Bioinformatics Chapter 5.

Neural Networks: The Theory

Outline

- Introduction to Neural Networks
- MLPs as Universal Approximators
- Bayesian Learning: Priors and Likelihoods
- Backpropagation Learning as MLE
- Other Statistical Learning Models
- Summary

1. Introduction

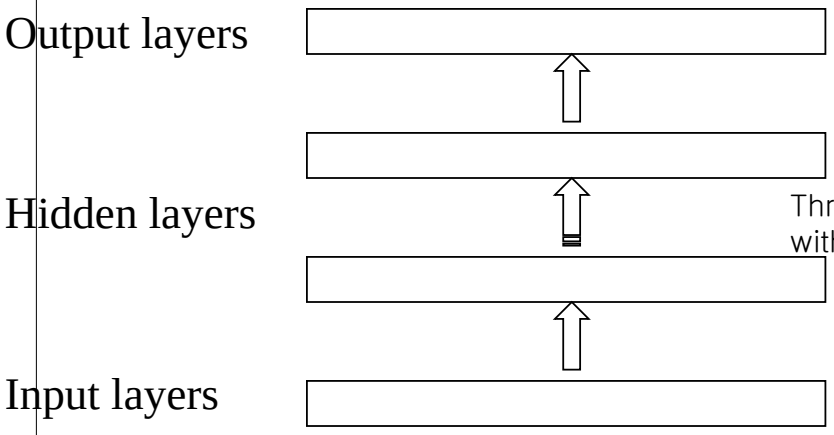
- Neural Networks
  - ♦ Developed with the goal of modeling information processing and learning in the brain.
  - ♦ Applied to a number of practical applications in various fields, including computational molecular biology.
  - ♦ Can be viewed as a broad class of parameterized graphical models consisting of networks with interconnected units.
- *Remark: For a more general introduction to NNs, see the NN tutorial notes on the course web site at* http://scai.snu.ac.kr/Courses

1. Introduction

- Classification
  - ♦ In terms of architecture
    - ■ Recurrent network
      - – Contains directed loops.
    - ■ Feed-forward network
      - – Devoid of directed loops.
    - ■ Layered network
      - – The units are partitioned into classes, also called *layer*s.
  - ♦ In most current applications of NNs to molecular biology, the architecture used are layered feed-forward architectures (a.k.a. multilayer perceptrons or MLPs).

1. Introduction

- Architecture of MLP
  - Layered feed-forward architecture
  - The number of weight layers is often referred to as ‘depth’ of a network.



1. Introduction

- Input/Output of a unit
  - ♦ Input
    - Weighted sum of incoming outputs from previous layer.

$$x_i = \sum w_{ij} y_j + w_i$$

Notation: x_i :=: net_i in NN tutorial

  - ♦ Output
    - Output from transfer function f_i

$$y_i = f_i(x_i) = f_i\left(\sum w_{ij} y_j + w_i\right)$$

1. Introduction

- Transfer function of a unit
  - ♦ *Threshold function*
    - ■ Simulates a binary decision
    - ■ Discontinuous

$$f(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise,} \end{cases}$$

  - ♦ *Logistic (sigmoid) transfer function*
    - ■ Continuous and differentiable replacement of threshold gate

$$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$$

1. Introduction

  - ♦ *Normalized exponential unit (softmax)*
    - ■ Used to compute the probability of an event with *n* possible outcomes.

$$y_i = \frac{e^{-x_i}}{\sum_{k=1}^{n} e^{-x_k}}$$

    - ■ When *n*=2, the normalized exponential is equivalent to a logistic function

$$y_1 = \frac{e^{-x_1}}{e^{-x_1} + e^{-x_2}} = \frac{1}{1 + e^{-(x_2 - x_1)}}$$

1. Introduction

- Regression and classification
  - ♦ Regression
    - ■ The goal is to approximate or fit a given surface.
  - ♦ Classification
    - ■ The goal is to able to classify a given input into a relatively small number of classes
- Data types
  - ♦ Training data: for building (learning) the models
  - ♦ Validation data: for stopping criterion
  - ♦ Test data: for evaluation of generalization accuracy

2. Multilayer Perceptrons as Universal Approximators

- Any reasonable real function can be approximated to any degree of precision by a three-layer network (with one hidden layer) as long as the hidden layer can be arbitrarily large.
  - ♦ Input layer
  - ♦ A hidden layer of sigmoidal units
  - ♦ One layer of linear output units
- Many different mathematical variations and proofs.
- Example:
  - ♦ Any Boolean function can be realized by ORing (output layer) of ANDs (hidden layer).

