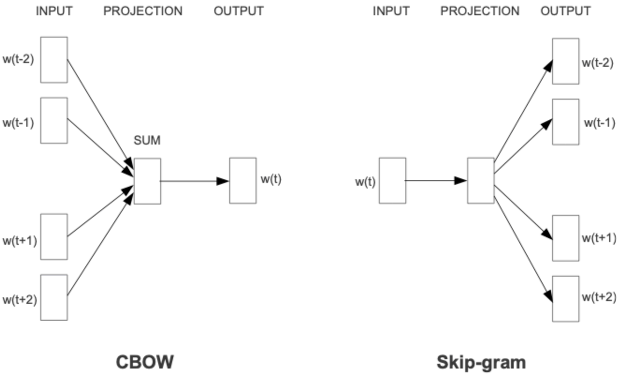
特色：

- Language Model (字-> 向量)
- Self-supervised Learning
- Word similarity

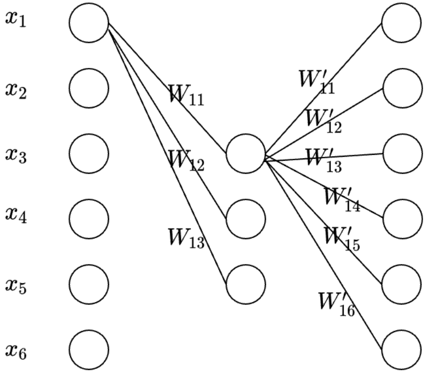
Machine Learning

- Supervised Learning 監督式學習 (ground truth labels+objective function)
- Self-supervised Learning 自監督式學習（標籤是與生俱來）
- Semi-supervised Learning 半監督式學習（帶少量標籤、部分標籤）
- Unsupervised Learning 無監督式學習（不帶標籤）K-means, auto-encoder

Cbow and Skip-gram



Model



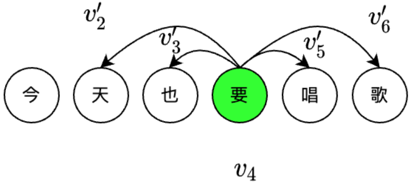
$$y^T = (x^T W) W', p = \mathrm{softmax}(y)$$

$$W : \mathbb{R}^6 \to \mathbb{R}^3,\ W = \begin{pmatrix} W_{11} & W_{12} & W_{13} \\ W_{21} & W_{22} & W_{23} \\ \vdots & \ddots & \vdots \\ W_{61} & W_{62} & W_{63} \end{pmatrix}.$$

$$x_i^T = (0, \cdots, 1_{i-th}, \cdots, 0)\ ,\ x_i^T W = v_i\text{: input vector},\ W = \begin{pmatrix} v_1 & \\ \vdots & \\ v_6 & \end{pmatrix}$$

$$W' : \mathbb{R}^3 \to \mathbb{R}^6,\ W' = \begin{pmatrix} W'_{11} & W'_{12} & \cdots & W'_{16} \\ W'_{21} & W'_{22} & \cdots & W'_{26} \\ W'_{31} & W'_{62} & \cdots & W'_{63} \end{pmatrix} = \begin{pmatrix} v'_1 & v'_2 & \cdots & v'_6 \end{pmatrix}.$$

Training: Loss functions



Given a sequence of words $w_1, w_2, \cdots, w_N$,

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t),$$

where the basic Skip-gram formulation defines $p(w_{t+j}|w - t)$ using the softmax function:

$$p(w_O|w_I) = \frac{\exp((v'_{w_O})^T v_{w_I})}{\sum_{j=1}^{N}\exp((v'_{w_j})^T v_{w_I})}$$

where v_w is the input vector of w and v'_w is the output vector of w.

$$v^T = x^T W$$
$$y^T = v^T W' = v^T \cdot \begin{pmatrix} v'_1 & v'_2 & \cdots v'_N \end{pmatrix} = \left(v^T \cdot v'_1, \cdots, v^T \cdot v'_N\right)$$

$$y = \begin{pmatrix} (v')_1^T \cdot v \\ (v')_2^T \cdot v \\ \vdots \\ (v')_N^T \cdot v \end{pmatrix} \Longrightarrow p = \operatorname{softmax}(y) = \begin{pmatrix} \frac{\exp((v')_1^T \cdot v)}{\sum_{j=1}^N \exp((v')_j^T \cdot v)} \\ \frac{\exp((v')_2^T \cdot v)}{\sum_{j=1}^N \exp((v')_j^T \cdot v)} \\ \vdots \\ \frac{\exp((v')_N^T \cdot v)}{\sum_{j=1}^N \exp((v')_j^T \cdot v)} \end{pmatrix}$$

Loss is a Cross-Entropy

$$L = -\frac{1}{N} \sum_{i=1}^{N} t_i \log p_i$$

$x_i^T = (0, \cdots, 1_{i-th}, \cdots, 0), t_i = (0, \cdots, , 0, 1, 1, 0_i, 1, 1, 0, \cdots)$

Negative sampling

- Noise words : if $P_\alpha(w) = \frac{\text{count}^\alpha(w)}{\sum_{w'} \text{count}^\alpha(w')} > P(w)$, then w is a rare word for $\alpha = 3/4$.

$$P_\alpha(a) = \frac{0.99^{0.75}}{0.99^{0.75} + 0.01^{0.75}} = 0.97 \qquad P(a) = 0.99$$

$$P_\alpha(b) = \frac{0.01^{0.75}}{0.99^{0.75} + 0.01^{0.75}} = 0.03 \qquad P(b) = 0.01$$

- Negative samples: 從 (c,c,t,c,c) 外抽樣, $P_\alpha(w)$:noise distribution

Negative sampling

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{-c\le j\le c, j\neq 0}\left(\log\sigma((v'_{t+j})^T v_t) + \sum_{i=1}^{k}\mathbb{E}_{w_i\sim P_n(w_i)}[\log\sigma(-(v'_i)^T v_t)\right),$$

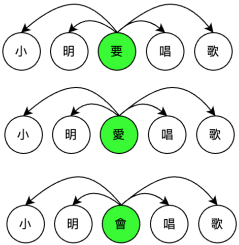
Subsampling of frequent words: discarded $P(w) = 1 - \sqrt{\frac{t}{f(w)}}$, t threshold 10^{-5}

$$P(a) = 1 - \sqrt{\frac{10^{-5}}{10^{-3}}} = 1 - 0.1 = 0.9$$

$$P(b) = 1 - \sqrt{\frac{10^{-5}}{10^{-1}}} = 1 - 0.01 = 0.99$$

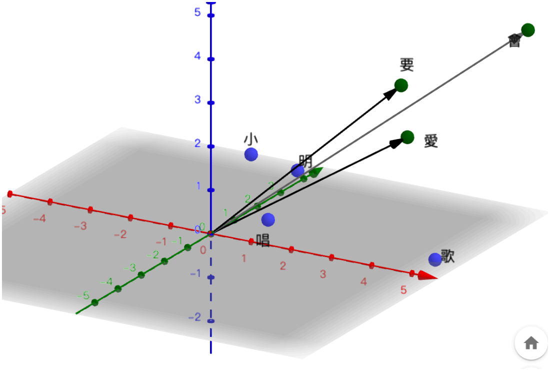
$$P(c) = 1 - \sqrt{\frac{10^{-5}}{10^{-7}}} = 1 - 10 < 0 \quad \text{無條件保留}$$

Word Similarity



Word Similarity of u, v : $\cos(\theta) = \frac{u \cdot v}{|u||v|}$

Vector Space



Word embedding

Not -1hot