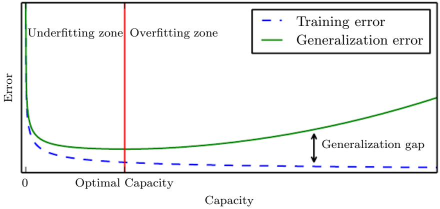
Reading Group on Deep Learning Regularization (Chapter 7)

Gildas Mazo (MISTIS)

17 November 2016

Reminder from Chapter 5



Regularization:

> *any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error*

Limit model capacity

Chapter structure

7.x x$\in \{1,2\}$ penalties
7.3 under-constrained problems
7.4 dataset augmentation
7.5 noise robustness
7.6 semi-supervised learning
7.7 multi-task learning
7.8 early stopping
7.9 parameter tying and sharing
7.10 sparse representations
7.11 bagging and ensemble methods
7.12 dropout
7.13 adversarial training
7.14 tangent distance, prop and manifold tangent classifier

Chapter structure

- 7.x x$\in \{1,2\}$ penalties
- 7.3 under-constrained problems
- 7.4 dataset augmentation
- 7.5 noise robustness
- 7.6 semi-supervised learning
- 7.7 multi-task learning
- 7.8 early stopping
- 7.9 parameter tying and sharing
- 7.10 sparse representations
- 7.11 bagging and ensemble methods
- 7.12 dropout
- 7.13 adversarial training
- 7.14 tangent distance, prop and manifold tangent classifier

Regularizing with penalties

$$\min_{\mathbf{w}} \underbrace{J(\mathbf{w})}_{\text{"old" objective function}} + \underbrace{\alpha}_{\text{tuning parameter}} \underbrace{\Omega(\mathbf{w})}_{\text{penalty}}, \qquad \alpha \geq 0$$

L2 regularization $\Omega(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$
ridge regression, Tickhonov regularization

L1 regularization $\Omega(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_1^2$
Least Absolute Shrinkage and Selection Operator

Link with constrained optimization

$$\min_{\mathbf{w}} J(\mathbf{w}) \text{ such that } \Omega(\mathbf{w}) \leq k, \qquad k > 0$$

Skrinkage effect: details on board

Shrinkage effect

$$\tilde{\mathbf{w}} = \sum_i q_i \underbrace{\frac{\lambda_i}{\lambda_i + \alpha}}_{\text{skrinks}} \underbrace{q_i^\top \mathbf{w}^*}_{\text{coord. of } \mathbf{w}^* \text{ in dir. } q_i}$$

- shrinkage effect $0 < \frac{\lambda_i}{\lambda_i+\alpha} \leq 1$
- $\frac{\lambda_i}{\lambda_i+\alpha}$ increasing in λ_i

λ_i	curvature	shrinkage	variability
small	small	great	small
great	great	small	great

Remember the picture

Early stopping

Optimization problem

$$\min_{\mathbf{w}} J(\mathbf{w})$$

Solve by building a sequence $\mathbf{w}^{(\tau)}$, $\tau = 0, 1, \ldots$:

$$\mathbf{w}^{(\tau)} = \mathbf{w}^{(\tau-1)} - \varepsilon \nabla_{\mathbf{w}} J(\mathbf{w}^{(\tau-1)})$$

Standard GD go on until $\nabla_{\mathbf{w}} J(\mathbf{w}^{(\tau)}) \approx 0$

Early stopping stop early

Approximately equivalent to L2 regularization
"proof" on board

Dropout

Standard neural network lawyer $(l+1)$, i-th hidden unit

$$y_i^{(l+1)} = f\left[\mathbf{w}_i^{(l+1)\top}\mathbf{y}^{(l)} + b_i^{(l+1)}\right]$$

Dropout model lawyer $(l+1)$, i-th hidden unit

$$r_i^{(l)} \sim \text{Ber}(p)$$
$$y_i^{(l+1)} = f\left[\mathbf{w}_i^{(l+1)\top}(\mathbf{r}^{(l)} \star \mathbf{y}^{(l)}) + b_i^{(l+1)}\right]$$

Remove units at random

- Links with noise adding
- Links with bagging

Learning

[As far as I understood... please check!]

