We can revisit the multivariate interpolation problem in a higher-dimensional space.

PROBLEM: Given a set of N different points $\{\underline{x}_i \in \mathbb{R}^{m_0}\}_{i=1,2,\ldots,N}$ and a corresponding set of N real nos $\{d_i \in \mathbb{R}\}_{i=1,\ldots,N}$ find a function $F: \mathbb{R}^N \to \mathbb{R}^1$ /

$$F(\underline{x}_i) = d_i \; ; \; i = 1, \ldots, N$$

The idea of radial basis functions can help!

(stemming from the Gaussian hidden units we saw in the XOR problem)

Suppose $F(\underline{x}) = \sum_{i=1}^{N} w_i \, \varphi(\|\underline{x} - \underline{x}_i\|)$

$\|\cdot\|$ is the L_2-norm and $\varphi(\cdot)$ is a set of N arbitrary 'smooth' non-linear functions. (L_2-norm is a reason for the radial symmetry)

Let $\varphi_{ji} = \varphi(\|\underline{x}_j - \underline{x}_i\|)$, $j, i = 1, \ldots, N$

Let $\underline{d} = [d_1 \ \ldots \ d_N]^T$ (desired)

$\underline{w} = [w_1 \ \ldots \ w_N]^T$ (linear weight)

Let us form a matrix eqn under the interpolation constraints.

$$\begin{bmatrix} \varphi_{11} & \varphi_{12} & \cdots & \varphi_{1N} \\ \vdots & & & \\ & & & \\ \varphi_{N1} & \varphi_{N2} & & \varphi_{NN} \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} d_1 \\ \vdots \\ d_N \end{bmatrix}$$

φ_{N1}: data point; φ_{N2}: basis

$$\Phi := [\varphi_{ji}] \quad j, i = 1, \ldots, N$$

$$\Phi \underline{w} = \underline{d}; \quad \underline{w} = \Phi^{-1} \underline{d}$$

(existence of inverse?)

Micchelli's Theorem :

Let $\{\underline{x}_i\}_{i=1}^{N}$ be a set of distinct points in $\mathbb{R}^{m_0}$. Then the $N \times N$ interpolation matrix Φ whose j,i th element is $\varphi_{j,i} = \varphi(\|\underline{x}_j - \underline{x}_i\|)$ is non singular.

Plenty of such functions $\varphi(\cdot)$ exist

Examples:

1) Multiquadrics:

$$\varphi(r) = (r^2 + c^2)^{\frac{1}{2}} \qquad c > 0; \quad r \in \mathbb{R}$$

2) Inverse multiquadrics

$$\varphi(r) = (r^2 + c^2)^{-1/2} \qquad c > 0; \quad r \in \mathbb{R}$$

3) Gaussian functions

$$\varphi(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right) \qquad \sigma > 0; \; r \in \mathbb{R}$$

Inv. multiquadrics & Gaussians are localized i.e., $\varphi(r) \to 0$ as $r \to \infty$

Multiquadrics is unbounded as $r \to \infty$

Radial Basis Function Networks

Ingredients

1) Input Layer: Consists of m_0 source nodes, m_0 is the dimensionality of the input vector $\underline{x}$.

2) Hidden layer: Consists of the same # of computational units as the size of the training samples N; each unit is mathematically described by a radial basis function

$$\varphi_j(\underline{x}) = \varphi\left(\|\underline{x} - \underline{x}_j\|\right); \quad j = 1, 2, \ldots, N$$

