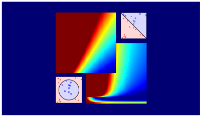
Machine Learning Techniques
(機器學習技巧)



Lecture 12: Deep Learning

Hsuan-Tien Lin (林軒田)

htlin@csie.ntu.edu.tw

Department of Computer Science
& Information Engineering

National Taiwan University
(國立台灣大學資訊工程系)

Agenda

Lecture 12: Deep Learning

- Optimization and Overfitting
- Auto Encoder
- Principle Component Analysis
- Denoising Auto Encoder
- Deep Neural Network

Error Function of Neural Network

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\left(y_n - \theta\left(\cdots\theta\left(\sum_j w_{jk}^{(2)}\cdot\theta\left(\sum_i w_{ij}^{(1)}x_i\right)\right)\right)\right)^2$$

- generally **non-convex** when multiple hidden layers
  - not easy to reach **global minimum**
  - GD/SGD with **backprop** only gives **local minimum**
- different initial $\mathbf{w}_0 \Longrightarrow$ different **local minimum**
  - somewhat '**sensitive**' to initial weights
  - **large weights** $\Longrightarrow$ **saturate** (small gradient)
  - advice: try **some random** & **small** ones

neural network (NNet):
difficult to optimize, but **practically works**

VC Dimension of Neural Networks

roughly, with θ**-like transfer functions**:
$d_{\text{VC}} = O(D \log D)$ where $D = \#$ of weights

- can **implement 'anything'** if enough neurons (D large)
—no need for **many layers**?
- can **overfit** if too many neurons

NNet: **watch out for overfitting!**

Regularization for Neural Network

basic choice:

old friend weight-decay (L2) regularizer $\Omega(\mathbf{w}) = \sum \left(w_{ij}^{(\ell)} \right)^2$

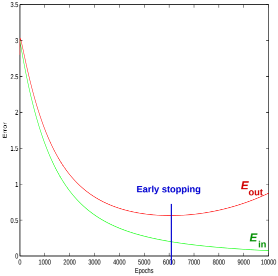
- 'shrink' weights:
  large weight $\rightarrow$ large shrink; small weight $\rightarrow$ small shrink
- want $w_{ij}^{(\ell)} = 0$ (sparse) to effectively **decrease** d_{VC}
  - L1 regularizer: $\sum \left| w_{ij}^{(\ell)} \right|$, but **not differentiable**
  - weight-elimination ('scaled' L2) regularizer:
    large weight $\rightarrow$ median shrink; small weight $\rightarrow$ median shrink

weight-elimination regularizer: $\sum \frac{\left(w_{ij}^{(\ell)} \right)^2}{\beta^2 + \left(w_{ij}^{(\ell)} \right)^2}$

Yet Another Regularization: Early Stopping

GD/SGD (backprop) visits
more weight combinations as t increases

- smaller t effectively decrease d_{VC}
- better 'stop in the middle': **early stopping**



when to stop? **validation!**

Fun Time

Learning the Identity Function

identity function: $\mathbf{f}(\mathbf{x}) = \mathbf{x}$

- a vector function composed of $f_i(\mathbf{x}) = x_i$
- learning each f_i: regression with data
  $(\mathbf{x}_1, y_1 = x_{1,i}), (\mathbf{x}_2, y_2 = x_{2,i}), \ldots, (\mathbf{x}_N, y_N = x_{N,i})$
- learning $\mathbf{f}$: learning f_i jointly with data
  $(\mathbf{x}_1, \mathbf{y}_1 = \mathbf{x}_1), (\mathbf{x}_2, \mathbf{y}_2 = \mathbf{x}_2), \ldots, (\mathbf{x}_N, \mathbf{y}_N = \mathbf{x}_N)$

but wait, why **learning** something
known & **easily implemented**? **:-)**

Why Learning Identity Function

if $\mathbf{g}(\mathbf{x}) \approx \mathbf{f}(\mathbf{x})$ using some **hidden** structures on the **observed data $\mathbf{x}_n$**

- for unsupervised learning:
  - density estimation: larger (structure match) when $\mathbf{g}(\mathbf{x}) \approx \mathbf{x}$ better
  - outlier detection: those $\mathbf{x}$ where $\mathbf{g}(\mathbf{x}) \not\approx \mathbf{x}$