Universal Approximation Properties

• Consider the approximation of $y = f(x)$ where both x and y are one-dimensional.

• Since f is continuous, there exists an integer n such that

$$|x_2 - x_1| \le \frac{1}{n} \Rightarrow |f(x_2) - f(x_1)| \le \varepsilon$$

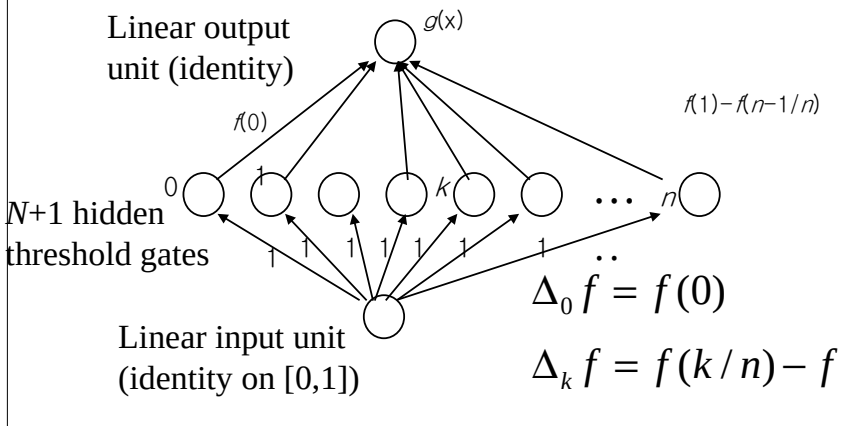
• It is sufficient to approximate f with a function g such that

$$g(0) = f(0)$$

$$g(x) = f(k/n) \text{ for } x \text{ in } ((k-1)/n, k/n]$$

Architecture for Universal Approximation

• The function g can be realized exactly by an NN with one input unit, $n+1$ hidden threshold gate units, and one linear output unit.



$$\Delta_0 f = f(0)$$

$$\Delta_k f = f(k/n) - f(k-1/n)$$

$$g(x) = f(0) + \sum_{j=1}^{K} \Delta_j f = f(0) + \sum_{j=1}^{K} f(k/n) - f(k-1/n)$$

3. Bayesian Learning: Priors and Likelihoods

- Given a data set $D=\{(d_1,t_1),(d_2,t_2),\ldots,(d_K,t_K)\}$ which consists of a set of independent input-output pairs (d_i,t_i)
  - ♦ Likelihood of a data
    $$P((d_i,t_i)\,|\,\mathbf{w}) = P(d_i\,|\,\mathbf{w})P(t_i\,|\,d_i,\mathbf{w}) = P(d_i)P(t_i\,|\,d_i,\mathbf{w})$$
  - ♦ Likelihood of D
    $$P(D\,|\,\mathbf{w}) = \prod_{i=1}^{K} P(t_i\,|\,d_i,\mathbf{w})$$
  - ♦ Posterior
    $$P(\mathbf{w}\,|\,D) = \frac{P(D\,|\,\mathbf{w})P(\mathbf{w})}{P(D)} = \frac{\prod_{i=1}^{K} P(t_i\,|\,d_i,\mathbf{w})P(\mathbf{w})}{P(D)}$$
    $$\frac{\text{Likelihood} * \text{Prior}}{\text{Evidence}}$$

