Lecture 7. Regularization 정규화 Generalization 일반화

L2 Regularization : $\tilde{J}(w;X,y) = \frac{1}{2}w^Tw + J(w;X,y)$

L1 Regularization : $\tilde{J}(w;X,y) = \alpha\|w\|_1 + J(w;X,y)$

Always called "Weight decay"

Noise → on hidden units : similar with regularization
Noise → on weights : reflact uncertainty (more stable)
Noise → on output targets : avoid over-confident

Parameter typing : $\Omega(w^{(A)}, w^{(B)}) = \|w^{(A)} - w^{(B)}\|_2^2$

- Weight decay : $\tilde{J}(\theta;X,y) = J(\theta;X,y) + \alpha\Omega(\theta)$
- Sparse Representation : $\tilde{J}(\theta;X,y) = J(\theta;X,y) + \alpha\Omega(h)$

augmentation

Summary : Most Popular Regularizers : L2, Early stopping, Dropout, Data

Lecture 8. Optimization 최적화

- Learning : Care about Pererformance (Including Test Set)
- Optimization : Care about $J(\theta)$

Emperical Risk Minimization : Learning Problem → Optimization Problem

$$\mathbb{E}_{x,y\sim\hat{p}(x,y)}[L(f(x;\theta),y)] \leftarrow \text{minimization}$$

Surrogate Loss : log-likehood ↔ 0-1 loss
(Reason : real loss is hard to optimization)

- Batch Optimization : Use all train set (also called deterministic)
- Minibatch Optimization : Use some train set
- Stochatic Optimization : Use single sample to train (also called online)

Larger Batch → Accurate ↑, Efficiency ↓ (low reword), Memory consuming ↑, Generalization ↓

Minibatch Select : Unbiased Estimation (By suffling and chucking)

△ Most time : Shuffle once and pass through is okay.

Challenge :
- ill condition : input ex affect output extrimely.
  - Taylor expansion $\longmapsto$ $x \mapsto x - \epsilon g$
- local minima : many ~~identify~~ model identifiability.
- saddle points & flat regions :
- cliffs & exploding gradients
- local and grobal structure

Stochastic Gradient Decent (SGD)

N.B. Stochastic, but use mini-batch

Algorithm.

Requirement : Learning Rate ε_k ($\frac{k}{\tau}$) $\quad \varepsilon_k = (1-\alpha)\varepsilon_0 + \alpha\varepsilon_\tau$

Requirement : Initial Parameter θ
- τ : few hundreds
- ε_τ : 1% of ε_0

$k \leftarrow 1$

While Do not meet stopping Condition do.

Small-batch $\{x^{(1)}, \cdots x^{(m)}\}$, $y^{(i)}$

Gradient $\hat{g} \leftarrow \frac{1}{m} \nabla_\theta L(f(x^{(i)}; \theta), y^{(i)})$

Update : $\theta \leftarrow \theta - \varepsilon \hat{g}$

$k+1$

end while

Other decay : $\alpha = \alpha_0 e^{-kt}$ & $\alpha = \alpha_0 / (1+kt)$

Momentum : $v \leftarrow \alpha v - \varepsilon \nabla_\theta g$ $\qquad g = \frac{1}{m} \sum_{i=1}^{m} L(f(x^{(i)}; \theta), y^{(i)})$

- poor condition (like ill condition)
- variance in stochastic gradient

Parameter initialization strategies (Must break Symmetry)

- Gaussian Initialization : By Gaussain or uniform distribution
- Normalized Initialization : $W_{i,j} \sim U\left(-\sqrt{\frac{6}{m+n}}, \sqrt{\frac{6}{m+n}}\right)$
- Sparse Initialization : fix

Grad

Adam Algorithm $r \leftarrow r + g \odot g$ (time applied element-wise) — Scale does matter / Near 0.

$\frac{\varepsilon}{\delta + \sqrt{r}} \cdot \odot g$. Problem : activation → 0

RMSProp Algorithm $r \leftarrow \rho r + (1-\rho) g \odot g$ (prevent too small learning rate)

$\frac{\varepsilon}{\delta + \sqrt{r}} \odot g$

Adam (RMSProp + Momentum)

$$s \leftarrow \rho_1 s + (1-\rho_1) g$$
$$r \leftarrow \rho_2 r + (1-\rho_2) g \odot g$$
$$\hat{s} \leftarrow \frac{s}{1-\rho_1^t}$$
$$\hat{r} \leftarrow \frac{r}{1-\rho_2^t}$$
$$\Delta\theta = -\varepsilon \cdot \frac{\hat{s}}{\sqrt{\hat{r}} + \delta}$$

Batch Normalization

Lecture 9. Convolutional Network. ⟶ all connections / local connections

- Convolution Layers
- Pooling Layers
- Non-Linearity Layers

Convolution. $f(x,y) \longrightarrow h(x,y) \longrightarrow g(x,y)$, $g = f * h$

$$g(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x+u, y+v)\cdot h(u,v)\cdot du\,dv$$

Pooling : max pooling.

