LEC7 DSA5204

Re-cap:

Regularization
- $\downarrow$ training loss
- $\uparrow$ generalization

① parameter norm penalty
② early stop
③ inject noise

Note: up to now, we do not change the model architecture.

---

1. Ensemble

a) Bagging (Boostrap Aggregating)

→ train m models through different subsets of data
→ then, aggregate the results
- regression : mean
- classification : vote

(uncorrelated)
idea: combine weak models to enhance performance
$\downarrow$
small variance

→ Theoretical Analysis (Toy Example)

consider a simple scalar model:

$$f_i(x) = f^*(x) + \varepsilon_i(x)$$

assume
$$\begin{cases} \mathbb{E}[\varepsilon_i(x)] = 0 \\ \mathbb{E}[\varepsilon_i^2(x)] = \sigma(x)^2 \\ \mathbb{E}[\varepsilon_i(x)\varepsilon_j(x)] = 0 \end{cases}$$

aggregate model: $\bar{f}(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$

$\boxed{\text{Expectation Error}}$

<u>consider</u>:
$$\begin{cases} E(x) := \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}[(f_i(x) - f^*(x))^2] \\ \bar{E}(x) = \mathbb{E}[(\bar{f}(x) - f^*(x))^2] \end{cases}$$

$$E(x) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}[\varepsilon_i^2(x)]$$
$$= \sigma(x)^2$$

$$\bar{E}(x) = \mathbb{E}\left[\left(\frac{1}{n}\sum_{i=1}^{n}\varepsilon_i(x)\right)^2\right]$$

$$= \frac{1}{n^2}\sum_{i=1}^{n}\mathbb{E}[\varepsilon_i^2(x)] + \frac{1}{n^2}\sum_{i\neq j}\mathbb{E}[\varepsilon_i(x)\varepsilon_j(x)]$$

$$= \frac{1}{n}\sigma(x)^2$$

<u>Rmk</u>: The <u>unrealistic</u> assumption is : <u>uncorrelation of model</u>

$\downarrow$

Trade-off between <u>Bias</u> and <u>Correlation</u>

- $\rightarrow$ Large Sample Size $\Rightarrow$ $\begin{cases} \text{Bias} \downarrow \\ \text{correlation} \uparrow \end{cases}$
- $\rightarrow$ Small Sample Size $\Rightarrow$ $\begin{cases} \text{Bias} \uparrow \\ \text{correlation} \downarrow \end{cases}$

2. Dropout → can be viewed as "Ensemble" (approximate)

efficient

overcome the disadvantage of "Ensemble" → Store many sub-models

☆ Suggest to put Dropout at the end

Dropout → apply random mask to weights

↓

Average to achieve aggregation

Mathematically, it can be formulated as:

$$f(x; W, b, \mu) = \sigma(W \cdot (\mu \circ x) + b)$$

f: for each layer

μ: node mask

→ μ can be viewed as d-dim Bernoulli R.V.

→ $\mu_i \sim \text{Bernoulli}(p)$

$$\begin{cases} \text{w.p. } p \text{, we keep the node} \\ \text{w.p } (1-p) \text{, we ignore the node (mask)} \end{cases}$$

→ this only works for "Training Step"

→ While Testing (inference), replace W with $p \circ W$

Rmk:

1. The Entire Model Ensemble is : (Stochastic Version)

$$\{ f(x;\theta,\mu) : \mu_i \in \{0,1\} \quad i \in [m] \}$$

2. There are 2^m models (distinct)

3. each of the model is not independent since they share the same θ

4. Ensemble $\longrightarrow$ $\frac{1}{n}\sum_{i=1}^{n} f_i(x)$

Stochastic Ensemble $\longrightarrow$ $\mathbb{E}_\mu[f(x;\theta,\mu)]$

$\rightarrow$ in training, use Dropout to Approximate $\mathbb{E}_\mu[\cdot]$ & introduce independence to decrease correlation

$\rightarrow$ in testing (inference), use "Weight Scaling Inference" to approximate $\mathbb{E}_\mu[\cdot]$

$$\boxed{W \rightarrow p \circ W}$$

$\rightarrow$ Theoretical Analysis (Toy Example)

$\downarrow$

We want to show :

for LR, Dropout $\Longleftrightarrow$ L2-norm regularization

$\downarrow$

(Simple Case)

[E.g.] → Linear Regression with Dropout

$$f(x; w, \mu) = \boxed{\frac{1}{p}} w^T (\mu \circ x)$$

→ compensate for $\mathbb{E}[\mu_j] = p$

here $\mu_j \sim \text{Bernoulli}(p)$ $\quad j \in [d]$

we want to minimize to achieve $\hat{w}$

⟹ Empirical Risk (w.r.t $\mu \sim \text{Bernoulli}(p)$)

$$\tilde{R}(w) = \mathbb{E}_\mu \left[\frac{1}{2} \left\| X \cdot \left(\frac{1}{p} D_\mu \right) \cdot w - y \right\|_2^2 \right]$$