$\underline{x}_j$: jth point defines the center of the radial basis function

There are 'n_0' weights connecting source to hidden nodes

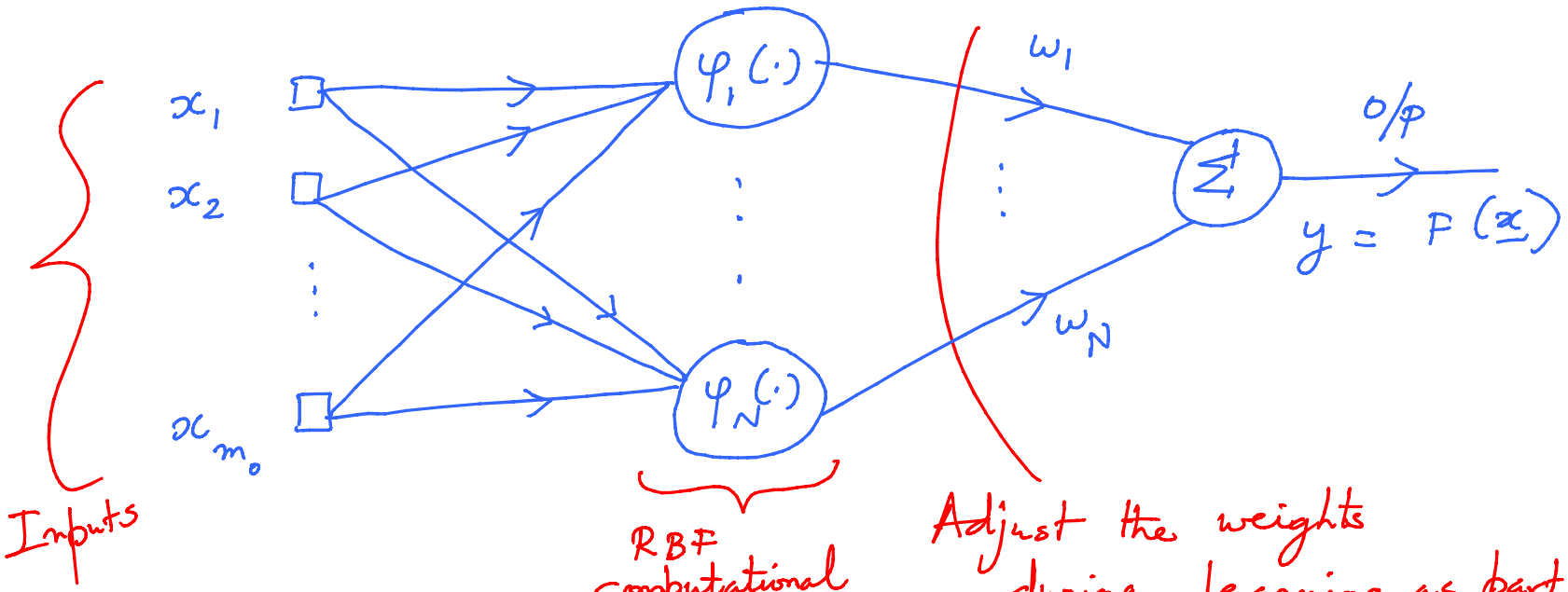
3) O/p Layer

This is a single computational unit. There is no restriction on the size of the o/p layer; (typically) o/p layer size << hidden layer size

$$\varphi_j(\underline{x}) = \varphi(\|\underline{x} - \underline{x}_j\|)$$

$$= \exp\left(-\frac{1}{2\sigma_j^2}\|\underline{x} - \underline{x}_j\|^2\right); \quad j = 1, 2, \ldots, N$$

Sketch of the RBF n/w architecture



This is not like a traditional neuron with an activation function over a local receptive field as in a MLP

Hybrid learning procedure for RBFs

Idea:

1) Obtain the hidden layer elements i.e., the centers in each of the computational units through a clustering algorithm. (We do not need every data point to be a center of an RBF unit)
2) Solve for the optimal weight W linking the hidden layer and the o/p layer

Sketch of the algo

I/p layer : The size of the i/p layer is determined by the dim. of the i/p vector $\underline{x}$, say m_0.

Hidden Layer : The size of the hidden layer m_1 is determined by the # clusters which is a trade-off between performance & complexity

1) Using an algorithm such as the K-means, obtain the cluster mean $\{\hat{\underline{\mu}}_j\}_{j=1}^{K}$ based on the inputs $\{\underline{x}_i\}_{i=1}^{N}$. Compute the $\varphi_j(\cdot)$ using these means for all the data points. (φ_j is based on the Gaussians)

2) Typically, the same σ is applied to all Gaussians

$$\sigma \sim \frac{d_{max}}{\sqrt{2K}} \quad \text{where} \quad d_{max} := \max_{i,j} \|\hat{\underline{\mu}}_i - \hat{\underline{\mu}}_j\|$$

(empirical rule)

The above choice of σ ensures that individual Gaussians are not too 'peaky' or 'flat'. (Heuristic).

