DOE and nonlinear regression

MAE301 Applied Experimental Statistics

Max Yi Ren

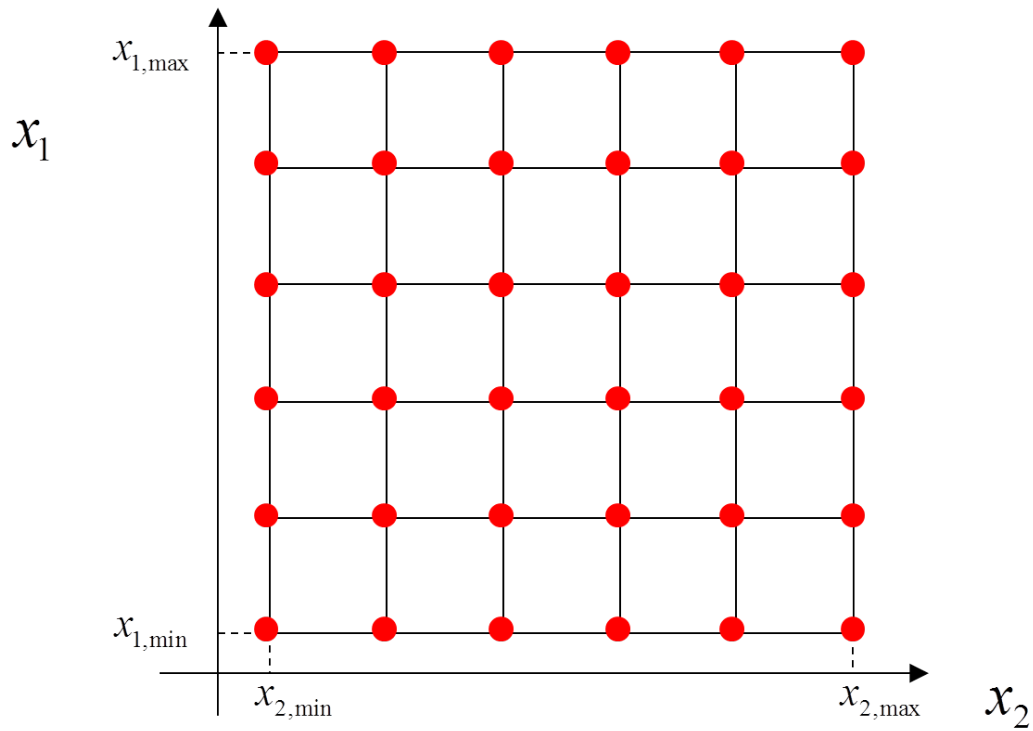
Department of Mechanical Engineering
Arizona State University

November 10, 2015

1. design of experiments
2. nonlinear models
3. summary

Sampling

Design of experiments (optimal experiment design):
Effectively sample the design space to create a statistical model with high prediction performance.
A naive way is to use a *full factorial experiment*:



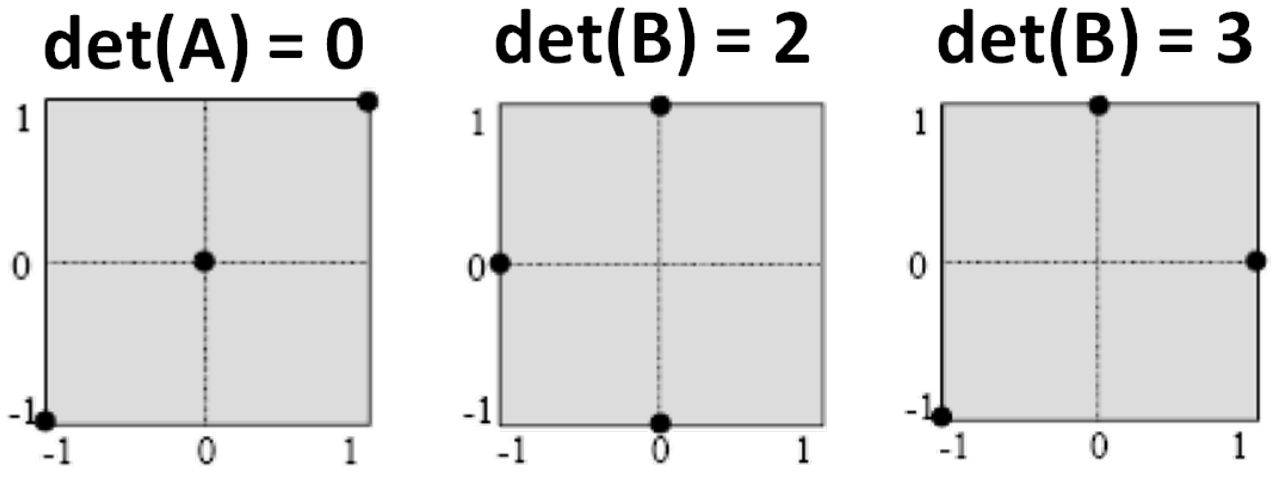
Full factorial sampling costs l^p samples.

