Pattern Information Processing: Neural Networks

Masashi Sugiyama
(Department of Computer Science)

Contact: W8E-505
sugi@cs.titech.ac.jp
http://sugiyama-www.cs.titech.ac.jp/~sugi/

Linear/Non-Linear Models

- Linear model: $f_{\boldsymbol{\alpha}}(\boldsymbol{x})$ is linear with respect to $\boldsymbol{\alpha}$

  (Note: not necessarily linear with respect to $\boldsymbol{x}$)

$$f_{\boldsymbol{\alpha}}(\boldsymbol{x}) = \sum_{i=1}^{b} \alpha_i \varphi_i(\boldsymbol{x})$$

- Non-linear model: Otherwise

Today's Plan

- Neural networks
- Least-squares in neural networks: Error back-propagation algorithm

Non-Linear Models

- A popular choice: A hierarchical model

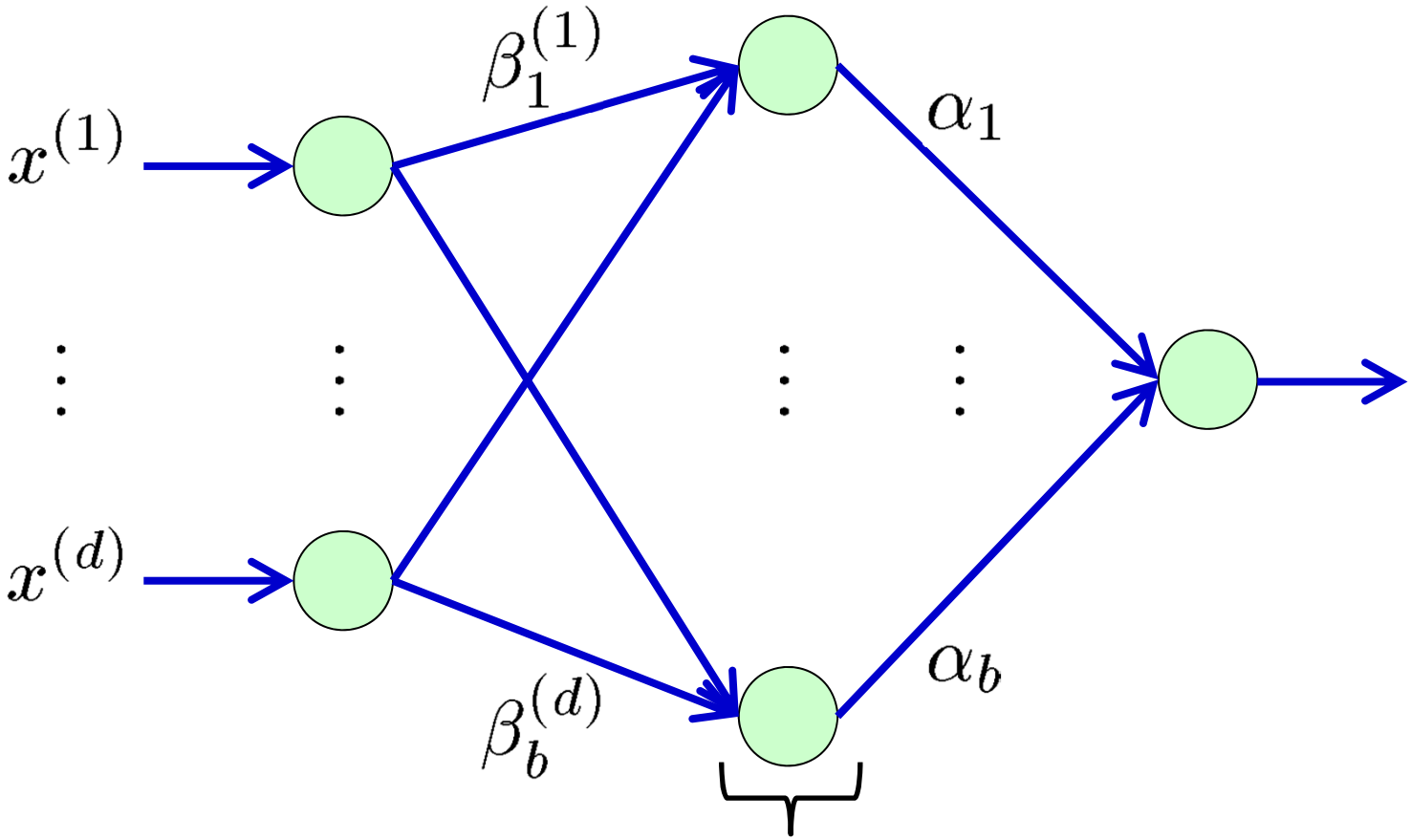
$$f_{\boldsymbol{w}}(\boldsymbol{x}) = \sum_{i=1}^{b} \alpha_i \varphi(\boldsymbol{x}; \boldsymbol{\beta}_i)$$

