A Statistical Overview of Neural Networks

Speaker:Ming-Chun Wu

Institute of Statistical Science
Academia Sinica

October 7, 2016

OUTLINE

- An Overview of Supervised Learning
- Neural Networks
  - Logistic Regression
  - Multilayer Perceptron
  - Deep Neural Networks
- Autoencoder: Unsupervised Neural Network
- Examples

~~Un~~supervised Learning

From the viewpoint of statistical decision theory

- Goal: Infer unknown y from observed data x

$$x \longrightarrow \boxed{\text{Decision Rule: } h} \longrightarrow y=?$$

- How to derive good decision rules?
  - Zero-one loss
  - Statistical decision theory:

$$\min_h \int \int \mathbf{1}[y \neq h(x)] p(x, y) dx dy \tag{1}$$

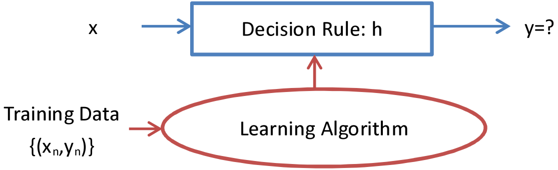
$$= \min_h \int \left(\int \mathbf{1}[y \neq h(x)] p(y|x) dy \right) p(x) dx \tag{2}$$

  - Bayesian MAP decision rule: $h(x) = \arg\max_y p(y|x)$

~~Un~~supervised Learning

Optimal decision rule: $h(x) = \arg\max_{y} p(y|x)$

- Unknown $p(y|x)$ in practical situations!
- Supervised learning: estimate $p(y|x)$ from training data

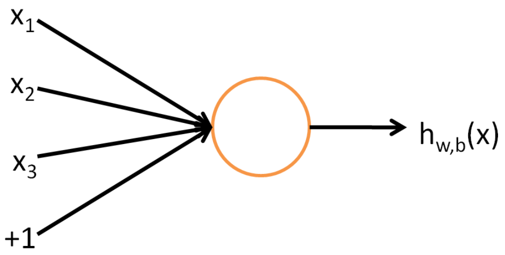


- Goal: design good learning algorithm!

Linear Classification

Consider simple binary classification $y \in \{-1, 1\}$

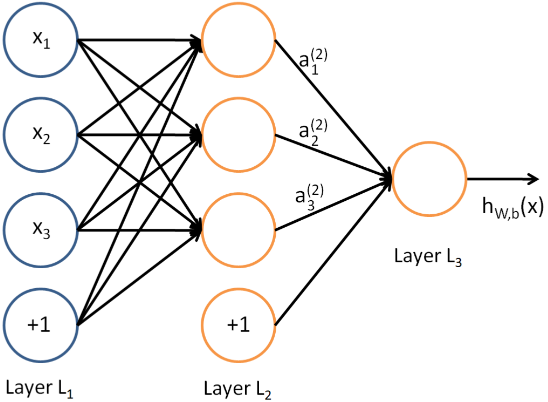
- Linear regression: $y = (w^T x + b) + n$ and n is zero mean noise
- Logistic regression: assume $p(y = 1|x) = sigmoid(w^T x + b)$
- Decision rule: $h_{w,b}(x) > \eta$? where η is the threshold
  - Linear regression: $h_{w,b}(x) = sign(w^T x + b)$ and $\eta = 0$
  - Logistic regression : $h_{w,b}(x) = sigmoid(w^T x + b)$ and $\eta = 0.5$
  - General linear classifier: $h_{w,b}(x) = s(w^T x + b)$



Neural Network

Multilayer perceptron

- Limited power of linear classifier
- How to implement nonlinear decision boundary?
- Combination of linear classifiers



Training Neural Networks

- Training data $\mathcal{D} = \{(x_n, y_n)\}$
- Loss function $l(y, h_w(x))$
- Minimize $L(\mathcal{D}) = \sum l(y_n, h_w(x_n))$
- $\nabla L(\mathcal{D}) = \sum \nabla l(y_n, h_w(x_n))$
- Compute gradient via back propagation
- Gradient descent (GD): $w \leftarrow w - \alpha \sum \nabla l(y_n, h_w(x_n))$
- Stochastic gradient descent (SGD):
  - Use a small batch of data to compute gradient
  - $w \leftarrow w - \alpha \sum_{subset} \nabla l(y_n, h_w(x_n))$