Here, $D_\mu = \text{diag}(\mu_1, \ldots, \mu_d)$

$$= \mathbb{E}_\mu \left[\frac{1}{2} \| \underbrace{Xw - y}_{\text{standard LR}} + \underbrace{X \left[\frac{1}{p} D_\mu - I \right] w}_{\boxed{\text{new term}}} \|_2^2 \right]$$

$$= \frac{1}{2} \|Xw - y\|_2^2 + \frac{1}{2} \mathbb{E}_\mu \left[\left\| X \left[\frac{1}{p} D_\mu - I \right] w \right\|_2^2 \right]$$

$$\boxed{\mathbb{E}\left[\frac{1}{p} D_\mu \right] = I}$$

$$+ \mathbb{E}_\mu \left[(Xw - y)^T X \underbrace{\left[\frac{1}{p} D_\mu - I \right]}_{=0} w \right]$$

$$= \frac{1}{2} \|Xw - y\|_2^2 + \frac{1}{2} \mathbb{E}_\mu \left[\left\| X \underbrace{\left[\frac{1}{p} D_\mu - I \right]}_{R(p)} w \right\|_2^2 \right]$$

$$R(p) := \frac{1}{p} D_\mu - I_d$$

$$= \frac{1}{2} \|Xw - y\|_2^2 + \frac{1}{2} w^T \mathbb{E}_\mu \left[R(p)^T X^T X \, R(p) \right] w$$

$X^T X \in \mathbb{R}^{d \times d}$ (covariance matrix)

consider $\left(\mathbb{E}_\mu \left[R(p)^T X^T X R(p) \right] \right)_{ij}$ $\quad \left(R(p)_{ij} = \delta_{ij} \left(\frac{\mu_i}{p} - 1 \right) \right)$

$$= \mathbb{E}_\mu \left[\sum_{k,\ell} \left(\frac{\mu_i}{p} - 1 \right) \delta_{ik} (X^T X)_{k\ell} \left(\frac{\mu_\ell}{p} - 1 \right) \delta_{\ell j} \right]$$

$$= (X^T X)_{ij} \ \mathbb{E}_\mu \left[\left(\frac{\mu_i}{p} - 1 \right) \left(\frac{\mu_j}{p} - 1 \right) \right]$$

$$= (X^T X)_{ij} \cdot \delta_{ij} \underline{\left(\frac{1}{p} - 1 \right)}$$

$$\boxed{\frac{var(\mu_i)}{p^2}}$$

$$= \frac{1}{2} \|Xw - y\|_2^2 + \underline{\frac{1}{2} \left(\frac{1}{p} - 1 \right) \sum_{i=1}^{d} (X^T X)_{ii} \, w_i^2}$$

$\Rightarrow$ Weighted L_2-norm regularization

(the weight is data-driven)

---

3. Batch Normalization (BN)

$$\boxed{\hat{y}(x; w) = x w_1 w_2 \cdots w_L}$$

① Motivation : [1-D Deep LR] $\longrightarrow$ In slide !

② Idea : 1. For Deep NN, Learning Rate depends on the scale of weights & activations at different layers

2. A constant learning rate cannot fullfil our requirements as training goes on !

③ BN Layer :

$H = \{ h^{(1)}, \ldots, h^{(B)} \}$ is a batch of hidden state values

$$\rightarrow BN(h^{(i)}; \gamma, \beta) = \gamma \odot \left(\underline{[h^{(i)} - \mu] \oslash \sigma} \right) + \beta$$

$$= \gamma \odot (\widetilde{h^{(i)}}) + \beta$$

$\widetilde{h^{(i)}} \rightarrow \begin{cases} \text{mean} & 0 \\ \text{std} & 1 \end{cases}$

$\Rightarrow H = \{h^{(1)}, \dots, h^{(B)}\} \rightarrow \begin{cases} \mu & \underline{\text{mean}} \\ \sigma^2 & \underline{\text{variance}} \end{cases}$

$\Rightarrow$ γ & β are <u>learnable parameters</u>

④ <u>Training & Inference</u>

→ Training : simple

→ <u>Inference</u> (Testing) : use moving average (MA)

$\Downarrow$

record recent B samples and update

⑤ Effect (observation & intuition)

→ break the inter-dependency of different layers

→ the scaling is controlled individually with γ & β.

→ It is far beyond naive scaling operation

→ γ & β are learnable

→ gradient can flow !

---

4. <u>Data Augmentation</u>

→ input - output dist : $(x, y) \sim \mu$

→ $(x^{(i)}, y^{(i)}) \sim \mu$ $\quad i = 1, 2, \dots N$ (draw N samples)

→ Empirical Loss

$$R_{emp}(\theta) = \frac{1}{N} \sum_{i=1}^{N} L(f(x^{(i)}, y^{(i)}))$$

→ Expected Loss

$$R_{pop}(\theta) = \mathbb{E}_{(x,y)\sim\mu} [L(f(x), y)]$$

(Statistical)

⇒ Learning Theory tells us:

$$\text{generalization gap} \le \frac{\text{Complexity of Model}}{\#\text{ of samples}}$$

⇒ $\boxed{\text{better generalization}}$

↕

- reduce complexity
- more data

⇒ More Data ← Data Augmentation

↙

Introduce Prior Knowledge to model

(Soft Encode Prior Knowledge)

---

5. Learning Rate Decay

↓

Control the Noise for SGD

→ Choice Criterion for ε_k (k^{-1} decay schedule)

$$\begin{cases} \sum \varepsilon_k = +\infty \\ \sum \varepsilon_k^2 < +\infty \end{cases} \longrightarrow \varepsilon_k = \frac{a}{1+bk} \quad \text{(one choice)}$$

→ In practice, we do not favor this choice

→ one reason: our ultimate goal is not $\min_\theta R_{emp}(\theta)$

but $\boxed{\min_\theta R_{pop}(\theta)}$

→ one choice is: exponential decay schedule

$$\varepsilon_k = \varepsilon_0 \gamma^{\lfloor \frac{k}{k_0} \rfloor}$$

initial learning rate: ε_0

decay ratio: $\gamma \in [\frac{1}{10}, \frac{1}{2}]$

iteration to wait before decay: k_0