D-optimal design

Consider the OLS solution:

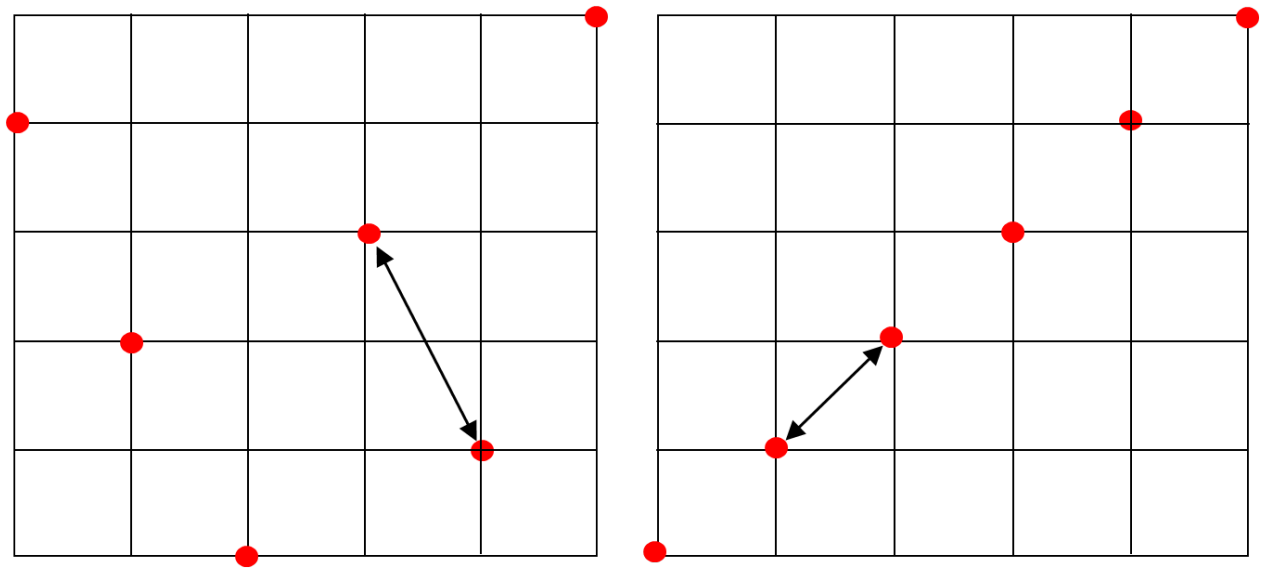
$$\beta^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}. \tag{1}$$

The variance between β^* and the unknown true model can be reduced with larger $|\mathbf{X}^T\mathbf{X}|$. Therefore, it is preferred to maximize $|\mathbf{X}^T\mathbf{X}|$ for a fixed amount of samples.