$$\boldsymbol{w} = (\boldsymbol{\alpha}^\top, \boldsymbol{\beta}_1^\top, \boldsymbol{\beta}_2^\top, \ldots, \boldsymbol{\beta}_b^\top)^\top$$

- Basis functions are parameterized by $\boldsymbol{\beta}_i$.

Three-Layer Networks

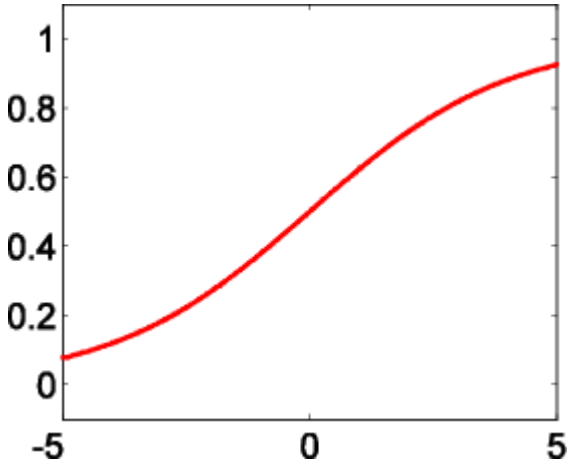
- Such a hierarchical model can be represented as a 3-layer network.



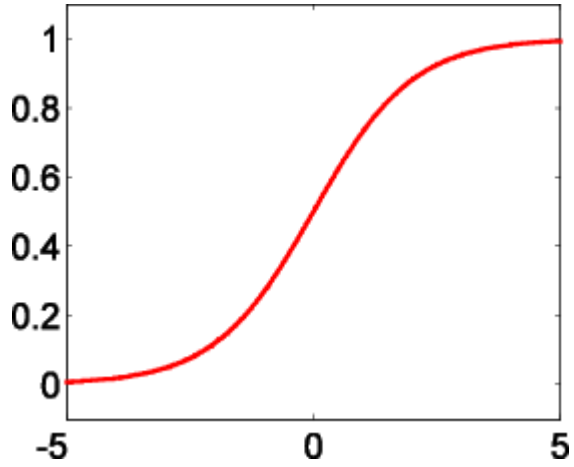
Sigmoid Function

- A typical basis function: Sigmoid function

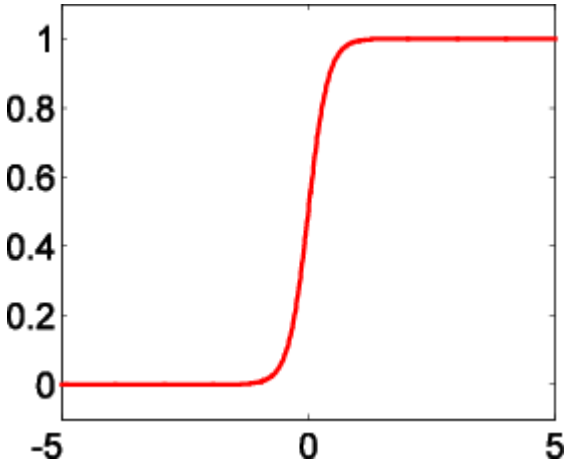
$$\varphi(\boldsymbol{x};\boldsymbol{\beta},h)=\frac{1}{1+\exp\left(-\langle\boldsymbol{x},\boldsymbol{\beta}\rangle-h\right)}$$



