Two way improve k-means ·
① repeat ② better 初始化

RBF = only one hd layer, diff activation · local approxmate

MLP: same actvation in layers, inner product, global approximate, more hd.

Example: 3 $\{x, d\}$ $\{(1,0), 3\}$ $\{(0,1), -1\}$. $\{(1,1), 5\}$ · $\sigma_i = 1$

1) compute hidden layers ·

$\varphi_{11} = 1$ $\quad \varphi_{12} = \exp(-\frac{(1-0)^2+(0-1)^2}{2})$
$= \exp(-1)$ ·

$\varphi_{13} = \exp(-0.5)$ ....

$$\Rightarrow \varphi = \begin{pmatrix} 1 & 0.3679 & 0.6025 \\ 0.3679 & 1 & 0.6025 \\ 0.6025 & 0.6025 & 1 \end{pmatrix}$$

2) calculate related weight vector and predict values of each input

$W = \underbrace{(\varphi^T\varphi)^{-1}}_{\checkmark} \underbrace{\varphi^T}_{\checkmark} \cdot d \quad d = \begin{pmatrix} 3 \\ -1 \\ 5 \end{pmatrix}$.

$\Rightarrow W = \begin{pmatrix} 0.0288 \\ -6.3007 \\ 8.7780 \end{pmatrix} \Rightarrow$ prediction · $e = \varphi W = \begin{pmatrix} 3 \\ -1 \\ 5 \end{pmatrix}$.

3) suppose only two examples.
$\{(1,0), 3\}$ $\{(0,1), -1\}$ are centers ·
recalculate $\quad \varphi = \begin{pmatrix} 1 & 0.3679 \\ 0.3679 & 1 \\ 0.6025 & 0.6025 \end{pmatrix}$.

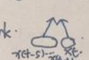
$\Rightarrow w\,? \Rightarrow e = \varphi w\ ? \neq d$ ·

4) for input vector (0, 0) calculater hd and output with 2), 3) ·

$\phi \to$ 把 (0, 0) 代入, w 已知 $\Rightarrow e$?

$\varphi = (0.6025, 0.6025, 0.3679)$ ·

Time-Series Model ·
时序相关 / wide-sense stationarity ·
difficulty: limit data / noise / non stationarity.
(model maintain state) dynamic network ·
RBF, MLP (static)
Time-Delayed Network ·

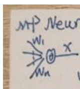



Elman Network (simple RNN) ·
$y = F_w(z^t)$, $z^t = F_v(z^{t-1}, x)$
(context layer)
RNN: $F(x(t), s(t-1)) / F(s(t))$
unfold to feed forward.
Vanish Gradients: BPTT with $\sigma(\cdot)$ no · dectectable slope
LSTM $\odot$ o(t) ·
C $\odot$ $i(t) = F(x(t), y(t-1))$
$z(t) = F(x(t), y(t-1))$
CEC (memory cell):
$c(t) = z(t) \odot i(t) + c(t-1)$ 不受 non-linear
$y(t) = h(c(t)) \odot o(t)$
No-decaying error / no fine tune.
$h(c(t))$ unbondaded
PCA: $AV = \lambda V \Rightarrow |A - \lambda I| = 0$ ·
$S = U \Lambda U^{-1}$ if $S \in R^{m \times m}$ ·
$\Lambda = diag(\lambda_1, \cdots \lambda_m)$ V is 单位向量
$A \in R^{m \times n} = U \Sigma V^T$ ·
$U\ (AA^T)\ V(A^TA)\ \sigma_i = \sqrt{\lambda_i}$
$\Sigma = diag(\sigma_1, \cdots \sigma_r)$ · $\lambda_1, \cdots \lambda_r\ AA^T$
Eginface $b = v_1^T x$ face $= V$ ·
$v_1^T x \cdot v = \lambda_1$