Latin hypercube

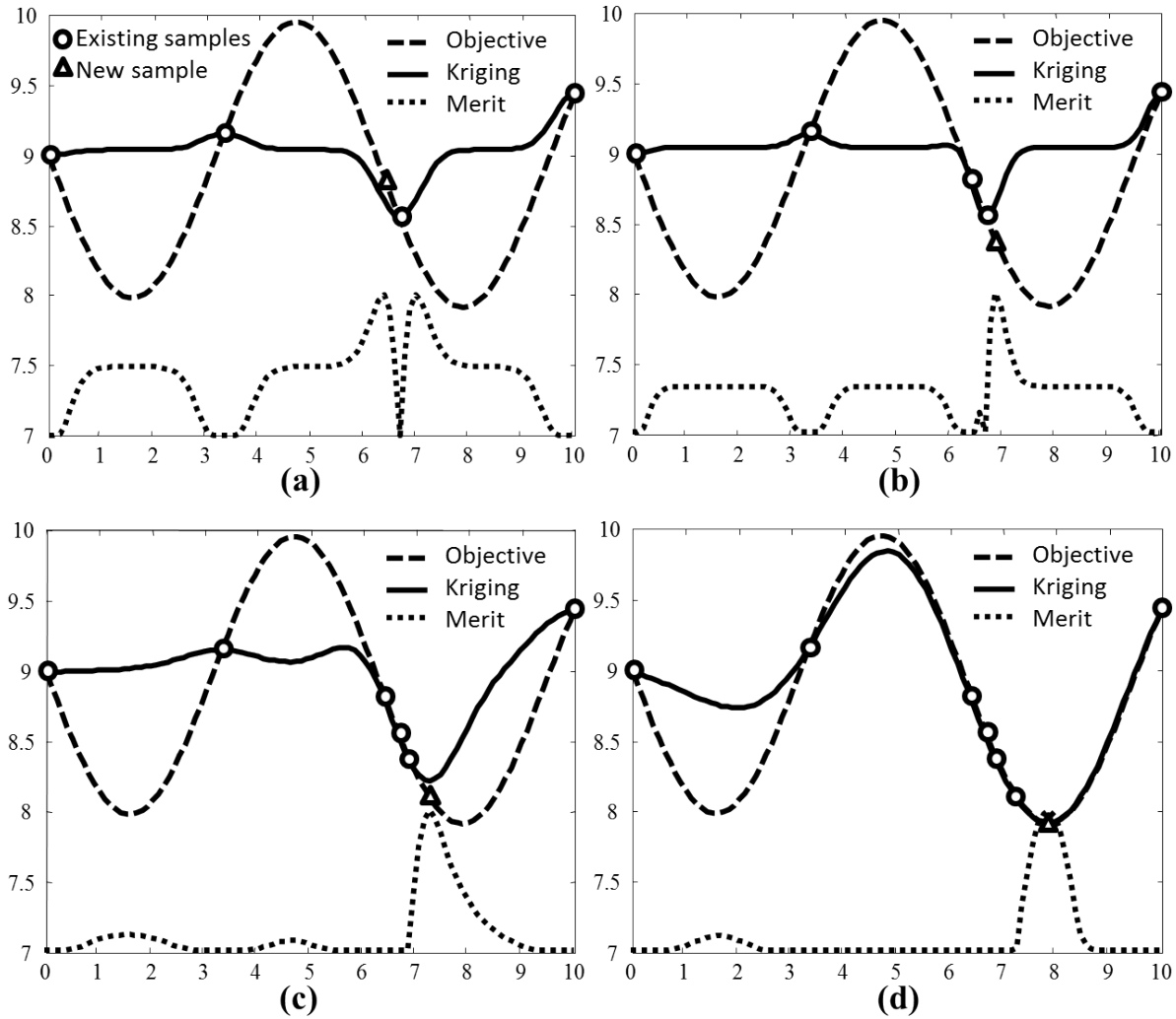
Latin hypercube sampling (LHS) uses l samples regardless of the number of variables (p), and is therefore widely adopted (for linear and nonlinear models).



In LHS, there does not exist samples that share the same value on any variable. To implement LHS, one should also include dispersion criteria, e.g., maximizing the minimum distance between sample points, or minimizing the correlation.

Active learning (Adaptive sampling)

One example in optimization:



The Efficient Global Optimization (EGO) algorithm finds new samples based on the current model and model uncertainty.

Active learning (Adaptive sampling)

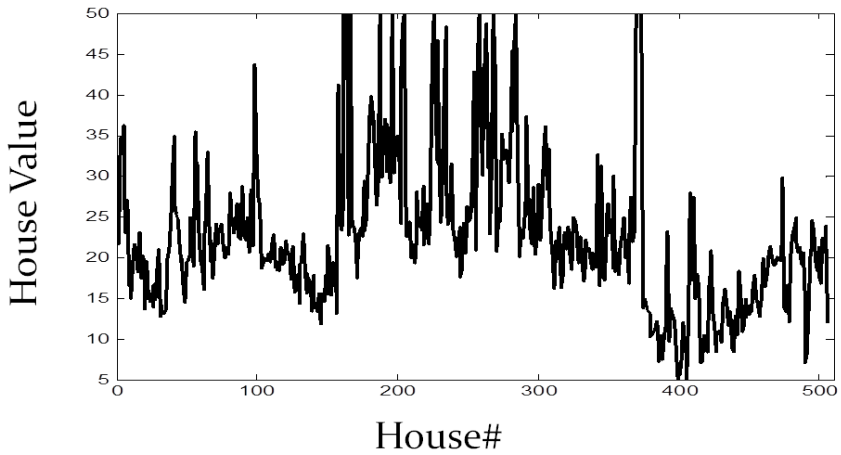
Another example is the 20 question game: The most efficient way is to find a most uncertain answer, i.e., an answer with 50% chances to be answered "yes" and "no".