- Negative log-posterior

$$-\log P(\mathbf{w}\,|\,D) = -\log \frac{\prod_{i=1}^{K} P(t_i\,|\,d_i,\mathbf{w})P(\mathbf{w})}{P(D)}$$

$$= -\sum_{i=1}^{K} \log P(t_i\,|\,d_i,\mathbf{w}) - \sum_{i=1}^{K} \log P(d_i) - \log P(\mathbf{w}) + \log P(D)$$

$$= -\sum_{i=1}^{K} \log P(t_i\,|\,d_i,\mathbf{w}) - \log P(\mathbf{w}) + C$$

$$= -(\log \text{likelihood} + \log \text{prior}) + C$$

MAP and MLE

- Bayesian approach (full Bayesian):

$$P(\mathbf{w} \mid D) = \frac{P(D \mid \mathbf{w})P(\mathbf{w})}{P(D)}$$

- Maximum a posteriori (MAP) approach (partially Bayesian):

$$\mathbf{w}_{MAP} = \arg\max_{\mathbf{w} \in W} P(\mathbf{w} \mid D) = \arg\max_{\mathbf{w} \in W} P(D \mid \mathbf{w})P(\mathbf{w})$$
$$= \arg\min_{\mathbf{w} \in W} -\log P(\mathbf{w} \mid D)$$

- Maximum likelihood estimation (MLE):

$$\mathbf{w}_{MLE} = \arg\max_{\mathbf{w} \in W} P(D \mid \mathbf{w})$$
$$= \arg\min_{\mathbf{w} \in W} -\log P(D \mid \mathbf{w})$$

4. Backpropagation Learning as MLE

- Objective (Error) Function

$$E(\mathbf{w}) \equiv \frac{1}{2} \sum_{i=1}^{K} (t_i - f(d_i))^2$$

- Learning rule

$$w_i \leftarrow w_i + \Delta w_i \quad , \quad \Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

- Backpropagation
  - Network weights are updated sequentially, from the output layer back to the input layer.
  - Propagate an error signal backward along the neural network connections.

- On-line version

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_i}\frac{\partial y_i}{\partial w_{ij}} = \frac{\partial E}{\partial y_i} f_i'(x_i) y_i$$

  - The gradient-descent learning equation

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = -\eta \delta_i y_j$$

  - Backpropagated error can be computed recursively

$$\frac{\partial E}{\partial y_i} = \sum_{k \in N^+(i)} \frac{\partial E}{\partial y_k} f_k'(x_k) w_{ki}$$

Backpropagation Performs (Approximately) ML Estimation!

$$\mathbf{w}_{MLE} = \arg\max_{\mathbf{w} \in W} P(D \mid \mathbf{w})$$

$$= \arg\min_{\mathbf{w} \in W} -\log P(D \mid \mathbf{w})$$ Gaussian assumption of noise

$$= \arg\min_{\mathbf{w} \in W} -\sum_{i=1}^{K} \log P(t_i \mid d_i, \mathbf{w}) \quad P(t_i \mid d_i, \mathbf{w}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(t_i - f(d_i))^2}{2\sigma^2}\right)$$

$$\approx \arg\min_{\mathbf{w} \in W} \frac{1}{2}\sum_{i=1}^{K} (t_i - f(d_i))^2$$ The error function for BP

$$= \arg\min_{\mathbf{w} \in W} E(\mathbf{w})$$

Bayesian Learning of MLPs: Priors

- Zero mean Gaussian prior
  - The most natural and widely used priors for neural network parameters. $P(\mathbf{w} \mid D) = \frac{P(D \mid \mathbf{w})P(\mathbf{w})}{P(D)}$ $P(\mathbf{w}) \propto \exp(-\frac{\| \mathbf{w} \|^2}{2\sigma^2})$
  - Contribution to the negative log-posterior
    - $\| \mathbf{w} \|^2 / 2\sigma^2$
    - Regularization factor that penalizes large weights.
    - In gradient descent learning, this adds *weight decay*,
- *Weight sharing*
  - Prior obtained when different groups of units are assumed to have identical incoming connection weights.
  - Useful in problems characterized by some form of translational invariance.

Other Priors for Bayesian MLPs

- Other priors
  - Laplace approximation
    - Used to determine optimal hyperparameters.
  - Monte Carlo methods
    - Used for the integration of priors and Bayesian learning in MLPs.
- Advantages of Bayesian learning
  - Automatic determination of regularization parameters without the need for validation set.
  - Avoidance of overfitting when using large networks.
  - Quantification of prediction uncertainty.

