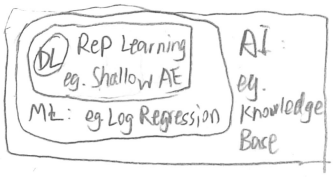
EIE4122 Deep Learning & DNN

§ Historial

1958 Rosenblatt's Perceptron Algo (X XOR Problem)
Backpropagation Algo (1986~1974)(x very deep)
mid-90's SVM (outperforms NN)
2006 Restricted Boltzman Machine to pretrain DNN



§ ML Basis

scalar: $x \in \mathbb{R}$
Vector: $\vec{x} \in \mathbb{R}^D$
Matrices: $\vec{A} \in \mathbb{R}^{M \times N}$
Tensor: $\vec{A} \in \mathbb{R}^{M \times K \times R}$

Vector Norm: $\|\vec{x}\|_p = (\sum_i |x_i|^p)^{\frac{1}{p}}$ $(P \geq 1)$
Frobenius Norm: $\|A\|_F = \sqrt{\sum_{i,j} A_{ij}^2}$
Dot product of 2 Vect: $\vec{x}^T \vec{w}$
Mat Mul: $\vec{Z} = \vec{W}\vec{X} = (\vec{X}^T \vec{W}^T)^T$

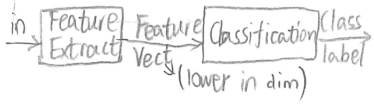
MLP:

$\vec{O} = f(\vec{W}\vec{x} + \vec{b})$

Vector space(V): set of vectors
(linear) subspace: $\mathbb{V}$ that is a subset of some larger $\mathbb{V}$

$\mathrm{Var}(f(x)) = E[(f(x) - E[f(x)])^2]$
$\mathrm{Var}(x) = E[(x - E[x])^2] = E[(x-\mu)^2] = \sigma$
$\mathrm{Cov}(f(x), g(y)) = E[(f(x) - E[f(x)])(g(y) - E[g(y)])]$
$\mathrm{Cov}(x, y) = E_{x,y}[(x - E[x])(y - E[y])] = \sigma_{xy}$

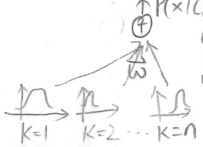
Structure of PR sys.



Bayes
$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$ → likelihood, prior: prob without any given info, → Marginal Prob
↓ posterior

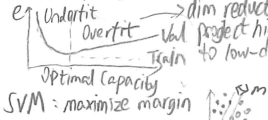
for K-class classification
$k = \underset{i=1}{\overset{K}{\operatorname{argmax}}}\, p(\vec{x}|\omega_i)$ (if $P(\vec{x}|\omega_k) > P(\vec{x}|\omega_i)$)
or $k = \underset{i=1}{\overset{K}{\operatorname{argmax}}}\, P(\vec{x}|\omega_i) P(\omega_i)$

GMM (Gaussian Mixture Model)



EM Algo (最大期望算法)

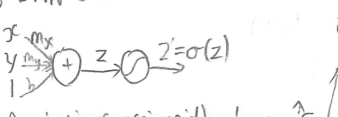
Model capacity



SVM: maximize margin

$f(x) = \sum_{i \in S} \alpha_i y_i \vec{x}_i^T \vec{x} + b$
good for binary classification

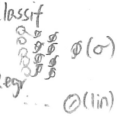
§ DNN Basic



Activation Func:
σ (sigmoid) $= \frac{1}{1+e^{-z}}$
Relu $= \max(z, 0)$
$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

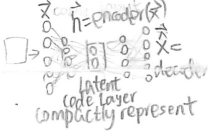
developed in 2010
alleviate gradient vanishing
allow hidden layer to output true 0 (speed up training, sparse rep)

DNN (MLP)



§ Deep Arch

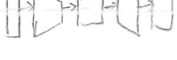
AE (autoencoders)



using $E_{mse}(\vec{w}) = \frac{1}{N}\sum_{i=1}^{N} \|\vec{x}_i - \vec{x}'_i\|^2$

VAE (Variational Auto Encoder)

DAE (Denoising Autoencoder)



loss func: $E_{mse}(\vec{w}) = \frac{1}{N}\sum_{i=1}^{N} \|\vec{x}_i - \hat{\vec{x}}_i\|^2$