β :Small



β :Medium



β :Large

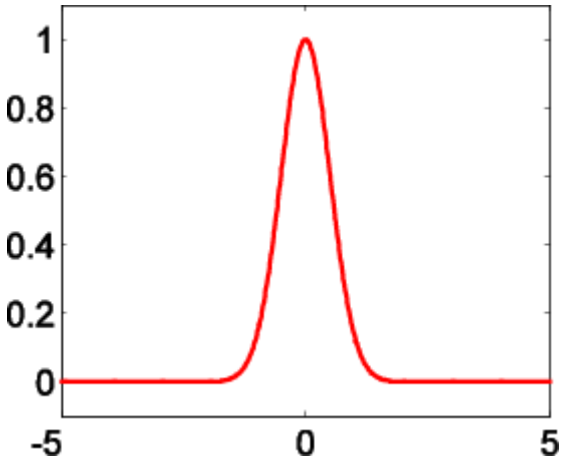
Perceptrons

- The behavior of the sigmoid functions is similar to neurons in our brain.
- For this reason, hierarchical models with sigmoid functions are called artificial neural networks or perceptrons.
- Mathematically, a 3-layer neural network can approximate any continuous functions with arbitrary small error ("a universal approximator").

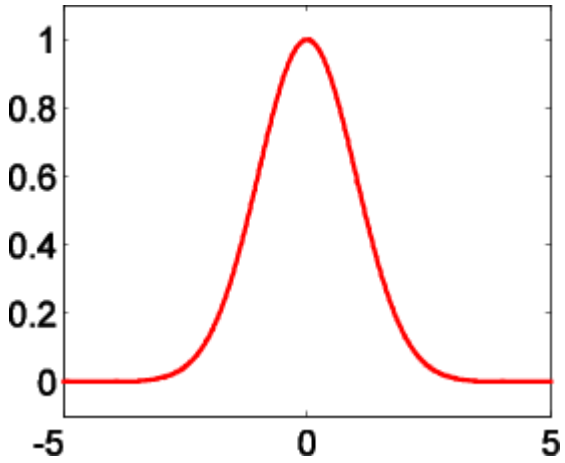
Gaussian Radial Basis Function

- Another popular basis function:
  Gaussian radial basis function

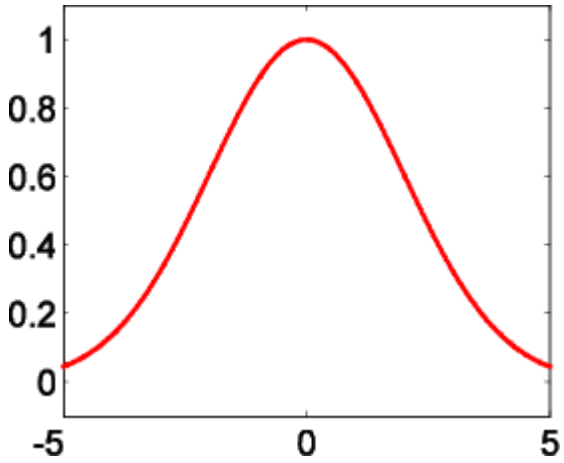
$$\varphi(\boldsymbol{x};\boldsymbol{\beta},c)=\exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{\beta}\|^2}{2c^2}\right)$$



c :Small



c :Medium



c :Large

Least-Squares Learning

$$f_{\boldsymbol{w}}(\boldsymbol{x}) = \sum_{i=1}^{b} \alpha_i \varphi(\boldsymbol{x}; \boldsymbol{\beta}_i)$$

$$\boldsymbol{w} = (\boldsymbol{\alpha}^\top, \boldsymbol{\beta}_1^\top, \boldsymbol{\beta}_2^\top, \ldots, \boldsymbol{\beta}_b^\top)^\top$$

- Least-squares learning is often used for training hierarchical models.