Deep Neural Network

With multiple layers

- Very powerful since high model complexity
- Fit training data very well

But

- Over-fitting
  - Bias-variance trade off

$$E_{\mathcal{D}}(y - h(x))^2 = Var_{\mathcal{D}}(h(x)) + Bias^2(h(x)) \tag{3}$$

  - VC bound: with probability $1 - \delta$ and VC dimension d

$$Error_{test} \leq Error_{train} + \sqrt{\frac{8}{n} \ln \left(\frac{4(2n)^d}{\delta} \right)} \tag{4}$$

- Need many training data
- Hard optimization in training step

Deep Neural Network

For over-fitting: adding regularization

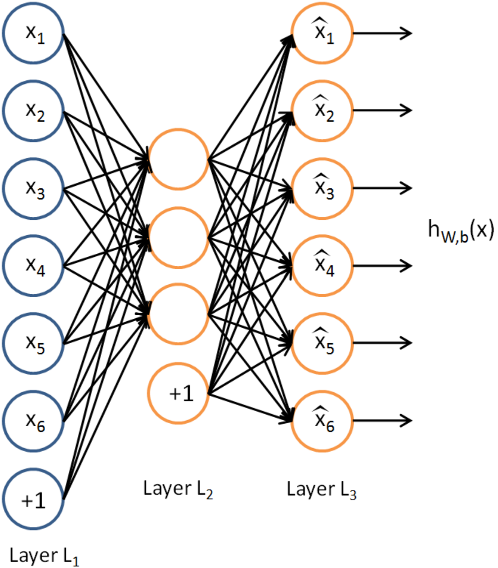
- l_1 or l_2 penalty
- Early-stopping in GD/SGD
- Validation

For optimization : pre-training then fine-tuning

- Deep belief network: Using restricted Boltzmann machine
- Stacked autoencoder

Autoencoder

Unsupervised learning with data $\mathcal{D} = \{(x_n, x_n)\}$



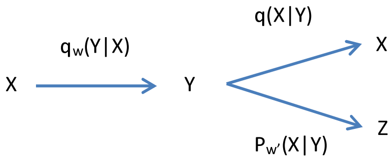
Autoencoder

Why autoencoder?

- Dimension reduction, lossy compression: PCA = linear autoencoder
- Feature extraction

Information theoretical interpretation:

- Compress/encode X as Y
- Goal: Y contain as much information as X



Basic Information Theory

- Entropy $H(X) = E_{p(x)}[-\log p(x)]$
- Conditional entropy $H(X|Y) = E_{p(x,y)}[-logp(x|y)]$
- Mutual Information $\mathrm{I}(X;Y) = H(X) - H(X|Y)$
- KL divergence $D(q(x)||p(x)) = E_{q(x)}[-logp(x)] - H(q(x))$
- Cross entropy $H(q(x)||p(x)) = E_{q(x)}[-logp(x)]$

From Mutual Information To Autoencoder

- Marginal distribution of X: $q(x)$

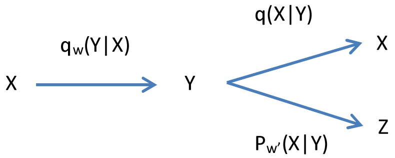
$$\arg\max_{w} \mathrm{I}(X;Y) = \arg\max_{w} H(X) - H(X|Y) \tag{5}$$

$$= \arg\max_{w} E_{q_w(y|x)q(x)}[\log q(x|y)] \tag{6}$$

- Estimate unknown $q(x|y)$ with $p_{w'}(x|y)$
- $\forall w', D(q(x|y)||p_{w'}(x|y)) \geq 0$ we have

$$E_{q_w(y|x)q(x)}[\log q(x|y)] \geq \max_{w'} E_{q_w(y|x)q(x)}[\log p_{w'}(x|y)] \tag{7}$$