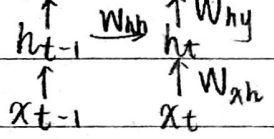
Poololing : $(F-1)/2$; e.g. $3 \mapsto 1$, $5 \mapsto 2$, $7 \mapsto 3$.

ConvNets : Input ⟶ (Conv ⟶ Nonlinearity ⟶ Pooling) ⟶ Output (Conv Layer)

e.g. LeNet : Conv → Pooling → Conv → Pooling → Conv → FC.

Lecture 10. Recurrent Networks

Recurrent Neural Netronk :



$h_t = f_W(h_{t-1}, x_t)$

$= \tanh(W_{hh} h_{t-1} + W_{hx} x_t)$

$y_t = W_{hy} \cdot h_t$

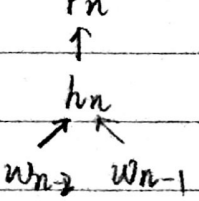
Origination : Larguege Models $p(w_1, w_2, \cdots w_N)$

$= p(w_1)\cdot p(w_2|w_1)\cdots p(w_N|w_1,\cdots w_{N-1})$

$= p(w_1)\cdot p(w_2|w_1)\cdots p(w_N|w_{N-1},\cdots \underbrace{w_{N-k}}_{\Delta})$

Coss function : $L = -\frac{1}{N}\sum_n w_n^T \log \hat{p}_n$

Trigram NN Language Model ⟶

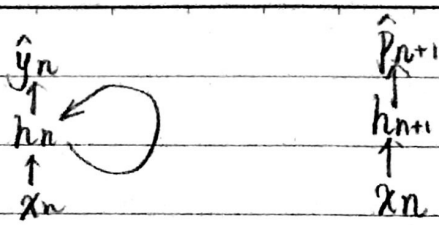


$h_n = g(V[w_{n-1}; w_{n-2}] + C)$

$\hat{p}_n = \text{softmax}(Wh_n + C)$

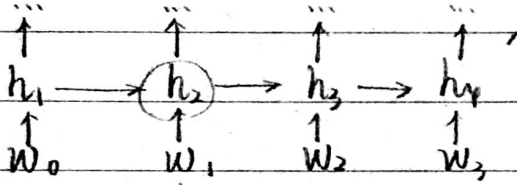
RNN $h_n = g(V[x_n ; h_{n-1}] + c)$

$\hat{y}_n = Wh_n + b$



v.s. { N-Gram → Limited history
RNN → Entire history (Long range correlations)

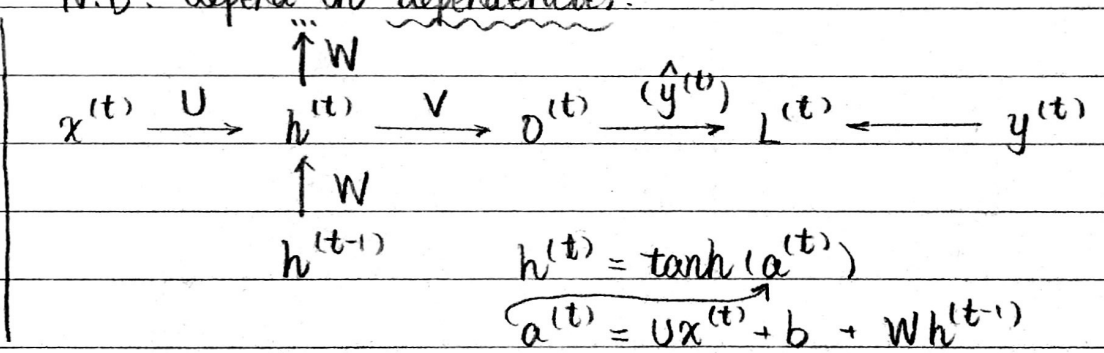
Train RNN



Overall $\mathcal{F}$

e.g. $\frac{\partial \mathcal{F}}{\partial h_2} = \frac{\partial \mathcal{F}}{\partial Cost_2} \cdot \frac{\partial Cost_2}{\partial \hat{p}_2} \frac{\partial \hat{p}_2}{\partial h_2} + \frac{\partial \mathcal{F}}{\partial Cost_3} \cdot \frac{\partial Cost_3}{\partial \hat{p}_3} \frac{\partial \hat{p}_3}{\partial h_3} \cdot \frac{\partial h_3}{\partial h_1} + \cdots$

N.B. depend on dependencies.