Table: An example of the group identification problem.

Group	Heroes		Villains	
Object	Superman	Batman	Catwoman	Joker
Prior Probability	0.25	0.25	0.25	0.25
Q1:Mask?	Yes	Yes	No	No
Q2:Can fly?	Yes	No	No	No
Q3:Female?	No	No	Yes	No

Regression methods

When the response cannot be linearly approximated by input variables, or we don't know what features (e.g., polynomial terms) to use for modeling the observations, OLS may not work well.
An example from Matlab House.data: $\mathbf{X}$ (506×13): 506 houses with 13 parameters, $\mathbf{y}$ (506×1): house values.



	OLS	feed-forward NN	SVR RBF
CV R^2	0.66	0.77	0.83

Table: Crossvalidation on House.data

Ridge regression (RR)

Recall the solution of OLS

$$\boldsymbol{\beta}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}.$$

When $\mathbf{X}^T\mathbf{X}$ is ill-conditioned, we can try

$$\boldsymbol{\beta}^* = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y},$$

where λ is an unknown parameter.
This solution corresponds to minimizing

$$||\mathbf{X}\boldsymbol{\beta} - \mathbf{y}||^2 + \lambda||\boldsymbol{\beta}||^2.$$

This objective tries to minimize MSE within a sphere of possible $\boldsymbol{\beta}$. λ represents your believe of the observations, i.e., the larger λ is, the less believe you have. One can use cross-validation on the training data to find the optimal value of λ.

Feed-forward neural networks (NNFF)