Unsupervised learning
eg: Hebb's rule ·
Oja's rule $w_{ji}(n+1) = \frac{w_{ji}(n) + \eta x_i(n) y(n)}{\sqrt{\sum_i [w_{ji}(n) + \eta x_i(n) y(n)]^2}}$
单层线性无监督
$\Rightarrow w_{ji}(n+1) = w_{ji}(n) + \eta y_j(n)(x_i(n) - y_j(n) w_{ji}(n))$.
deflation method.
PCA in NN encoder/decoder ·
k-means not optimal.
Competitive learing are winner-takes-all units ·
Winner: $h_j = \sum w_{ji} x_i$, $W_{j^*} \cdot x \geq W_j \cdot x$ ·
$\Rightarrow \begin{cases} y_{j^*} = 1 \\ y_j = 0 \text{ if } j \neq j^* \end{cases}$ $\quad \Delta w_{ji} = \eta y_i (x_i - w_{ji})$
leaky learning $w(t+1) = w(t) + \begin{cases} y_w (x - w(t)) \\ y_L (x - w(t)) \end{cases}$.
$\Rightarrow$ losing unit $\to$ dense · $\quad y_w >> y_L$
Maxnet (WTA Ex) ·
Ex: SCL · $\Delta w_{ji} = \eta y_j (x_i - w_{ji})$ ·
$\begin{cases} y_{j^*} = 1 \\ y_j = 0\ j \neq j^* \end{cases}$ $\quad \eta = 0.5$
3 prototype · Eudcilian distance
[0.14 0.75 0.71] [0.99 0.51 0.37] [0.73 0.81 0.87]
$\to$ 算 distance $\to$ 只对一个 update 把 6 个点都算一遍 ·
SOM (Self Organize Map) reduce dim / vector quantization
transform input into 1-2 d.
$\begin{cases} \text{Competition} \\ \text{Cooperation} \cdot \\ \text{Synaptic Adaption} \cdot \end{cases}$ $\begin{cases} \text{approximate} \\ \text{density matching} \\ \text{topological ordering} \\ \text{feature selection} \cdot \end{cases}$
$W_j(t+1) = W_j(t) + \eta(t) h_{j,i}(t)(x - W_j(t))$
$\eta_0 \exp(-\frac{t}{\tau_1})$ · $\exp(-\frac{d_{ji}^2}{2\sigma(t)})$

Associative Memory & Pattern Association
auto $(x = y)$ associate / hetero - ~
The Hopfield Network · (dM / energy)
$w_{ij} = \frac{1}{N} \sum_{p=1}^{P} x_i^p x_j^p$ $\quad w_{ij} = w_{ji}$, $w_{ii} = 0$ ·
Asychronous update rule ·
$H_i(t+1) = \sum_{j=1, j \neq i}^{n} w_{ij} V_j(t) + I_i$ ·
$V_i(t+1) = sgn(H_i(t+1)) = \begin{cases} 1 & \geq 0 \\ -1 & < 0 \end{cases}$
limitation: number store limited
unstable if shares bits common
Ex: 4-d patterns $x_1 = (1, 1, -1, -1)$
$x_2 = (-1, -1, 1, 1)$
$x_3 = (-1, 1, 1, -1)$
$x_4 = (1, 1, 1, -1)$ ·
1) compute W ·
$W_{11} = W_{22} = \cdots W_{44} = 0$ ·
$W_{12} = W_{21} = \frac{1}{4}(1 + 1 - 1 + 1) = \frac{2}{4}$ $\quad W = \frac{1}{4}\begin{pmatrix} 0 & 2 & -2 & -2 \\ 2 & 0 & 0 & \cdots \\ -2 & 0 & 0 & \cdots \\ -2 & -4 & 0 & \cdots \end{pmatrix}$
$W_{13} = W_{31} = \frac{1}{4}(-1 - 1 - 1 + 1) = -\frac{2}{4}$
⋮
2) Recover $x_5 = [1, ?, -1, -1]$
$y^j(t+1) = \sum_{n=1, n \neq j}^{4} W_{jn} x^n(t)$ ·
$= \frac{1}{4}(2 * 1 + 0 * (-1) + (-4) * (-1) + (0))$
$= \frac{6}{4} \geq 0 \Rightarrow x^5(0) = 1$ ·
3) node 1 > 2 > 3 > 4 noisy [1, -1, 1, -1]
找式子 node 1: $y^1(0) = \frac{1}{4}(2 * (-1) + (-2) * 1 + (-2) * (-1)) + 1 = \frac{2}{4} \Rightarrow x^1(0) = 1$ ·
node 2: $y^2(0) = \frac{1}{4}(\cdots) = \frac{2}{4} \Rightarrow x^2(0) = 1$ ·
nod 3: $y^3(0) = \frac{2}{4}$ $x^3(0) = 1$ 4 node $-\frac{10}{4}, -1$
4) no external
$\Rightarrow$ $-\frac{2}{4}, -1$
$\frac{2}{4}, 1$
$\frac{2}{4}, 1$
$-\frac{2}{4}, -1$

