Function Approximation and RBF Neural Networks

What is function approximation ?

Consider two sets $F \subseteq R^m$ and $L \subseteq R^l$, and a mapping f from F to L. Suppose that for a set $\Omega \subset F$, a mapping g is given such that

$$g(x) \approx f(x), \quad \text{for } x \in \Omega$$

The so called function approximation problem is to find a mapping $\hat{f}$ satifying

$$\left\| \hat{f}(x) - f(x) \right\| \leq \varepsilon, \text{for } x \in F$$

where $\varepsilon > 0$ is the tolerance, and $\|\bullet\|$ can be any norm.

Some practical considerations

Since the original mapping is unknown, it cannot be used to evaluate the quality of a solution. In practice, we should find $\hat{\boldsymbol{f}}$ such that

$$\left\| \hat{\boldsymbol{f}}(\boldsymbol{x}) - \boldsymbol{g}(\boldsymbol{x}) \right\| < \varepsilon, \text{for } \boldsymbol{x} \in \Omega$$

In general, function approximation is an ill-posed problem. A solution can be good for the training set Ω, but bad for $\boldsymbol{x} \in \boldsymbol{F}\text{-}\boldsymbol{\Omega}$. To get solutions that generalize well, some regularization constraints are usually used.

Polynomial based approximation

One model for function approximation is polynomial. Talor expansion is one example. If the function is smooth enough, the Talor expansion is given as follows:

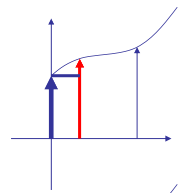
$$\boldsymbol{f(x)} = \boldsymbol{f(a)} + \frac{\boldsymbol{f'(a)}}{1!}(\boldsymbol{x}-\boldsymbol{a}) + \cdots + \frac{\boldsymbol{f}^{(n-1)}\boldsymbol{(a)}}{(\boldsymbol{n}-1)!}(\boldsymbol{x}-\boldsymbol{a})^{n-1} + \boldsymbol{R}_n$$

where $\boldsymbol{R}_n$ is the reminder or the approximation error. That is, we can get an approximated value of ***f(x)*** using information at only one point.

0-th order and 1-st order approximation

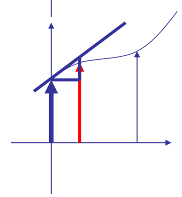
$\Rightarrow$ Nearest neighbor approximation

Replace x with the nearest example a, and approximate $f(x)$ with $f(a)$.



$\Rightarrow$ Linear approximation

If the first order derivative exists, we can approximate $f(x)$ with $f(a)+f'(a)(x-a)$.



General polynomial approximation

In general, a function $f(x)$ can be approximated by

$$f(x)=\sum_{i=0}^{n-1} a_i x^i + \varepsilon$$

The coefficients can be found using the training examples in Ω. Methods include

1) Solving a simultaneous linear equation.
2) Solving a quadratic optimization problem in case there are noises.

Approximation based on Fourier transformation

Analysis :

$$\boldsymbol{F}(\omega)=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} \boldsymbol{f}(\boldsymbol{x})\boldsymbol{e}^{-j\omega x}\boldsymbol{dx} =< \boldsymbol{f}, \exp(-\boldsymbol{j}\omega\boldsymbol{x}) >$$

Synthesis (Approximation)

$$\boldsymbol{f}(\boldsymbol{x})=\frac{1}{\sqrt{2\pi}}\int_{a}^{b} \boldsymbol{F}(\omega)\boldsymbol{e}^{j\omega x}\boldsymbol{dx} =< \boldsymbol{F}(\omega), \exp(\boldsymbol{j}\omega\boldsymbol{x}) >$$

where $-\infty \le \boldsymbol{a} < \boldsymbol{b} \le \infty$

Basis function approximation

In fact, both polynomial and transformation based approximations can be formulated in a more general form as follows :

$$\boldsymbol{f(x)}=\sum_{k=0}^{N} \boldsymbol{a_k \varphi_k(x)} + \varepsilon$$

where $\boldsymbol{\varphi_k(x)}$ is called a basis function.

Quizzes : What are the basis functions in the approximation methods discussed so far ?

Function approximation with local basis functions

One example of local basis function is the Gaussian function given as follows :

$$\varphi_k(\boldsymbol{x}) = \exp\left(-\frac{1}{2\boldsymbol{\sigma}_k^2}\|\boldsymbol{x}-\boldsymbol{x}_k\|^2\right)$$

where $\boldsymbol{x}_i$ and $\boldsymbol{\sigma}_i$ are, respectively, the center and the variance of the $\boldsymbol{k}$-th basis function. This kind of basis function is called radial basis function (RBF).

RBF neural networks

- A RBF neural network is a three layer neural network
  - Input neurons: same as the MLP
  - Output neurons: linear combinations
  - Hidden neurons: basis functions
- For a one-output network, the output is given by

$$\boxed{\boldsymbol{o} = \sum_{\boldsymbol{k}=1}^{\boldsymbol{N}} \boldsymbol{w}_k \varphi(\|\boldsymbol{x}-\boldsymbol{x}_i\|)}$$

Training of RBF neural networks

- Provide the training patterns with known function values.
- Use the training patterns as the centers of the radial basis functions.
- Find the weights of the output neuron by solving simultaneous linear equation.