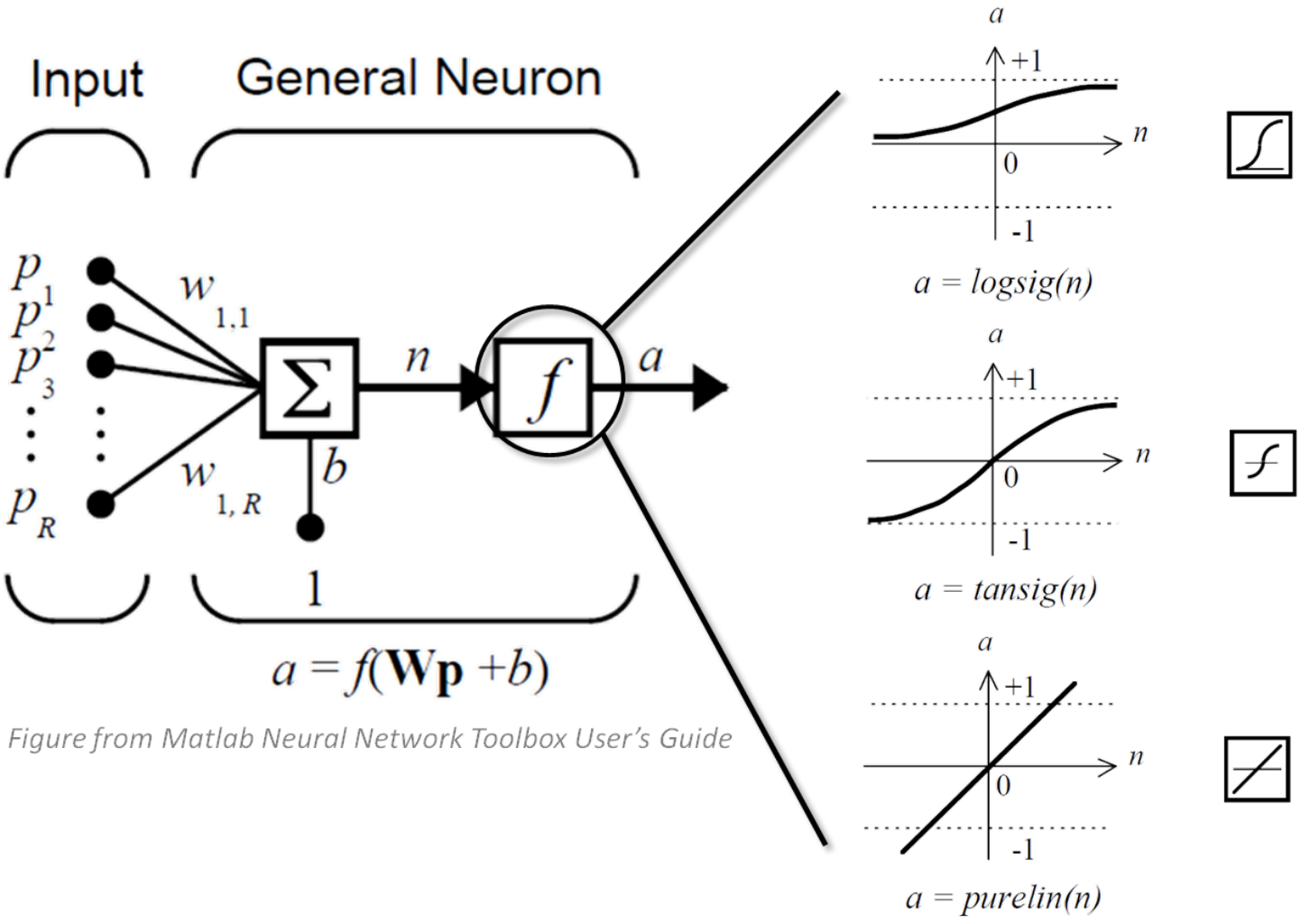
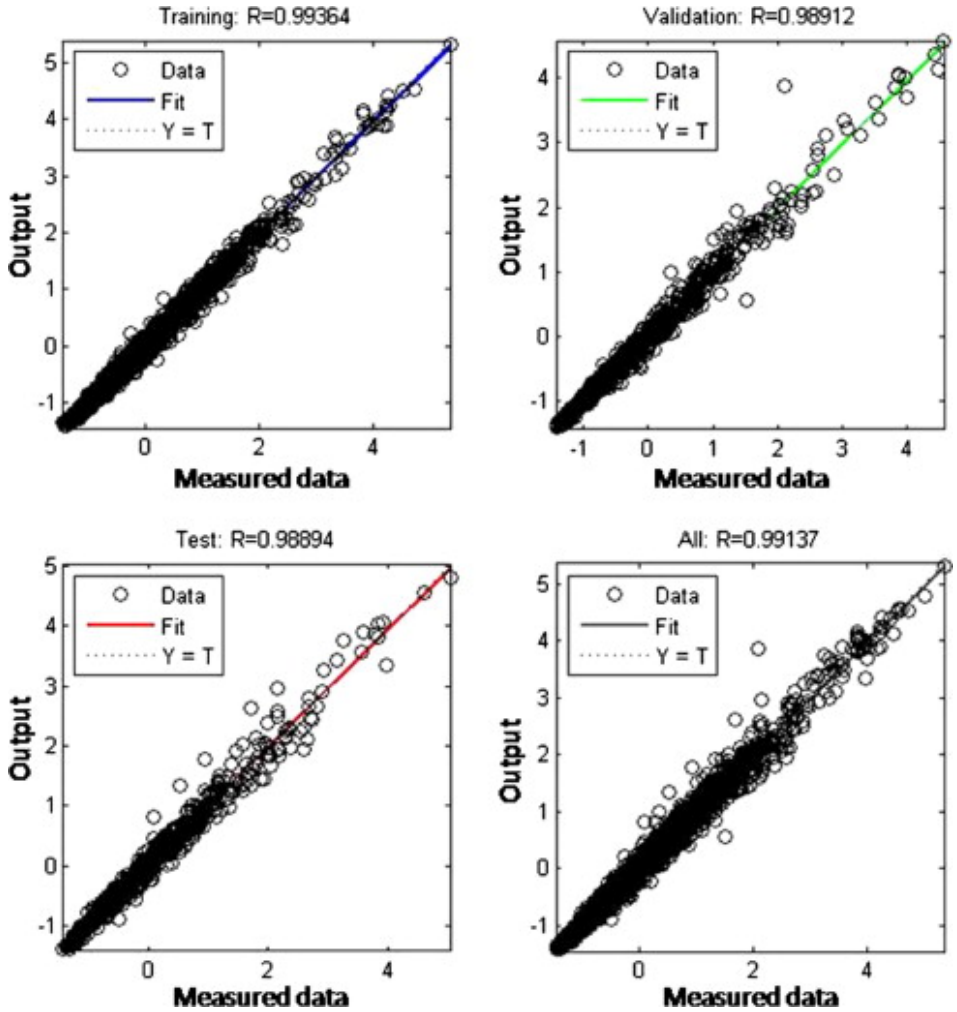


Figure from Matlab Neural Network Toolbox User's Guide

A simplest feed-forward neural net. One may add arbitrary number of layers and neurons to the model.