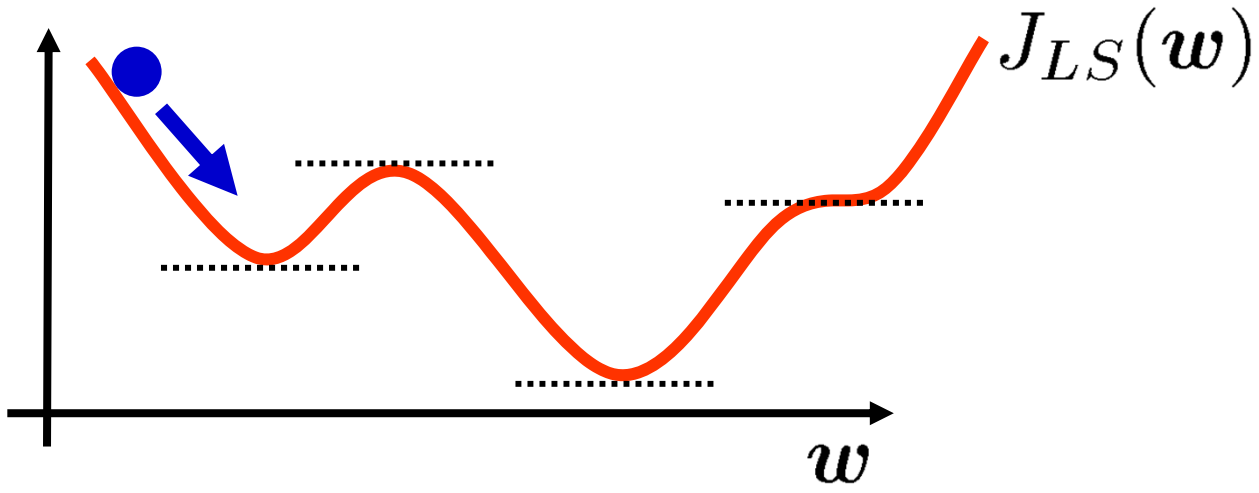
$$\min_{\boldsymbol{w}} J_{LS}(\boldsymbol{w})$$

$$J_{LS}(\boldsymbol{w}) = \sum_{i=1}^{n} \left(f_{\boldsymbol{w}}(\boldsymbol{x}_i) - y_i\right)^2$$

How to Obtain A Solution

- No analytic solution is known.
- Simple gradient search is usually used.

$$\boldsymbol{w}^{new} \longleftarrow \boldsymbol{w}^{old} - \varepsilon \nabla J_{LS}(\boldsymbol{w}^{old})$$



- One of the local minima can be found.

Error Back-Propagation

- Efficient calculation of gradient for sigmoid basis functions:

$$o_{i,j} = \varphi(\boldsymbol{x}_i; \boldsymbol{\beta}_j, h_j)$$

$$\delta_i = (f_{\boldsymbol{w}}(\boldsymbol{x}_i) - y_i)$$

$$\boldsymbol{x} = (x^{(1)}, \ldots, x^{(d)})^\top$$

$$\frac{\partial J_{LS}}{\partial \alpha_j} = 2 \sum_{i=1}^{n} o_{i,j} \delta_i$$

$$\frac{\partial J_{LS}}{\partial \beta_j^{(k)}} = 2\alpha_j \sum_{i=1}^{n} o_{i,j} \left(1 - o_{i,j}\right) x_i^{(k)} \delta_i$$

$$\frac{\partial J_{LS}}{\partial h_j} = 2\alpha_j \sum_{i=1}^{n} o_{i,j} \left(1 - o_{i,j}\right) \delta_i$$

Homework!

Error Back-Propagation (cont.)

- When the output values of the network are calculated, the input points are propagated following the forward path.
- On the other hand, when the gradients are calculated, the output error $(f_{\boldsymbol{w}}(\boldsymbol{x}_i) - y_i)$ is propagated backward.
- For this reason, this algorithm is called the error back-propagation.
- However, it is gradient descent so global convergence is not guaranteed.

Stochastic Gradient Descent

- In the usual gradient method, all training examples are used at the same time.
- In practice, the following stochastic method would be computationally advantageous.
  - Randomly choose one of the training examples (say, $(\boldsymbol{x}_i, y_i)$)
  - Update the parameter vector by

$$\boldsymbol{w}^{new} \longleftarrow \boldsymbol{w}^{old} - \varepsilon \nabla J_i(\boldsymbol{w}^{old})$$

$$J_i(\boldsymbol{w}) = (f_{\boldsymbol{w}}(\boldsymbol{x}_i) - y_i)^2$$

  - Repeat this procedure until convergence.