3) After we obtain the hidden layer, obtain $\phi(\underline{x}_i) = \begin{bmatrix} \varphi(\underline{x}_i, \hat{\underline{\mu}}_1) \\ \vdots \\ \varphi(\underline{x}_i, \hat{\underline{\mu}}_K) \end{bmatrix}$ over all $i = 1$ to N

From $\{(\phi(\underline{x}_i), d_i)\}_{i=1}^{N}$, obtain $\hat{\underline{w}} = [w_1 \dots w_K]^T$

by solving $\boxed{\phi \hat{\underline{w}} = \underline{d}}$

(If ϕ is a rectangular matrix, obtain the pseudo-inverse ϕ^{-1} d is

Typically, the direct solution of $\hat{\underline{w}}$ using ϕ^{-1} d is avoided by using adaptive algorithms

Following our earlier notation,

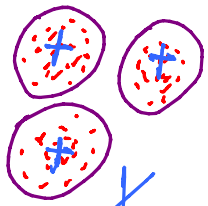
$$\phi^T(\underline{x}_i) = \left[\varphi(\underline{x}_i, \underline{\mu}_1), \cdots, \varphi(\underline{x}_i, \underline{\mu}_K)\right]$$

$$\therefore \quad \phi^T(\underline{x}_i)\,\underline{w} = d_i \; ; \text{ where } \underline{w} = \left[w_1 \cdots w_K\right]^T \quad \text{①}$$

Premultiply ① by $\phi(\underline{x}_i)$ on both sides and sum up from $i = 1$ to N ②

$$\underbrace{\sum_{i=1}^{N} \phi(\underline{x}_i)\,\phi^T(\underline{x}_i)}_{R}\,\underline{w} = \underbrace{\sum_{i=1}^{N} \phi(\underline{x}_i)\,d_i}_{\underline{r}}$$

$$R\,\underline{w} = \underline{r} \; ; \quad \underline{w} = R^{-1}\,\underline{r} \quad \text{③}$$



K-means clustering

K-means is a simple idea for clustering. It is unsupervised in nature

GOAL: Given a set of N observations $\{\underline{x}_i\}_{i=1}^{N}$, find a Codebook C that assigns these observations to K clusters in such a way that the average measure of distortion is minimized from the cluster mean

$$J(C) = \sum_{j=1}^{K} \sum_{C(i)=j} \| \underline{x}_i - \hat{\underline{\mu}}_j \|^2 \quad i = 1, \ldots, N$$

$\hat{\underline{\mu}}_j$: Estimated mean for cluster 'j'

Except for a scaling factor of N_j where N_j is the # data points $\in$ cluster 'j',
$J(C)$ is a measure of overall cluster variance.

Now, how do we minimize $J(C)$

Approach: Use the familiar gradient descent approach

Step 1: For a given code book C, the total cluster variance is minimized w.r.t the assigned set of cluster means $\{\hat{\underline{\mu}}_j\}_{j=1}^{K}$

i.e., $\min_{\{\hat{\underline{\mu}}_j\}_{j=1}^{K}} \sum_{j=1}^{K} \sum_{C(i)=j} \| \underline{x}_i - \hat{\underline{\mu}}_j \|^2$

$i = 1, \ldots, N$

Step 2: Having computed $\{\hat{\underline{\mu}}_j\}_{j=1}^{K}$ in Step 1, optimize the encoder as

$$C(i) = \arg\min_{1 \le j \le K} \| \underline{x}_i - \hat{\underline{\mu}}_j \|^2$$

Iterate Steps 1 and 2 until convergence

Recursive Least Squares Algo

Recall that $R\,\underline{w} = \underline{r}$

Suppose we are estimating w i.e., $\hat{w}$ as a function of data points in an online manner

$$R(n) = \sum_{i=1}^{n} \phi(\underline{x}_i)\,\phi^T(\underline{x}_i)$$

where $\phi(\underline{x}_i) = \left[\varphi(\underline{x}_i, \underline{\mu}_1), \ \cdots \ \varphi(\underline{x}_i, \underline{\mu}_K)\right]^T$

$$\varphi(\underline{x}_i, \underline{\mu}_j) = \exp\left(-\frac{1}{2\sigma_j^2}\,\|\underline{x}_i - \underline{\mu}_j\|^2\right)$$

$j = 1, \cdots, K$

K: # clusters

The $K \times 1$ cross correlation vector is

$$\underline{r}(n) = \sum_{i=1}^{n} \underbrace{\phi(\underline{x}_i)}_{\text{hidden response}} \underbrace{d(i)}_{\text{desired response at the o/p of RBF n/w}}$$