  —learning **'typical' representation** of data
- for supervised learning:
  - **hidden structure**: essence of $\mathbf{x}$ that can be used as $\boldsymbol{\Phi}(\mathbf{x})$

  —learning **'informative' representation** of data

auto-encoder:
NNet for learning identity function

Simple Auto-Encoder

simple auto-encoder: a d-$\tilde{d}$-d NNet

- d outputs: backprop easily applies
- $\tilde{d} < d$: **compressed** representation;
  $\tilde{d} \geq d$: **[over]-complete** representation
- data: $(\mathbf{x}_1, \mathbf{y}_1 = \mathbf{x}_1), (\mathbf{x}_2, \mathbf{y}_2 = \mathbf{x}_2), \ldots, (\mathbf{x}_N, \mathbf{y}_N = \mathbf{x}_N)$
  —often categorized as **unsupervised learning technique**
- if $\mathbf{x}$ contain binary bits,
  - naïve solution exists (but **unwanted**) when **[over]-complete**
  - **regularized** weights needed in general
- sometimes constrain $w_{ij}^{(1)} = w_{ji}^{(2)}$ as '**regularization**'
  —more **sophisticated** in calculating gradient

auto-encoder for **representation** learning:
outputs of **hidden neurons** serve as $\mathbf{\Phi}(\mathbf{x})$

Fun Time

Linear Auto-Encoder Hypothesis

$$h_k(\mathbf{x}) = \theta\left(\sum_j w_{jk}^{(2)} \cdot \theta\left(\sum_i w_{ij}^{(1)} x_i\right)\right)$$

consider three special conditions:

- constrain $w_{ij}^{(1)} = w_{ji}^{(2)} = w_{ij}$ as '**regularization**'
  —let $\mathrm{W} = [w_{ij}]$ of size $d \times \tilde{d}$
- θ does nothing (like linear regression)
- $\tilde{d} < d$

linear auto-encoder hypothesis:

$$\mathbf{h}(\mathbf{x}) = \mathbf{x}^T \mathrm{W} \mathrm{W}^T$$

Linear Auto-Encoder Error Function

$$\min_{\mathrm{W}} \quad E_{\text{in}}(\mathrm{W}) = \frac{1}{N} \left\| \mathrm{X} - \mathrm{X}\mathrm{W}\mathrm{W}^T \right\|_F^2$$

let $\mathrm{W}\mathrm{W}^T = \mathrm{V}\Lambda\mathrm{V}^T$ such that $\mathrm{V}^T\mathrm{V} = \mathrm{I}$ and Λ a diagonal matrix of rank at most $\tilde{d}$ (eigenvalue decomposition)

$$\begin{aligned}
& \left\| \mathrm{X} - \mathrm{X}\mathrm{V}\Lambda\mathrm{V}^T \right\|_F^2 \\
= \quad & \text{trace}\left(\left(\mathrm{X} - \mathrm{X}\mathrm{V}\Lambda\mathrm{V}^T\right)^T \left(\mathrm{X} - \mathrm{X}\mathrm{V}\Lambda\mathrm{V}^T\right) \right) \\
= \quad & \text{trace}\left(\mathrm{X}^T\mathrm{X} - \mathrm{X}^T\mathrm{X}\mathrm{V}\Lambda\mathrm{V}^T - \mathrm{V}\Lambda\mathrm{V}^T\mathrm{X}^T\mathrm{X} + \mathrm{V}\Lambda\mathrm{V}^T\mathrm{X}^T\mathrm{X}\mathrm{V}\Lambda\mathrm{V}^T \right) \\
= \quad & \text{trace}\left(\mathrm{X}^T\mathrm{X} - \Lambda\mathrm{V}^T\mathrm{X}^T\mathrm{X}\mathrm{V} - \Lambda\mathrm{V}^T\mathrm{X}^T\mathrm{X}\mathrm{V} + \Lambda\mathrm{V}^T\mathrm{X}^T\mathrm{X}\mathrm{V}\Lambda\mathrm{V}^T\mathrm{V} \right) \\
= \quad & \text{trace}\left(\mathrm{V}^T\mathrm{X}^T\mathrm{X}\mathrm{V} - \Lambda\mathrm{V}^T\mathrm{X}^T\mathrm{X}\mathrm{V} - \Lambda\mathrm{V}^T\mathrm{X}^T\mathrm{X}\mathrm{V} + \Lambda^2\mathrm{V}^T\mathrm{X}^T\mathrm{X}\mathrm{V} \right) \\
= \quad & \text{trace}\left((\mathrm{I} - \Lambda)^2\, \mathrm{V}^T\mathrm{X}^T\mathrm{X}\mathrm{V} \right)
\end{aligned}$$