Sequential Data

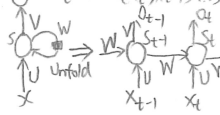
- Forced FNN

$\vec{O}_t = DNN(\vec{X}_t, \vec{X}_{t-1}, \vec{X}_{t-2}, \vec{X}_{t-3})$
⊗ limited temporal context
huge input dim if large context

more flexible smarter

- RNN

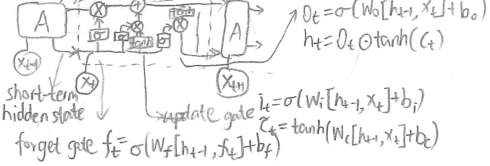
shared $\vec{W}, \vec{U}, \vec{V}$
$\vec{O}_t = RNN(\vec{x}_t, \vec{x}_{t-1}, \ldots, \vec{x}_{t-M})$



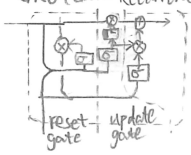
$\vec{s}_t = f(\vec{U}\vec{x}_t + \vec{W}\vec{s}_{t-1})$
$\vec{o}_t = \text{softmax}(\vec{o}'_t)$ $(\vec{o}'_t = \vec{V}\vec{s}_t)$
(Gen image description, mach tran, classify data sequence)
⊗ Can only look back a finite number of steps

long-term dependency

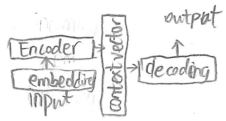
- LSTM



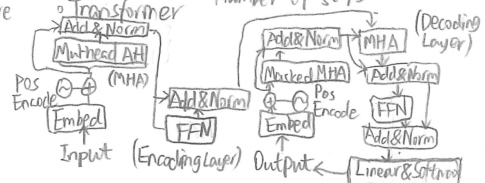
- GRU (Gate Recurrent Unit)

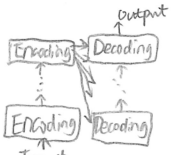


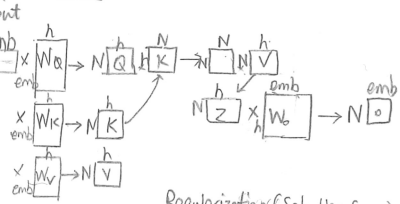
- Encoder-Decoder Architecture



- Transformer







§ DL. Math Fundamentals

$\vec{w} = \vec{w} - \eta \frac{\partial L(\vec{w})}{\partial \vec{w}}$

Minibatch SGD
$\vec{w}_B = \vec{w} - \eta \frac{\partial (\frac{1}{B}\sum_{(\vec{x},y) \in B} L(\vec{w}, y))}{\partial \vec{w}}$

Batchsize=50 (100)
1 epoch
2 iter

Batchsize too small: ziggle

Momentum (reduce ziggle, acce training)
$\Delta\vec{w} = \eta \frac{\partial L(\vec{w})}{\partial \vec{w}} + \gamma \Delta \vec{w}_{t-1}$ (especially noisy grad)

Loss Functions

$L_{mse} = \frac{1}{N}\sum_{i=1}^{N} \|\vec{y}_i - \vec{t}_i\|^2$
$L_{cce} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} t_{i,k} \log y_{i,k}$
$L_{bce} =$ (special case of CCE)

DNN Problem

1. Gradient Vanishing
Sol. ReLU Residual, Proper $\vec{w}$ init (eg. Xavier)
Batch norm, Layer norm, Scaled Exp LU
Xavier: $W_{ij}^{(l)} \sim N(0, \frac{2}{n_{l-1} + n_l})$

2. Gradient Exploding
Sol. Clip the gradient

3. (Almost) No data per class (few shot learning)
Sol. Transfer Learning, Meta Learning, Self-supervised

Regularization (Solution Space)

L2 Regularization
$L(\vec{w}) = L_{(CCE)}(\vec{w}) + \lambda \vec{w}^T \vec{w}$

L1 Regularization
$L(\vec{w}) = L_{(CCE)}(\vec{w}) + \lambda \|\vec{w}\|_1$