$\underline{\hat{w}}(n)$ needs to be optimized in the least square sense

(Why: $R^{-1} \sim O(K^3)$ — complex

It is computationally difficult for large K

i.e., We need an algo to overcome the inversion issue towards computational efficiency

$$\underline{r}(n) = \sum_{i=1}^{n-1} \underline{\phi}(\underline{x}_i)\, d(i) + \underline{\phi}(\underline{x}_n)\, d(n)$$

$$\underline{r}(n) = \underline{r}(n-1) + \underline{\phi}(\underline{x}_n)\, d(n)$$

$$\underline{r}(n) = R(n-1)\, \hat{\underline{w}}(n-1) + \underline{\phi}(\underline{x}_n)\, d(n)$$

Let us expand $R(n-1)$ further; Let us subsume $\underline{x}_n$ in $\underline{\phi}(\underline{x}_n)$ as $\underline{\phi}(n)$

$$\underline{r}(n) = \left[R(n-1) + \underline{\phi}(n)\, \underline{\phi}^T(n)\right] \hat{\underline{w}}(n-1) + \underline{\phi}(n)\left[d(n) - \underline{\phi}^T(n)\, \hat{\underline{w}}(n-1)\right]$$

i.e., adding & subtracting $\underline{\phi}(n)\, \underline{\phi}^T(n)\, \hat{\underline{w}}(n-1)$

$$R(n) = R(n-1) + \underline{\phi}(n)\,\underline{\phi}^T(n)$$

Let $\alpha(n) \triangleq d(n) - \underline{\phi}^T(n)\,\underline{\hat{w}}(n-1)$

$$= d(n) - \underline{\hat{w}}^T(n-1)\,\underline{\phi}(n)$$

$\alpha(n)$ is referred to as 'innovation'

Prior estimation error based on old $\underline{\hat{w}}(n-1)$

$$\underline{r}(n) = R(n)\,\underline{\hat{w}}(n-1) + \underline{\phi}(n)\,\alpha(n)$$

$$R(n)\,\underline{\hat{w}}(n) = R(n)\,\underline{\hat{w}}(n-1) + \underline{\phi}(n)\,\alpha(n) \quad \text{(I)}$$

$$\underline{\hat{w}}(n) = \underline{\hat{w}}(n-1) + R^{-1}(n)\,\underline{\phi}(n)\,\alpha(n)$$

Update rule

To solve for R^{-1} appearing in the update rule, we invoke the matrix inversion lemma

Consider the matrix $A = B^{-1} + C D C^T$

$$A^{-1} = B - B C \left(D + C^T B C \right)^{-1} C^T B^T$$

For our set up

$A = R(n)$ $\quad B^{-1} = R(n-1)$

$C = \phi(n)$ $\quad D = 1$

Plugging in,

$$R^{-1}(n) = R^{-1}(n-1) - R^{-1}(n-1)\underline{\phi}(n)\left(1 + \underline{\phi}^T(n) R^{-1}(n-1)\underline{\phi}(n)\right)^{-1} \underline{\phi}^T(n) R^{-1^T}(n-1)$$

$$= R^{-1}(n-1) - \frac{R^{-1}(n-1)\underline{\phi}(n)\underline{\phi}^T(n) R^{-1^T}(n-1)}{1 + \underline{\phi}^T(n) R^{-1}(n-1)\underline{\phi}(n)}$$

Let us define $P(n) \triangleq R^{-1}(n)$

$$R^T(n) = R(n); \quad R^{-1^T}(n-1) = R^{-1}(n-1) = P(n-1)$$

Let $\underline{g}(n) \triangleq R^{-1}(n)\,\underline{\phi}(n)$

gain vector

$\underline{g}(n) = P(n)\,\underline{\phi}(n)$

$$\underline{\hat{w}}(n) = \underline{\hat{w}}(n-1) + \underbrace{\underline{g}(n)\,\alpha(n)}_{\text{gain} \times \text{innovation}}$$

Summary of the RLS Algo

Given $\{\phi(i), d(i)\}_{i=1}^{N}$, do the following

for $n = 1, \cdots, N$

$$P(n) = P(n-1) - \frac{P(n-1)\,\underline{\phi}(n)\,\underline{\phi}^T(n)\,P(n-1)}{1 + \underline{\phi}^T(n)\,P(n-1)\,\underline{\phi}(n)}$$

$$\underline{g}(n) = P(n)\,\underline{\phi}(n)$$

$$\alpha(n) = d(n) - \underline{\hat{w}}(n-1)\,\underline{\phi}(n)$$

$$\underline{\hat{w}}(n) = \underline{\hat{w}}(n-1) + \underline{g}(n)\,\alpha(n)$$

($\alpha(n)$: scalar)

To initialize,

Set $\hat{\underline{w}}(0) = \underline{0}$

$P(0) = \lambda^{-1} I$ ← small +ve constant.