MP Neuron (1943)




$a_i^t = \begin{cases} 0 \\ 1 \end{cases}$
$w_i = \begin{cases} +1 \\ -1 \end{cases}$

$s^t = \sum_i w_i a_i^t \geq \theta$

Heaviside funtion: $H(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$

Hebb's rule: $w_{ji}^{k+1} = w_{ji}^{k} + \Delta w_{ji}^{k}$
where $\Delta w_{ji}^{k} = C a_i^k x_j^k$

$\Rightarrow$ propose correlation of Hebbian synapse ·
but: it only increase, wash out diff performance · in nearon ·

Supervised learning: Casuality
train set and test set identical

Perceptron: 1958 "error correcting rule"
$S_j = \sum_{i=0}^{n} w_{ji} a_i$, taken $n+1$ inputs, a bias input
bias is constant, but its weight can be changed ·

Perceptron learning (weight adjustment / delta rule)
$e_j = (t_j - x_j)$, $x_j = f(S_j) = \begin{cases} 1, & \sum_{i=0} w_{ji} a_i \geq \theta_j \\ 0, & < \theta_j \end{cases}$
$W_{ji} = W_{ji} + \Delta W_{ji}$, $\Delta W_{ji} = C \cdot e_j \cdot a_i$
perceptron converges: lr is small; linear seperable ·
perceptron performance: RMS (root-mean-square)
$\sqrt{\frac{\sum (x_i - \hat{x}_i)^2}{N}}$ only connected with weights ·
perceptron classification: decision boundary $w \cdot x^T = 0$
or $b + \sum_{i=1}^{n} x_i w_i = 0$; can not solve "XOR" use $w_1 x_1 + w_2 x_2 + \cdots = S > 0 / \leq 0$

Gradient Deseent Rule: $E(w) = \frac{1}{2} \sum_e (y_e - O_e)^2$.
$w_i = w_i - \eta \frac{\partial E}{\partial w_i}$

As $\frac{\partial E}{\partial w_i} = \sum_e (y_e - O_e)(-x_{ie}) \Rightarrow w_i = w_i + \eta \sum_e (y_e - x_e) x_{ie}$

For Batch learning: $\Delta w_i = \Delta w_i + \eta (y_e - x_e) x_{ie}$ then $w_i = w_i + \Delta w_i$

For incremental ~: $w_i = w_i + \eta (y_e - x_e) x_{ie}$

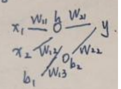
Sigmoidal · $\sigma(s) = \frac{1}{1+e^{-s}}$
$S = \sum_{i=0}^{d} w_i x$, $\sigma'(s) = \sigma(s)(1 - \sigma(s))$.
$w_i = w_i + \eta \sum_e (y_e - O_e) \sigma(s)(1 - \sigma(s)) x_{ie}$.
parabolas 抛物线 ellipses 椭圆线
$\begin{cases} \text{perceptron: threashold, converges linear seperable} \cdot \\ \text{GD: converage min error} \cdot \end{cases}$
MLP: learn arbitary / continous / Boolean function
hyperbolic tangent $o = \tanh(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}}$ ·
$\tanh'(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}}$
$f(x) = \sigma(\sum_{j=1}^{J} w_{jk} \sigma(\sum_{i=1}^{I} w_{ij} x_i + w_{0j}) + w_{0k})$

$o_j = \sigma(S_j) = \frac{1}{1+e^{-s_j}}$, $S_j = \sum_{i=0}^{d} w_{ij} o_j$
$O_k = \sigma(S_k) = \frac{1}{1+e^{-s_k}}$, $S_k = \sum_{i=0}^{d} w_{jk} o_j$
Backpropagation:
$\Delta W_{jk} = \eta \beta_k o_j$. $\Delta W_{0k} = \eta \beta_k$.
$\beta_k = O_k(1 - O_k)(y_k - O_k)$
$\Delta W_{ij} = \eta \beta_j o_i$, $\Delta W_{0j} = \eta \beta_j$
$\beta_j = o_j(1 - o_j)(\sum_k \beta_k W_{jk})$ 不包含偏置权重