Feed-forward neural networks (NNFF)

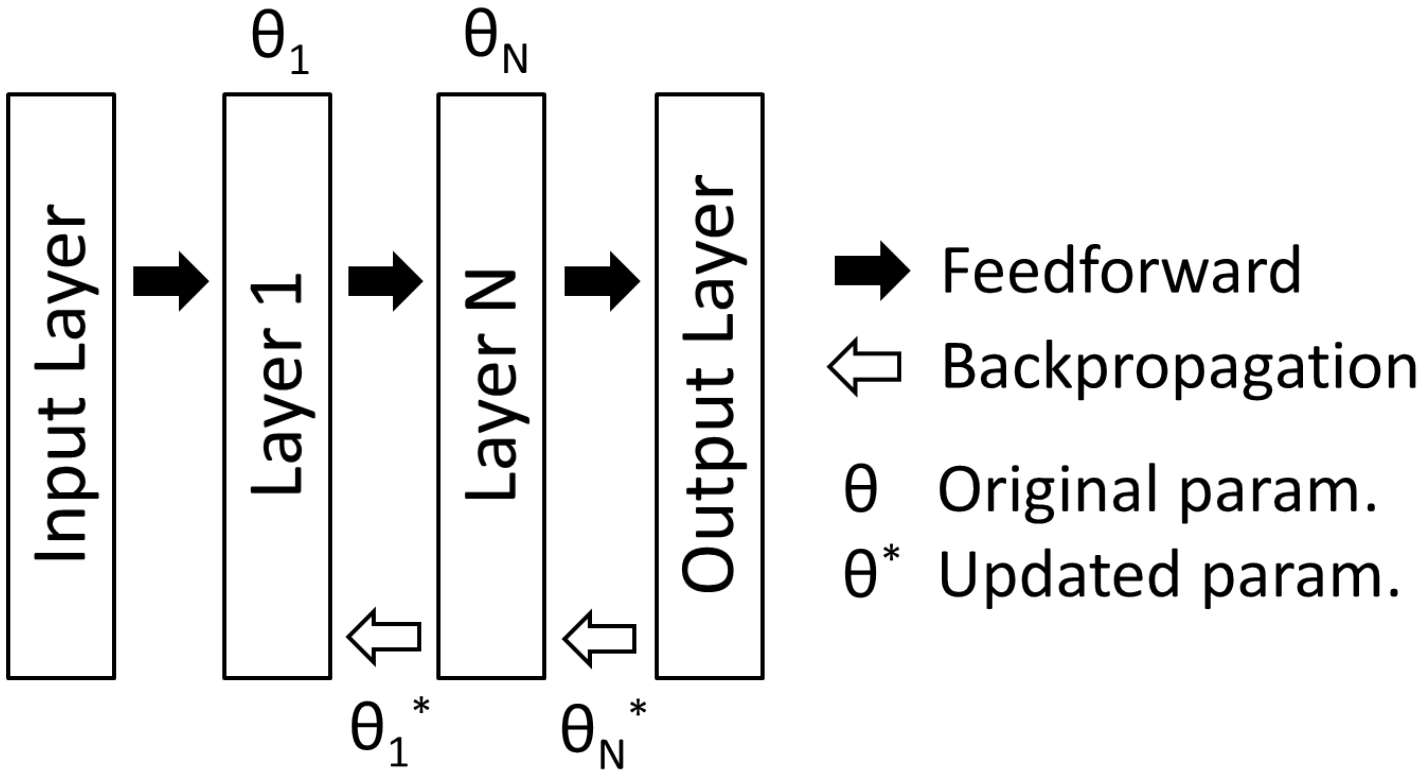
Matlab uses a portion of the training data for validation. The training (optimization) will terminate when gradient is close to zero or MSE of the validation set does not decrease for a few iterations.



Feed-forward neural networks (NNFF)

We find optimal network parameters to minimize the mean-squared error (MSE) of the testing data.

Training of an NNFF requires back-propagation: We start by minimizing MSE by w.r.t. parameters of the last hidden layer and fixing all other parameters; then move on to optimize the parameters at previous layers.



Feed-forward neural networks (NNFF)

Deep networks have shown to be superior in object/audio recognition. One good example is the AlexNet with seven hidden layers.