Comparison of RBFs and MLPs

Similarities

1) Both are non-linear layered feed forward networks
2) Both are universal approximators

Differences

1) RBF in basic form has a single hidden layer, where as, an MLP has more than 1 hidden layer

2) The neuronal model is the same in the MLP. For RBF, each unit can have a different μ.
Computations in the hidden layer in a MLP require local gradients. This is not so in a RBF

3) For RBF, hidden layer is non-linear, but o/p layer is linear.
However, both hidden and o/p layers of MLP are non-linear

4) The argument of activation fn in MLP, involves inner product i.e., $\varphi(\underline{w}^T \underline{x} + b)$. In case of RBF, one looks into the Euclidean norm i.e., $\varphi(\|\underline{x}_i - \underline{\mu}_j\|)$
for the j^{th} RBF unit

5) Given the same level of n/w complexity, MLP (?) could provide better accuracy than RBF
RBF is faster than MLP.

Kernel Regression

Motivation : Can we link RBFs to solve the regression problem ?

Let us revisit the kernel regression idea built on density estimation

$$y_i = f(\underline{x}_i) + \varepsilon_i \; ; \quad i = 1, \ldots, N$$

$f(\cdot)$ is unknown

Qn: What is a reasonable estimate of $f(\cdot)$?

If we look into the mean of the observables around a point $\underline{x}$ i.e., confine the observations in a small neighborhood around $\underline{x}$, we can form an estimate for $f(\underline{x})$

$$f(\underline{x}) = E(y|\underline{x}) \quad \text{(conditional mean)}$$

$$= \int_{-\infty}^{\infty} y \, P_{Y}(y|\underline{x}) \, dy$$

$$P_{Y}(y|\underline{x}) = \frac{P_{\underline{x}Y}(\underline{x}, y)}{P_{\underline{x}}(\underline{x})}$$

joint p.d.f of $\underline{x}$, y

marginal of $\underline{x}$

regression function

$$f(\underline{x}) = \frac{\int_{-\infty}^{\infty} y \, P_{\underline{x}\,Y}(\underline{x}, y)\, dy}{P_{\underline{x}}(\underline{x})} \qquad \text{(I)}$$

from $P_Y(y|\underline{x})$

A few points to note

1) Joint density $P_{\underline{x}\,y}(\underline{x}, y)$ is unknown
2) We may need a non parametric estimate

Typically a kernel defined by $K(\underline{x})$ has properties similar to a prob. density function (pdf)

1) Kernel $K(\underline{x})$ is continuous, bounded; and a real function of $\underline{x}$ symmetric about the origin where it attains a max. value e.g., Gaussian kernel

2) Volume under the kernel is unity (Normalization)

$$\int_{\mathbb{R}^m} K(\underline{x})\, d\underline{x} = 1$$

Assuming $\underline{x}_1, \underline{x}_2, \cdots, \underline{x}_N$ are independent and identically distributed random vectors, the

PARZEN ROZENBLATT density estimate of $P_{\underline{x}}(\underline{x})$ is

$$\hat{P}_{\underline{x}}(\underline{x}) = \frac{1}{N h^{m_0}} \sum_{i=1}^{N} K\left(\frac{\underline{x} - \underline{x}_i}{h}\right)$$

$\underline{x}_i$: data point

$\hat{P}_{\underline{x}}(\underline{x})$: estimate

$\underline{x} \in \mathbb{R}^{m_0}$

h: Controls the size (bandwidth) (II)

PROPERTY (BIAS)

If $h = h(N)$ is a function such that

$$\lim_{N \to \infty} h(N) = 0,$$ then

$$\lim_{N \to \infty} E\left[\hat{P}_{\underline{x}}(\underline{x})\right] = P_{\underline{x}}(\underline{x})$$

(Asymptotically unbiased)

Let us formulate the Parzen-Rosenblatt density estimate for the joint pdf $p_{\underline{x}\,y}(\underline{x}, y)$; assuming $(\underline{x}, y)$ pairs are independent and identically distributed

$$\hat{p}_{\underline{x}\,y}(\underline{x}, y) = \frac{1}{N h^{m_0+1}} \sum_{i=1}^{N} K\left(\frac{\underline{x} - \underline{x}_i}{h}\right) K\left(\frac{y - y_i}{h}\right)$$