5. Other Statistical Learning Models

- Gaussian regression
- Binomial classification
- Multinomial classification
- General exponential family cases

5.1 Gaussian Regression

- Gaussian probabilistic model for regression
  - $P(t \mid d, \mathbf{w}) = P(t \mid y(d), \mathbf{w}) = P(t \mid y)$ is Gaussian, with mean $y = y(d)$
  - If covariance matrix is diagonal and there are output units indexed by j

$$P(t \mid d, \mathbf{w}) = \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_j} e^{\frac{(t_j - y_j)^2}{2\sigma_j^2}}$$

  - If standard deviations are constant $\sigma_j = \sigma$
  - Negative log-likelihood

$$E = \sum_j \left(\frac{(t_j - y_j)^2}{2\sigma^2} - \frac{1}{2}\log 2\pi - \log \sigma \right)$$

  - The derivative of the negative log-likelihood (assuming the output transfer function f_i is the identity)

$$\frac{\partial E}{\partial y_j} = \frac{\partial E}{\partial x_j} = -\frac{t_j - y_j}{\sigma_j} = -\frac{t_j - y_j}{\sigma} \qquad y_i = f_i(x_i) = f_i\left(\sum w_{ij} y_j + w_i\right)$$

5.2 Binomial Classification

- Definition
  - Classification problem with only two classes. The target output is t = 0 or 1.
- Formulation
  - Probability: $y = y(d) = P(y = t) = P(t \mid d, \mathbf{w}) = y^t (1-y)^{(1-t)}$
  - Negative log-likelihood

$$E = -\log P(t \mid d, \mathbf{w}) = -t \log y - (1-t)\log(1-y)$$

  - Derivative $\frac{\partial E}{\partial y} = -\frac{t-y}{y(1-y)}$ Relative entropy
    - If the output function is a sigmoid function, i.e., $y = f(x) = \frac{1}{1+e^{-x}}$

$$\frac{\partial E}{\partial x} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial x} = -\frac{t-y}{y(1-y)} y(1-y) = -(t-y)$$

Remarks

- The likelihood error function is essentially the relative entropy between the predicted distribution and the target distribution.
- The derivative of E w.r.t. the total input activity x into the output unit, for each example, has the simple expression, $-(t-y)$.

5.3 Multinomial Classification

- Definition
  - Classification task with *n* possible classes.
- Formulation
  - For a given input, the target output *t* is a vector with a single 1 and *n*-1 zeros.
  - The most simple probabilistic model is a multinomial model.
  - Likelihood

$$P(t \mid d, \mathbf{w}) = \prod_{j=1}^{n} y_j^{t_j}$$

  - Negative log-likelihood

$$E = -\log P(t \mid d, \mathbf{w}) = -\sum_{j=1}^{n} t_j \log y_j$$

Multinomial Classification (cont'd)

- Derivative

$$\frac{\partial E}{\partial y_j} = -\frac{t_j}{y_j}$$

  - If the output layer consists of a set of normalized exponentials,

$$\frac{\partial E}{\partial x_j} = -(t_j - y_j)$$

  - The output transfer function should be normalized exponentials.
  - The likelihood error function is essentially the relative entropy between the predicted distribution and the target distribution.
  - The derivative of E w.r.t., the total input activity x into the output layer, for each example, has the simple expression, $-(t_j - y_j)$

5.4 General Exponential Family Cases

- The exponential family contains many of the common distributions such as
  - ♦ Gaussian, gamma, binomial, multinomial, exponential, beta, Poisson, negative binomial
- One can construct suitable transfer functions for the output layer, as well as suitable error functions to measure network performance.

Summary

- Learning principles in multilayer neural networks were examined in the framework of Bayesian inference.
- Backpropagation learning can be seen as maximum likelihood estimation (MLE).
- Bayesian learning for MLPs was presented.
- Comparisons have been made between MLPs and standard statistical learning methods.