Avoiding Overfitting

$$f_{\boldsymbol{w}}(\boldsymbol{x}) = \sum_{i=1}^{b} \alpha_i \varphi(\boldsymbol{x}; \boldsymbol{\beta}_i)$$

$$\boldsymbol{w} = (\boldsymbol{\alpha}^\top, \boldsymbol{\beta}_1^\top, \boldsymbol{\beta}_2^\top, \ldots, \boldsymbol{\beta}_b^\top)^\top.$$

- LS overfits to noisy samples.
  - Regularization:

$$\min_{\boldsymbol{w}} [J_{LS}(\boldsymbol{w}) + \lambda \|\boldsymbol{w}\|^2]$$

$$J_{LS}(\boldsymbol{w}) = \sum_{i=1}^{n} (f_{\boldsymbol{w}}(\boldsymbol{x}_i) - y_i)^2 \qquad \lambda > 0$$

  - Early stopping: Stop gradient descent before it converges

Special Structure of Neural Networks

- Parameters and functions are not one-to-one:

$$f_{\boldsymbol{w}}(\boldsymbol{x}) = \sum_{i=1}^{b} \alpha_i \varphi(\boldsymbol{x}; \boldsymbol{\beta}_i)$$

  - If $\alpha_i = 0$, any $\boldsymbol{\beta}_i$ gives the same function.
  - Any permutation of units in the hidden layer gives the same function.

- This non-identifiability causes great difficulty in the mathematical analysis of neural networks.

S. Watanabe, Algebraic Geometry and Statistical Learning Theory, Cambridge University Press, 2009

Special Structure of Neural Networks (cont.)

- Theory: Neural networks were shown to go together well with Bayesian learning:
  - Given a prior distribution of parameters, compute the posterior distribution of parameters.

$$p(\boldsymbol{w}|\text{Data}) = p(\text{Data}|\boldsymbol{w})p(\boldsymbol{w})$$

- Practice: Training neural networks from many different initial values and taking their ensemble sometimes work well.

Pre-Training of Hidden Layers

- Neural networks are prone to a poor local optimum solution.
- Recent idea ("a deep belief network"):
  - First, pre-training hidden layers one by one in an unsupervised manner.
  - Then all layers are fine-tuned by supervised training.
- The pre-training idea was shown to work well in experiments.

Hinton and Salakhutdinov. Reducing the dimensionality of data with neural networks. Science, 2006.

Homework

1. Prove that the gradients for sigmoid basis functions are given as

$$o_{i,j} = \varphi(\boldsymbol{x}_i; \boldsymbol{\beta}_j, h_j)$$

$$\delta_i = (f_{\boldsymbol{w}}(\boldsymbol{x}_i) - y_i)$$

$$\boldsymbol{x} = (x^{(1)}, \ldots, x^{(d)})^\top$$

$$\frac{\partial J_{LS}}{\partial \alpha_j} = 2 \sum_{i=1}^{n} o_{i,j} \delta_i$$

$$\frac{\partial J_{LS}}{\partial \beta_j^{(k)}} = 2\alpha_j \sum_{i=1}^{n} o_{i,j} \left(1 - o_{i,j}\right) x_i^{(k)} \delta_i$$

$$\frac{\partial J_{LS}}{\partial h_j} = 2\alpha_j \sum_{i=1}^{n} o_{i,j} \left(1 - o_{i,j}\right) \delta_i$$

Homework (cont.)

2. For your own toy 1-dimensional data, perform simulations using
   - 3-layer neural networks with sigmoids/Gaussians
   - Error back-propagation

   and analyze the results, e.g., by changing
   - Target functions
   - Number of training samples
   - Noise level
   - Number of neurons in the second layer (b)
   - Initialization heuristics
   - Number of ensembles

Notification of Final Assignment

1. Apply supervised learning techniques to your data set and analyze it.
2. Write your opinion about this course.

- Final report deadline: Aug 3rd (Fri.)
- E-mail submission is also accepted!

sugi@cs.titech.ac.jp

Mini-Workshop on Data Mining

- On July 10th and 24th, we will have a mini-workshop on data mining.
- Several students present their own data mining results.
- Those who give a talk at the workshop will have very good grades!

Mini-Workshop on Data Mining

- Application (just to declare that you want to give a presentation) deadline: June 19th.
- Presentation: 10-15 minutes (?).
  - Specification of your dataset
  - Methods used
  - Outcome
- Slides should be in English.
- Better to speak in English, but Japanese is also allowed.