$\underline{x} \in \mathbb{R}^{m_0}$

$y \in \mathbb{R}$

estimate of the joint density

Consider the numerator of (I)

which can be simplified as

$$\int_{-\infty}^{\infty} y\, \hat{P}_{\underline{x},y}(\underline{x},y)\,dy = \frac{1}{N h^{m_o+1}} \sum_{i=1}^{N} K\left(\frac{\underline{x}-\underline{x}_i}{h}\right) \int_{-\infty}^{\infty} y\, K\left(\frac{y-y_i}{h}\right) dy$$

Let us compute the integral

We need to compute the integral carefully

Consider $\int_{-\infty}^{\infty} y\, K\left(\frac{y - y_i}{h}\right) dy$

Let $z = (y - y_i)/h$ (Change of variable)

$y = y_i + zh$; $dy = h\, dz$

$$\therefore \int_{-\infty}^{\infty} (y_i + zh) K(z)\, h\, dz = h\left[\underbrace{\int_{-\infty}^{\infty} y_i\, K(z)\, dz}_{\text{Term 1}} + \underbrace{\int_{-\infty}^{\infty} zh\, K(z)\, dz}_{\text{Term 2}}\right]$$

Term 1 evaluates to y_i

since $\int_{-\infty}^{\infty} K(z) = 1$ (Normalization)

Term 2 evaluate to 0 since $\int_{-\infty}^{\infty} z\, K(z)\, dz = 0$

Zero mean by assumption

$$\therefore \int_{-\infty}^{\infty} y\, \hat{P}_{\underline{x}}(\underline{x}, y)\, dy = \frac{1}{N h^{m_0+1}} \cdot h^{m_0} \sum_{i=1}^{N} y_i\, K\left(\frac{\underline{x} - \underline{x}_i}{h}\right) \quad \text{(III)}$$

$$\therefore \quad \hat{f}_{reg} = E(y|\underline{x}) = \frac{\int_{-\infty}^{\infty} y \, \hat{P}_{\underline{x}\, y}(\underline{x}, y)\, dy}{P_{\underline{x}}(\underline{x})}$$

Recall! (I)

Using (II) and (III) in (I), we have the Kernel reg. estimator

$$\hat{f}_{reg}(\underline{x}) = \frac{\sum_{i=1}^{N} y_i \, K\left(\frac{\underline{x} - \underline{x}_i}{h}\right)}{\sum_{i=1}^{N} K\left(\frac{\underline{x} - \underline{x}_i}{h}\right)}$$

↖ compact form

NOTE:

"Denominator is "not" zero.

Ponder why?

Nadaraya Watson Regression Estimator

Let us define the normalized weighting function

$$w_{N,i}(\underline{x}) = \frac{K\left(\frac{\underline{x} - \underline{x}_i}{h}\right)}{\sum_{j=1}^{N} K\left(\frac{\underline{x} - \underline{x}_j}{h}\right)}$$

(weight to the 'i' th data point $\underline{x}_i$)

$$\sum_{i=1}^{N} w_{N,i}(\underline{x}) = 1 \quad \forall \ \underline{x}$$

Regression fn

$$\hat{f}_{reg}(\underline{x}) = \sum_{i=1}^{N} w_{N,i}(\underline{x})\, y_i$$

y_i: observable

$w_{N,i}(\underline{x})$: weight depends on $\underline{x}_i$

Weighted average of y- observables.

Link to RBF n/w

Since we assume spherical symmetry for the kernel in RBFs (Gaussian case)

L_2 norm $(\cdot)$ radial symmetry

$$K\left(\frac{\underline{x}-\underline{x}_i}{h}\right) = K\left(\frac{\|\underline{x}-\underline{x}_i\|}{h}\right) \quad \forall\, i$$

Define $\psi_N(\underline{x}, \underline{x}_i) = \dfrac{K\left(\frac{\|\underline{x}-\underline{x}_i\|}{h}\right)}{\sum_{j=1}^{N} K\left(\frac{\|\underline{x}-\underline{x}_j\|}{h}\right)}$; $i = 1, \ldots, N$

weighting fn

Again $\sum_{i=1}^{N} \psi_N(\underline{x}, \underline{x}_i) = 1 \quad \forall \underline{x}$