Problems to be solved

- Too many hidden neurons if the number of training patterns is large
  - Selection of important training data.
  - Re-location of the centers to improve the performance of the network.
  - Optimizing the variances of each RBF.
- For noisy data, the approximation is ill-posed
  - Regularization is needed.

How to regularize ?

- The basic idea of regularization is to add a penalty in the error function.
- Minimizing the error along with the penalty can find a solution as smooth as possible.
- The cost function is given by

$$\boldsymbol{E}(\hat{\boldsymbol{f}})=\sum_{\boldsymbol{i}=1}^{\boldsymbol{P}}(\boldsymbol{d}_i-\hat{\boldsymbol{f}}(\boldsymbol{x}_i))^2+\lambda\|\boldsymbol{P}\hat{\boldsymbol{f}}\|^2$$

where $\lambda>0$ is the regularization parameter, and P is a linear differential operator.

A generalized RBF neural network

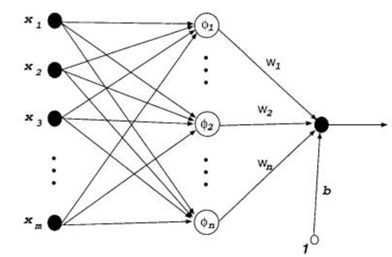
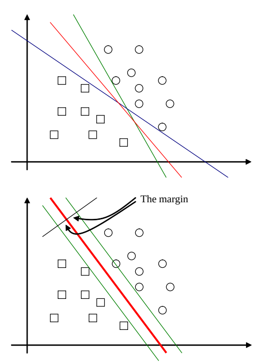


Figure 1: Radial basis function neural network

Support vector machine

- Support vector machine (SVM) is an algorithm for pattern classification.
- Proposed by V. Vapnik at AT&A in 1995.
- The first important feature of SVM is to make decisions with the maximum margin.



Problem formulation (1)

Consider a two-class problem. Suppose that we have l linearly separable examples $[(x_1,y_1),(x_2,y_2),\cdots,(x_l,y_l)]$, where $y_i\in\{-1,1\}$ is the teacher signal of $x_i\in D\subseteq R^n$.
We want to find a hyperplane to divide all examples into two classes, so that

1) The class labels are the same as the teacher signals.
2) The nearest examples of both sides are equal distance.

Problem formulation (2)

Suppose that the hyperplane to find is

$$\boldsymbol{H}: \boldsymbol{w}^t \boldsymbol{x} + \boldsymbol{b} = 0$$

The distance between an example $\boldsymbol{x}_i$ and $\boldsymbol{H}$ is given by

$$\text{distance}(\boldsymbol{H}, \boldsymbol{x}_i) = \frac{|\boldsymbol{w}^t \boldsymbol{x}_i + \boldsymbol{b}|}{\|\boldsymbol{w}\|}$$

$\boldsymbol{H}$ can be scaled so that for the nearest example we have

$$\boldsymbol{w}^t \boldsymbol{x} + \boldsymbol{b} = \pm 1 \quad \text{or} \quad \boldsymbol{y}(\boldsymbol{w}^t \boldsymbol{x} + \boldsymbol{b}) = 1$$

Thus the distance between $\boldsymbol{H}$ and the nearest examples is $1/\|\boldsymbol{w}\|$, and the margin is $2/\|\boldsymbol{w}\|$.

Problem formulation (3)

For an example outside the margin, we have

$$w^t x + b > 1 \quad \text{or} \quad w^t x + b < -1$$

or $\quad y(w^t x + b) > 1$

Now the problem can be formulated as follows :

$$\min \|w\|^2 / 2$$

$$\text{s.t.} \quad y_i(w^t x_i + b) \geq 1, \quad i = 1, 2, \cdots, l$$

Problem formulation (4)

The problem can be solved by optimizing the following Lagrangian function

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_i [y_i(w^t x + b) - 1]$$

where $\alpha_i \geq 0$ are the Lagrange multiplier s. This is called the primal Lagrangian. The dual form can be found by differentiating L with respect to w and b.

Problem formulation (5)

The maximum margin classification problem can be formulated as a quadratic optimizati on problem

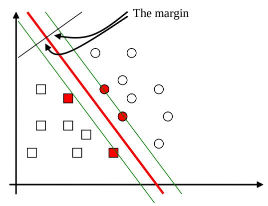
$$\max W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j x_i^t x_j$$

$$s.t. \quad \sum_{i=1}^{l} y_i \alpha_i = 0; \qquad \alpha_i \geq 0, \quad i = 1, 2, \cdots, l$$

If α^* is the optimal solution, the optimal w and b are given by

$$w = \sum_{i=1}^{l} y_i \alpha_i^* x_i, \quad b = -\frac{1}{2} w^t (x_{nearest}^{+1} + x_{nearest}^{-1})$$

Some remarks



- From the optimal solution we can see that the weight vector *w* is the linear combination of all training examples.
- Not all α_i take non-zero values.
- If α_i is non-zero, the corresponding training example x_i is called a **support vector.**
- Support vectors are the examples nearest to the hyperplane *H*.
- Only support vectors are useful for making decisions.