Linear Auto-Encoder Algorithm

$$\min_{\mathrm{V},\Lambda} \quad \text{trace}\left((\mathrm{I} - \Lambda)^2 \mathrm{V}^T \mathrm{X}^T \mathrm{X} \mathrm{V} \right)$$

- optimal rank-$\tilde{d}$ Λ contains $\tilde{d}$ '1' and $d - \tilde{d}$ '0'
- let $\mathrm{X}^T\mathrm{X} = \mathrm{U}\Sigma\mathrm{U}^T$ (eigenvalue decomposition), $\mathrm{V} = \mathrm{U}$ with (smallest $\sigma_i \Longleftrightarrow \lambda_j = 1$) is optimal
- so **optimal column vectors $\mathbf{w}_j = \mathbf{v}_j$ = top eigen vectors of $\mathrm{X}^T\mathrm{X}$**

optimal linear auto-encoding $\equiv$ **principal component analysis** (PCA) with $\mathbf{w}_j$ being **principal components** of unshifted data

Fun Time

Simple Auto-Encoder Revisited

simple auto-encoder: a d-$\tilde{d}$-d NNet

- want: **hidden structure** to capture essence of $\mathbf{x}$
- naïve solution exists (but **unwanted**) when **[over]-complete**
- **regularized** weights needed in general

regularization towards
more **robust hidden structure**?

Idea of Denoising Auto-Encoder

robust hidden structure should allow $g(\tilde{\mathbf{x}}) \approx \mathbf{x}$
even when $\tilde{\mathbf{x}}$ slightly different from $\mathbf{x}$

- denoising auto-encoder: run auto-encoder with data $(\tilde{\mathbf{x}}_1, \mathbf{y}_1 = \mathbf{x}_1), (\tilde{\mathbf{x}}_2, \mathbf{y}_2 = \mathbf{x}_2), \ldots, (\tilde{\mathbf{x}}_N, \mathbf{y}_N = \mathbf{x}_N)$, where $\tilde{\mathbf{x}}_n = \mathbf{x}_n +$ **artificial noise**
- PCA auto-encoder + **Gaussian noise**:

$$\min_{\mathrm{W}} \qquad E_{\text{in}}(\mathrm{W}) = \frac{1}{N} \left\| \mathrm{X} - (\mathrm{X} + \textbf{noise})\, \mathrm{W}\mathrm{W}^T \right\|_F^2$$

—simply L2-**regularized** PCA

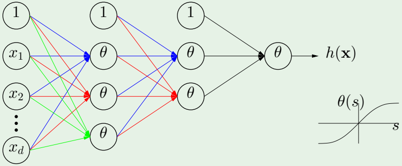
artificial noise as **regularization**!
—practically also useful for other types of NNet

Fun Time

Final remark: hidden layers

learned nonlinear transform

interpretation?



Shallow versus Deep Structures

shallow: few hidden layers; deep: many hidden layers

Shallow

- efficient
- powerful if enough neurons

Deep

- challenging to train
- needing more structural (model) decisions
- ‘meaningful’?

deep structure (deep learning)
re-gain attention recently

Key Techniques behind Deep Learning

- (usually) **unsupervised pre-training** between hidden layers, such as simple/denoising auto-encoder
  —viewing hidden layers as **'condensing'** low-level representation to high-level one
- **fine-tune with backprop** after initializing with those 'good' weights
  —because **direct backprop may get stuck more easily**
- speed-up: better optimization algorithms, and faster **GPU**
- generalization issue less serious with **big (enough) data**

currently very useful
for **vision and speech recognition**

Fun Time

Summary

Lecture 12: Deep Learning

- Optimization and Overfitting
- Auto Encoder
- Principle Component Analysis
- Denoising Auto Encoder
- Deep Neural Network