- input $\mathbf{x}$, output y, predictor $g_{\mathbf{w},\mu}(\mathbf{x})$
- μ binary vector encodes presence/absence of units

In principle minimize expected loss

$$\min_{\boldsymbol{\mu},\mathbf{w}} E_{\boldsymbol{\mu},\mathbf{x},y}\left[g_{\mathbf{w},\boldsymbol{\mu}}(\mathbf{x}) - y\right]^2$$

In practice minimize estimated loss

$$\min_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n}\left[g_{\mathbf{w},\boldsymbol{\mu}_i}(\mathbf{x}_i) - y_i\right]^2$$

where $\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_n \overset{iid}{\sim} \text{Ber}(p)$

Dataset augmentation

Quote

create fake data and add it to the training set

- easiest for classification
- effective for object recognition
- links with noise adding

Noise robustness

Apply noise to

- inputs
- weights
- outputs

Links with dropout

Semi-supervised learning

Recall the learning context: $y \sim F(y|\mathbf{x})$.
Given a class $\{f_\alpha, \alpha \in A\}$, guess Nature's response

Supervised in principle we do

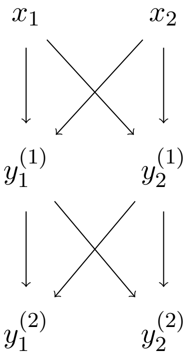
$$\min_\alpha R(\alpha) = \min_\alpha \int L(y, f_\alpha(\mathbf{x}))\, dF(\mathbf{x}, y).$$

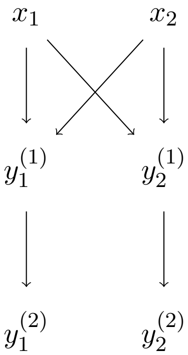
In practice: we have data $\{(\mathbf{x}_i, y_i)\}$ and do

$$\min_\alpha R_n(\alpha) = \min_\alpha \int L(y, f_\alpha(\mathbf{x}))\, dF_n(\mathbf{x}, y)$$

Semisupervised incorpore a priori knowledge about F
(e.g., build a joint distribution function)

Multi-task learning





Parameter Tying and parameter sharing

Learn a first task:

$$\min_{\mathbf{w}^{(A)}} \sum_i [y_i^{(A)} - f_A(\mathbf{w}^{(A)}, \mathbf{x}_i)]^2$$

Learn a second, but "stay close" to the first

$$\min_{\mathbf{w}^{(B)}} \sum_i [y_i^{(B)} - f_B(\mathbf{w}^{(B)}, \mathbf{x}_i)]^2 + \alpha \frac{1}{2} \|\mathbf{w}^{(A)} - \mathbf{w}^{(B)}\|_2^2$$

Sparse representations

Quote

> *place a penalty on the activations of the units in a neural network, encouraging their activation to be sparse*

Example: given input $\mathbf{x}$, find representation $\mathbf{h}$ such that

$$\mathbf{h} = \operatorname*{arg\,min}_{\mathbf{h}:\|\mathbf{h}\|_0 < k} \|\mathbf{x} - W\mathbf{h}\|^2$$

where $\|\mathbf{h}\|_0$ is number of non-zero entries

Bagging and other ensemble methods

Data: $\{\mathbf{x}_i, y_i\}_{i=1}^n$. For $k = 0, 1, \ldots, K$,

1. draw a bootstrap sample $\{\mathbf{x}_i^{(k)}, y_i^{(k)}\}_{i=1}^n$
2. learn $f_n^{(k)}(\cdot)$

Then average the predictions

$$f_n^{\mathsf{bag}}(\mathbf{x}) = \frac{1}{K} \sum_{k=1}^{K} f_n^{(k)}(\mathbf{x})$$

More generally, the idea is to combine several models