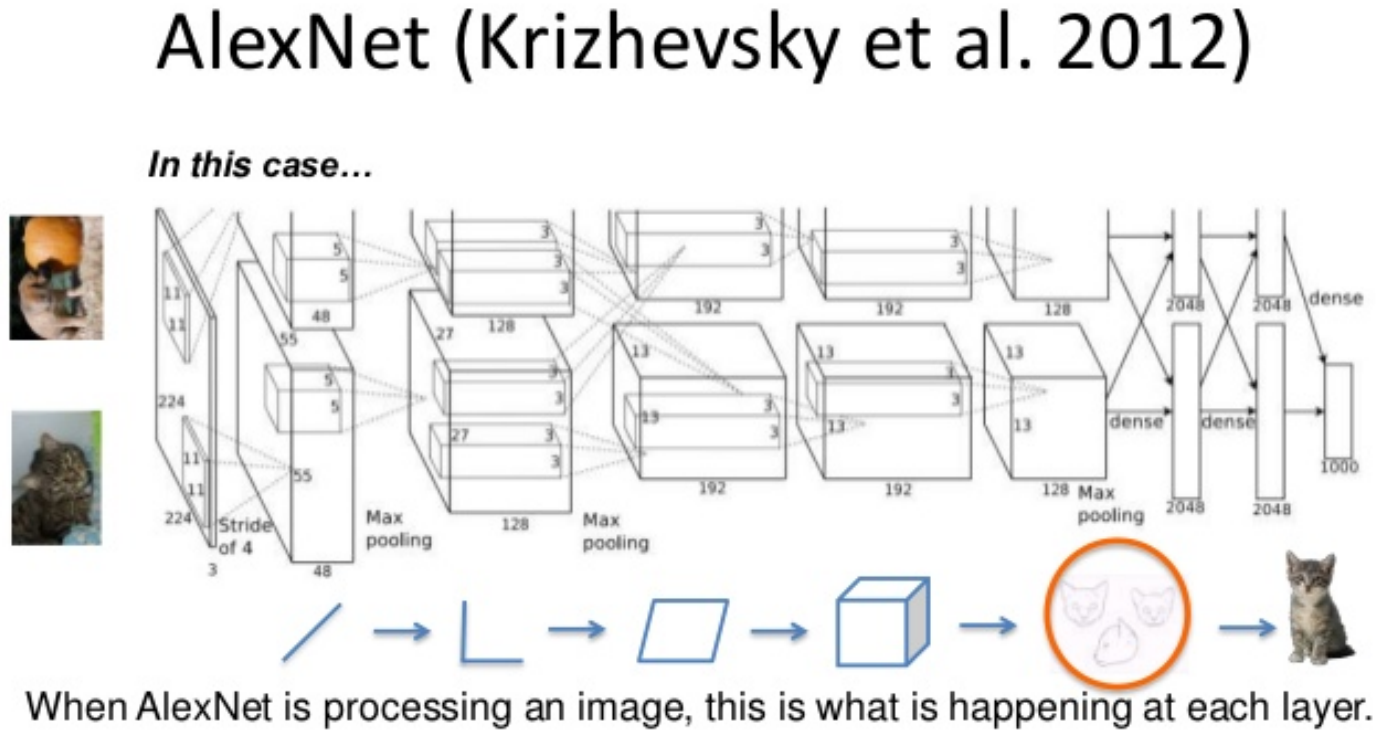


Figure: AlexNet overview. Image from http://image.slidesharecdn.com/

Radial-basis neural networks (NNRB)

When sample $\mathbf{y}$ are deterministic, the following NNRB model can be used for interpolation purpose:

$$y(\mathbf{x}) = \sum w_i \exp(-\gamma ||\mathbf{x} - \mathbf{x}_i||^2)$$

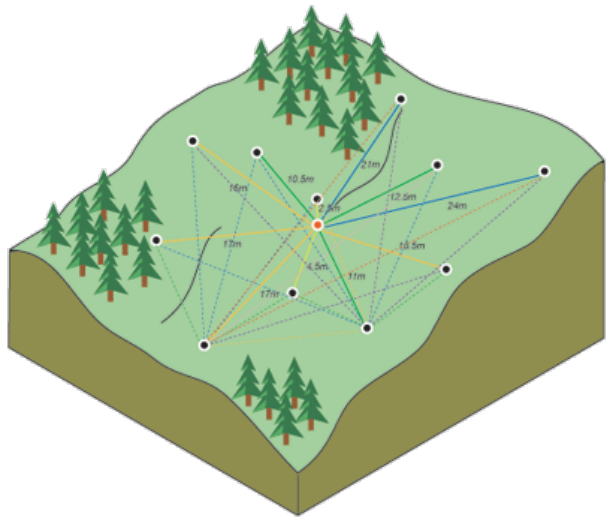
, where $\mathbf{w}$ are network weights.
Let $r_j(\mathbf{x}) = \exp(-\gamma ||\mathbf{x} - \mathbf{x}_j||^2)$, with n samples we have $\mathbf{Rw} = \mathbf{y}$, where the matrix $\mathbf{R}$ is