The regression estimate is a weighted sum of 'N' basis functions $\psi_N(\underline{x}, \underline{x}_i)$

Let $y_i = w_i$ for $i = 1, 2, \ldots, N$ (weights are simply observables)

$$\hat{f}_{reg}(\underline{x}) = \sum_{i=1}^{N} w_i \, \psi_N(\underline{x}, \underline{x}_i)$$

w_i: weight

$\psi_N(\underline{x}, \underline{x}_i)$: RBF basis functions (A)

NOTE:

1) (A) denotes the input/output mapping of a normalized RBF with $0 \leq \varphi_N(\underline{x}, \underline{x}_i) \leq 1 \quad \forall \underline{x}, \underline{x}_i$

2) $\varphi_N(\underline{x}, \underline{x}_i)$ is interpreted as the prob. of an event described by input vector $\underline{x}$ conditioned on $\underline{x}_i$

3) Density est. can be ill-posed & can be made well-posed by regularization.

Kernel functions can be of various forms

E.g. $K(\underline{x}) = \frac{1}{\sqrt{(2\pi)^{m_0}}} \exp\left(-\frac{\|\underline{x}\|^2}{2}\right)$ ← $\sigma^2 = 1$ Gaussian kernel

With a spread parameter σ,

$$K\left(\frac{\underline{x} - \underline{x}_i}{h}\right) = \frac{1}{(2\pi\sigma^2)^{m_0/2}} \exp\left(-\frac{\left\|\frac{\underline{x} - \underline{x}_i}{h}\right\|^2}{2\sigma^2}\right)$$

$i = 1, \ldots, N$

With $h = 1$, following NWRE

$$\hat{f}_{reg}(\underline{x}) = \frac{\sum_{i=1}^{N} y_i \exp\left(-\frac{\|\underline{x} - \underline{x}_i\|^2}{2\sigma^2}\right)}{\sum_{j=1}^{N} \exp\left(-\frac{\|\underline{x} - \underline{x}_j\|^2}{2\sigma^2}\right)}$$

Final form using Gaussian fns.

Basics of constrained optimization

Motivation: We are interested in minimizing / maximizing functions subject to constraints over the variables

Example: 1) Optimize the path from point A to point B, subject to traffic conditions over all connecting roads.

2) Optimize the square error in the MLP subject to a sparsity constraint in the network connectivity across layers etc ...

We have to deal with minimizing functions subject to equality and inequality constraints

Formulation

$$\min_{\underline{x} \in \mathbb{R}^n} f(\underline{x}) \quad \text{s.t.} \quad \begin{cases} c_i(\underline{x}) = 0, & i \in \text{Equality constraints } 'E' \\ c_i(\underline{x}) \geq 0, & i \in \text{Inequality constraints } 'I' \end{cases}$$

$f(\underline{x})$: objective function

$c_i(\underline{x}) \leq 0$

NOTE: The '$\leq$' in the Constraints inequality can be suitably transformed to '$\geq$' by a sign flip

Assumptions / terminology

1) We assume f and c_i's to be smooth and real valued operating on a subset of $\mathbb{R}^n$.

2) f is the objective function and $c_i \in E$, $c_j \in I$

Let $\Omega = \{ \underline{x} \mid c_i(\underline{x}) = 0, c_i \in E, c_j(\underline{x}) \geq 0, j \in I \}$

(Ω: overall constraint set)

Written succintly,

$$\min_{\underline{x} \in \Omega} f(\underline{x}) \qquad \text{(I)}$$

Recall : For an optimal solution $\underline{x}^*$ (Minimization)

$$\nabla f(\underline{x}^*) = \underline{0}$$

and $\nabla^2 f(\underline{x}^*) \geq 0$ (Positive Semi definite property)

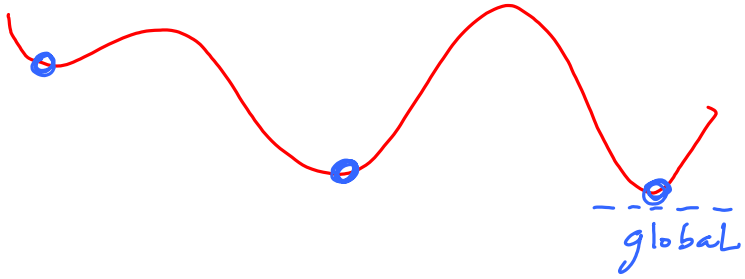
If $\nabla^2 f(\underline{x}^*) > 0$ (+ve definite)