Revision by example (online training)
↳ incremental ·
Revision by epoch = Batch learning
Issuses of BP $\begin{cases} \text{random presentation} \to \text{better results} \\ \text{random initialization} \cdot \\ \text{hidden layer (not add represtation power)} \\ \text{learning rate} \\ \text{Momentum (smooth / fast converge)} \end{cases}$
$\Delta W(t) = -\eta \frac{\partial E}{\partial w(t)} + \alpha \Delta W(t-1)$
Generalization & Overfitting (error small · new data high).

Reasons of overfit: number of free param is bigger
Measure: early stop / network purning / regularize / weight decay (penalty term)
k-fold cross validation (useful small sets)
MLP limitations: sensitive to hd layers / lr rate

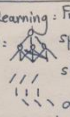
$\{(1,0), 1\}$ $\{(0,1), 0\}$ $\Delta W(t) = -\eta \frac{\partial E}{\partial W(t)} + \alpha \Delta W(t-1)$ · $\eta = 0.2, \alpha = 0.9$, weigh = 1
处理样本 1: forward: $z_1 = w_{11} \times 1 + w_{12} \times 0 + w_{13} \times b_1$ $\quad$ two input, one hidden one output ·
$= 1 + 0 + 1 = 2$
$h = \sigma(z_1) = \frac{1}{1+e^{-2}} = 0.8808$, $\sigma'(z) = 0.1040$ ·
$z_2 = w_{21} \times h + w_{22} \times b_2 = 0.8808 + 1 = 1.8808$ ·
$O_k = \sigma(z_2) = 0.8675$ $\quad \sigma'(z_2) = 0.1148$ ·
backward: $\beta_2 = (y - O_k) \sigma'(z_2) = -0.0152$ ·
$\beta_1 = \beta_2 \cdot W_{21} \cdot \sigma'(z_1) = -0.0016$ ·
$\Rightarrow W_{11}, W_{12}, W_{21}, W_{22}, W_{13}$ ↳ 不要把 bias 加入 ·




同样 样本 2: ···

Deep Learning: Function composition · $\quad N * N$, $m * m$ filter $\to (N - m + 1)^\wedge 2$ ·

CNN:



sparse connectivity (receptive fields)
shared weights (feature map)
convolution $h_{ij}^k = \tanh((W^k x)_{ij} + b_k)$

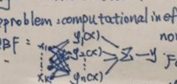
img $\to$ convolution $\to$ non-linearity $\to$ spatial pooling $\to$ normalization $\to$ feature map
(relu) $\max(0, x)$ ·
spatial pooling: $\begin{cases} \text{relative position} \\ \downarrow \text{spatial resolution} \cdot \\ \text{(max) down sample / translation invariance.} \end{cases}$

Relu $\begin{cases} \text{simple} \\ \text{reduce times} \\ \text{sigmoid saturate} \cdot \end{cases}$

Transfer learning: share param + new adaption ·

RBF Network · $F(x) = \sum_{i=1}^{N} w_i \phi(\|x - x_i\|)$
Regularization Network: $F(x) = \sum_{i=1}^{N} w_i \exp(-\|x - x_i\|^2 / 2\sigma_i^2)$
↳ problem: computational inefficient / large $\phi^T\phi$ param ·

RBF:




nonlinear function transform $\to$ easy linear mapping
$F(x) = \sum_{i=1}^{M} w_i \exp(-\frac{\|x - t_i\|^2}{2\sigma_i^2})$ $\quad$ M 个数 $t_i$ 中心点 $M < N$ ·
$\Phi W = d \Rightarrow W = (\Phi^T\Phi)^{-1}\Phi^T d$
kmeans: 选 k 个 data $\to$ 每个点到 k distance $\to$ 选 min 作为类 $\to$ 计算新的 k 个质点 Loop
↳ not suitable for non-convex shape / senstive to noise and out lier