- $\max_{w} \mathrm{I}(X;Y) \geq \max_{w,w'} E_{q_w(y|x)q(x)}[\log p_{w'}(x|y)]$

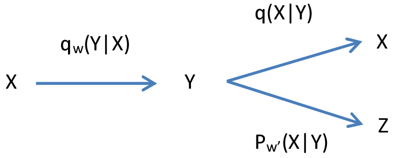


From Mutual Information To Autoencoder

- $\max_{w,w'} E_{q_w(y|x)q(x)}[\log p_{w'}(x|y)]$
- Encoder $y = f_w(x)$
- Autoencoder: minimize cross entropy loss

$$\min_{w,w'} E_{q(x)}[-\log p_{w'}(x|y = f_w(x))] \tag{8}$$

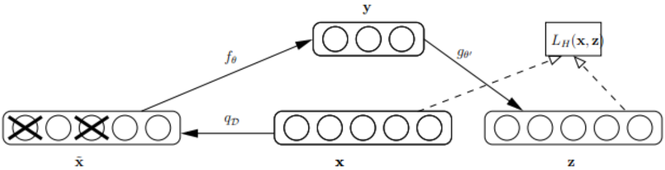
- Replace $q(x)$ with empirical distribution
- If $p_{w'}(x|y) \sim N(g_{w'}(y), \sigma^2)$, loss $= \|x - g_{w'}(f_w(x))\|^2$
- If $p_{w'}(x|y) \sim Bernoulli(g_{w'}(y))$,
  loss $= -x \log gf(x) + (1 - x) \log(1 - gf(x))$



Stacked Denoising Autoencoder

Denoising autoencoder

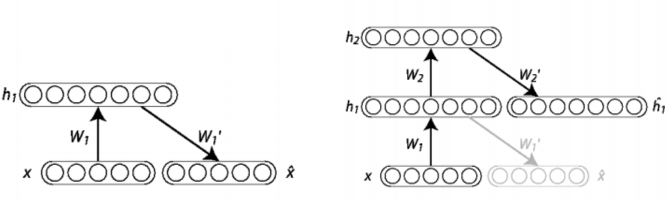
- Training with noisy data $\mathcal{D} = \{(\tilde{x}_n, x_n)\}$
- Noisy data $\tilde{x}_n$
- Additive noise, mask, etc.
- Robust feature extraction



Stacked Denoising Autoencoder

Deep learning with autoencoder

- Pre-training: stacked autoencoder with unlabeled data



- Fine-tuning with labeled data

Examples

Example: Convolutional Autoencoder

Figure 1: Upper: original images, lower: reconstructions from convolution autoencoder

Example: Denoising Autoencoder

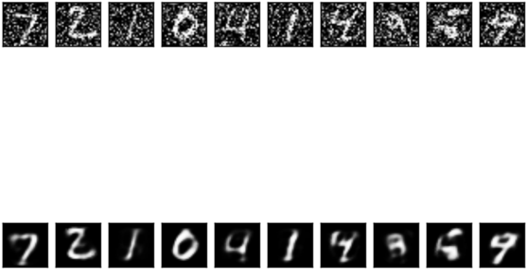
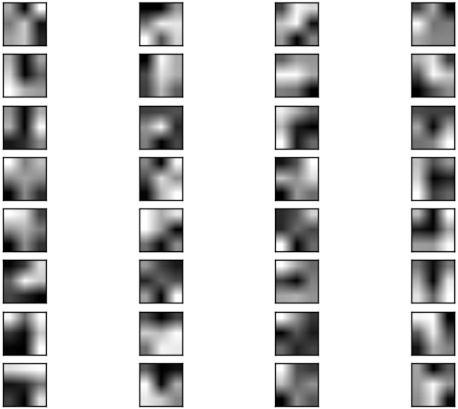


Figure 2: Upper: noisy images, lower: reconstructions from denoising autoencoder

Example: Features of Convolutional Autoencoder



QUESTION?

Thank You!