$$\begin{bmatrix} r_1(\mathbf{x}_1) & r_2(\mathbf{x}_1) & \cdots & r_n(\mathbf{x}_1) \\ r_1(\mathbf{x}_2) & r_2(\mathbf{x}_2) & \cdots & r_n(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ r_1(\mathbf{x}_n) & r_2(\mathbf{x}_n) & \cdots & r_n(\mathbf{x}_n) \end{bmatrix}$$

It can be proved that $\mathbf{R}$ is non-singular if samples $\mathbf{X}$ are distinct, and thus the weights can be solved as $\mathbf{w} = \mathbf{R}^{-1}\mathbf{y}$.

Kriging

Kriging: A geostatistical techniques to interpolate the elevation of the landscape as a function of the geographic location at an unobserved location from observations of its value at nearby locations.



The Kriging model has

$$\hat{Y}(\mathbf{x}) = \sum_{i=1}^{n} w_i(\mathbf{x}) Y(\mathbf{x}_i),$$

where $Y(\mathbf{x})$ is a random field on $\mathbf{x}$, $w_i(\mathbf{x})$ is the weight measuring the similarity between $\mathbf{x}$ and $\mathbf{x}_i$.

Kriging

The simple Kriging model assumes $E[Y(\mathbf{x})] = 0$, which results in the model

$$\hat{y}(\mathbf{x}) = \mathbf{r}(\mathbf{x})^T \mathbf{R}^{-1} \mathbf{y},$$

where the vector $\mathbf{r}(\mathbf{x})$ measures the similarities between $\mathbf{x}$ and all samples $\mathbf{x}_i$, and the matrix $\mathbf{R}$ measures the similarities among all samples. When we use the radial-basis (Gaussian) function for measuring similarity, simple Kriging results in the same model as from NNRB.

When assuming $E[Y(\mathbf{x})] = \text{const}$, we will have the Kriging model:

$$\hat{y}(\mathbf{x}) = \hat{b} + \mathbf{r}(\mathbf{x})^T \mathbf{R}^{-1} (\mathbf{y} - \hat{b}\mathbf{1}),$$

where

$$\hat{b} = \frac{\mathbf{1}^T \mathbf{R}^{-1} \mathbf{y}}{\mathbf{1}^T \mathbf{R}^{-1} \mathbf{1}}.$$

The prediction $\hat{y}$ at any sampled $\mathbf{x}$ matches the sampled value y. Therefore Kriging is widely used for metamodeling from computer simulations (with deterministic outputs).

Support vector regression (SVR)

SVR is the regression version of the original support vector machine (SVM) for classification. The idea is to balance the training error (MSE) and model complexity to prevent over-fitting:

$$
\begin{aligned}
\min_{\beta, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*} \quad & ||\beta||^2 + C_1 \sum \xi_i + C_2 \sum \xi_i^* \\
\text{subject to} \quad & \boldsymbol{\beta}^T \mathbf{x}_i + b - y_i \leq \varepsilon + \xi_i, \\
& y_i - \boldsymbol{\beta}^T \mathbf{x}_i - b \leq \varepsilon + \xi_i^*, \\
& \xi_i,\ \xi_i^* \geq 0, \forall i.
\end{aligned}
$$

Similar to Kriging, with a definition of similarity, SVR can train nonlinear models. It will have the same analytical solution to Kriging when training error is forced to zero.

Summary

- Sample using Latin hypercube, $\mathbf{X}$ needs to have similar scale in each dimension;
- Always try OLS first;
- Use AIC (or BIC) for (linear) model selection;
- Use crossvalidation (within the training data) for model parameter tuning;
- OLS, Kriging, RR, SVR are easier to tune than NNFF but NNFF can be more powerful if well-tuned;
- OLS, RR, SVR can be used when random noise exists in data;
- Kriging can be used when data is deterministically generated (through computer simulations);
- Choose a model (OLS, Kriging, RR, SVR, etc.) with the best test (or crossvalidation) performance. Retrain the model with all data.