How to make the decision?

The linear discriminant function is defined as

$$f(x) = \mathrm{sgn}(w^t x + b)$$
$$= \mathrm{sgn}(\sum_{i=1}^{l} y_i \alpha_i (x_i^t x) + b)$$

For any un-known example x, it is classified to class +1 if $f(x) = 1$; or to class -1 if $f(x) = -1$.

Support vector machine with soft margin

- So far we have assumed that all data are linearly separable.
- In practice, most problems are not linearly separable.
- The original problem can be relaxed by allowing some classification errors.
- That is, some data points can be inside the margin, or equivalently, the constraints can take the following form:

$$\boxed{y_i(w^t x + b) \geq 1 - \xi_i, \quad i = 1,2,\ldots,l, \text{ with } \xi_i \geq 0}$$

- This problem can be solved in the same way as before.

Support vector machine with soft margin

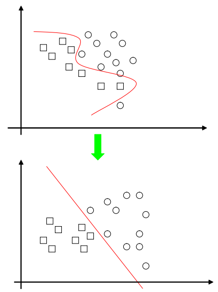
- In the dual form optimization problem, the constraints should be modified as

$$\boxed{0 \leq \alpha_i \leq C, \; i = 1,2,\cdots,l, \quad \text{and} \quad \sum_{i=1}^{l} \alpha_i y_i = 0}$$

where *C* is the upper bound (to be chosen by the user) on the Lagrange multipliers α_i.

Non-linear support vector machine

- For non-linear problems, soft margin alone is not enough.
- Another important feature of SVM is to use a non-linear mapping.
- All data are first mapped from a low dimensional space to a high dimensional space.
- All data will become linearly separable in the mapped space, if the dimensionality of that space is high enough.



Basic considerations

- However, if we try to find the optimal hyperplane in the high dimensional space, the computational cost will become very large.
- This problem can be avoided if we introduce the concept of kernel function.
- A function $\boldsymbol{k(x,y)}$ is a kernel function if it can be represented as

$$\boldsymbol{k(x,y)=\phi(x)^t\phi(y)=<\phi(x),\ \phi(y)>}$$

- Where x and y are n-dimensional vectors, and ϕ is the function for mapping x to the high dimensional space.

Examples of kernel functions

1) Polynomial kernel

$$k(x,y)=(<x,y>+c)^d$$

2) Gaussian kernel

$$k(x,y)=\frac{1}{c}e^{-\|x-y\|^2}$$

3) Sigmoid kernel

$$k(x,y)=\tanh[c<x,y>+\theta]$$

where c.d, and θ are parameters.

Optimal hyperplane in the mapped space

In the mapped space, the best hyperplane can be found by solving the following quadratic optimizati on problem

$$\max W(\alpha)=\sum_{i=1}^{l}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{l}y_iy_j\alpha_i\alpha_j\phi^t(x_i)\phi(x_j)$$

$$=\sum_{i=1}^{l}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{l}y_iy_j\alpha_i\alpha_jk(x_i,x_j)$$

$$s.t.\quad \sum_{i=1}^{l}y_i\alpha_i=0;\qquad \alpha_i\geq 0,\quad i=1,2,\cdots,l$$

How to make the decision?

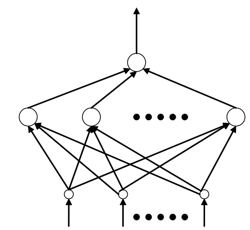
The linear discriminant function in the mapped space is defined as

$$f(x) = \text{sgn}(w^t \phi(x) + b)$$
$$= \text{sgn}(\sum_{i=1}^{l} y_i \alpha_i (\phi^t(x_i)\phi(x) + b)$$
$$= \text{sgn}(\sum_{i=1}^{l} y_i \alpha_i k(x_i, x) + b)$$

For any un-known example x, it is classified to class $+1$ if $f(x) = 1$; or to class -1 if $f(x) = -1$.

SVM is also a neural network!

- The original form of SVM is for 2-class problems, so there is only one output.
- For n-dimensional data, there are n inputs.
- The number of actually used hidden neurons is the number of support vectors.



Relation between SVM and RBF neural network

- SVM provides an efficient way for selecting the patterns to be used in an RBF-NN.
- However, SVM is not able to fine-tune the positions of the RBF centers and the widths of the basis functions.
- On the other hand, we can use different kernels in SVM for solving different problems.
- So far, SVM is known as one of the best method for pattern classification/recognition.

Team Project VI

- Try to find some free program on the internet for designing SVM.
- Down-load at least two databases from the UCI machine learning repository, and design the SVMs using the program.
- Compare SVMs and the multilayer feedforward neural networks (MLPs) trained using BP, and make some conclusions about the accuracy, computing cost, system size, etc.