Problems : Short-term dependency.

→ LSTM Model & Gated Recurrent Unit.

1. Mini-batch Gradient Descent

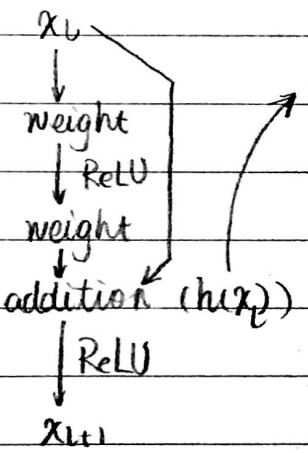
Q. Why the reward of larger samples is less than linear?

A. n : batch size. σ : Standard variance

batch → average → Standard variance : $\frac{\sigma}{\sqrt{n}}$ (Standard deviation)

i.e $\hat{\mu}$: estimate mean $SE(\hat{\mu}) = \sqrt{Var\left[\frac{1}{m}\sum_i x^{(i)}\right]} = \frac{\sigma}{\sqrt{m}}$ Δ

2. Residual neural network

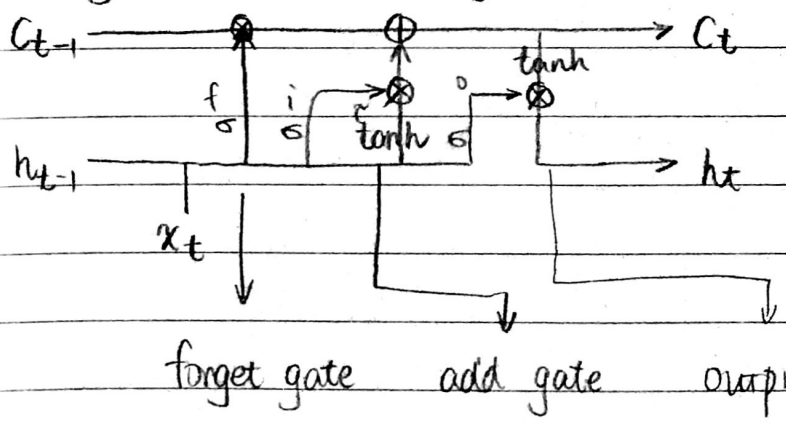


$x_{l+1} = x_l + \mathcal{F}(x_l, W_l)$

$h(x_l) = x_l + \mathcal{F}(x_l, W_l)$

F(x) in Slide.

3. Long short-term memory (LSTM) fico

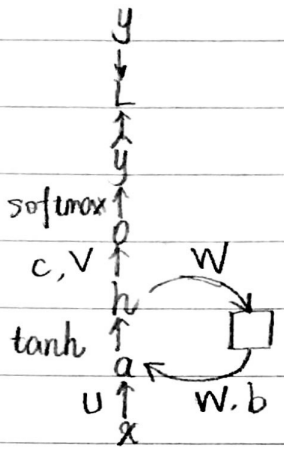


$f_t = \sigma(W_f[x_t, h_{t-1}] + b_f)$

forget gate add gate output gate

4. Back propagation through time (BPTT)

(next page.)



$a = Ux + Wh + b$ $\quad \theta = \{U, V, W, b, c\}$

$h = \tanh(a)$

$o = Vh + c$

$\hat{y} = \text{Softmax}(o)$

Remember ↑

$\nabla_c L = \sum_t \left(\frac{\partial o}{\partial c}\right)^T \cdot \nabla_o L = \sum_t \nabla_o L$

$\nabla_b L = \sum_t \left(\frac{\partial o}{\partial b}\right)^T \nabla_h L = \sum_t \underline{diag(1-h^2)\nabla_h L}$

$\nabla_V L = \sum_t \sum_i \left(\frac{\partial o}{\partial V}\right)^T \nabla_o L = \sum_t h^T \nabla_o L$

$\sum_t \sum_i \left(\frac{\partial L}{\partial o_i}\right)^T \nabla_V o_i$

△ o, h都是多了

$\nabla_W L = \sum_t \sum_i \left(\frac{\partial L}{\partial h_i}\right)^T \nabla_W h_i = \sum_t \nabla_h L \cdot diag(1-h^2) \cdot h$ ✓

5. Graph Convolution $h_v^k = \sigma\left(\underline{W_k} \cdot \sum_{u \in N(v)} \frac{h_u^{k-1}}{|N(v)|} + \underline{B_k} h_v^{k-1}\right)$

$N(v)$: v's neighbor's list