Strict inequality

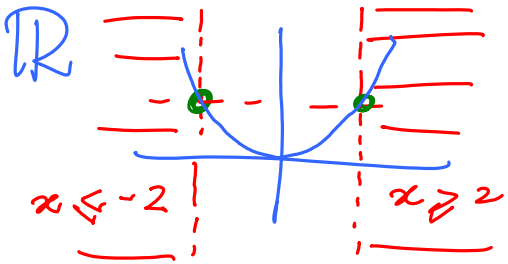
The solutions to opt. problems can have

(a) local solution

(b) global solution



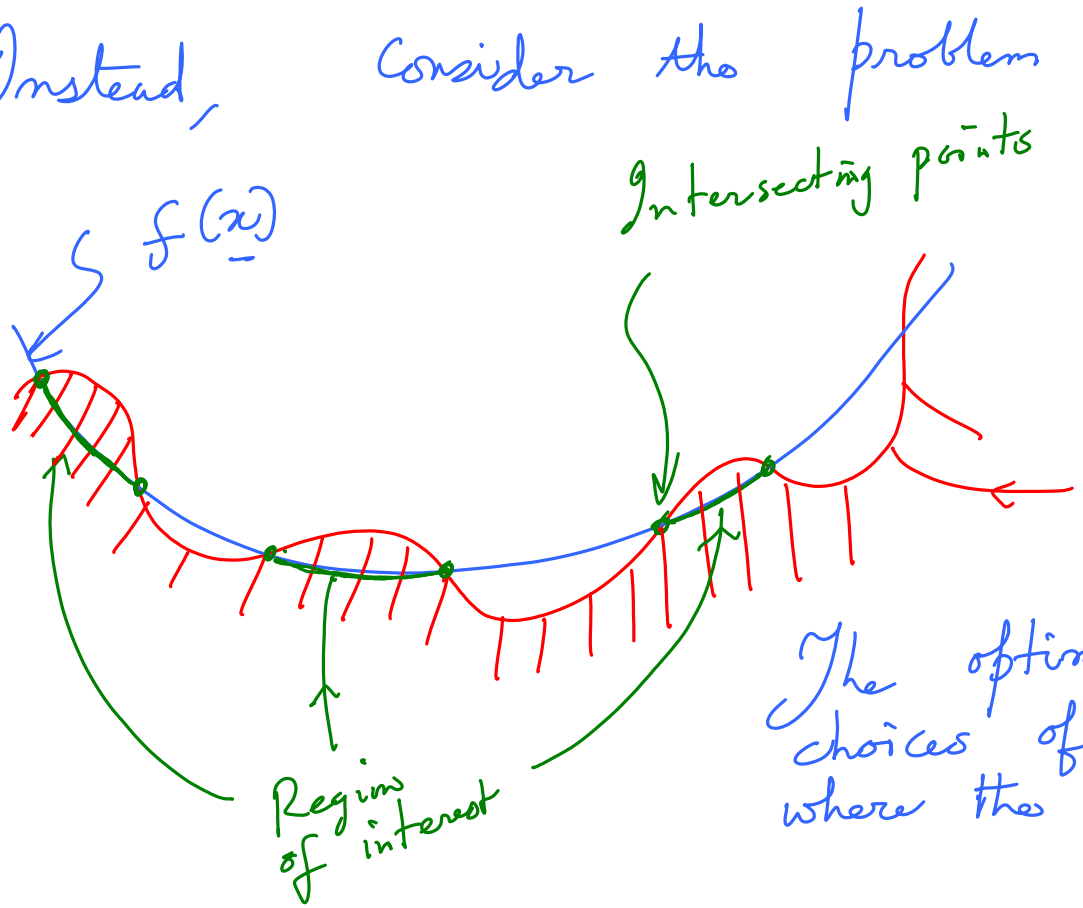
Example: $\min_{x \in \mathbb{R}} x^2$ s.t. $|x| \geq 2$ (constraint)

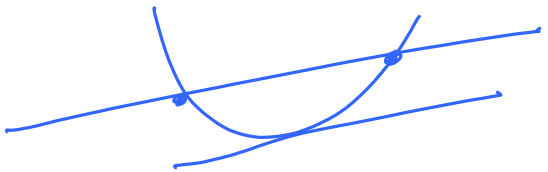


(There are 2 solns & not unique)

No constraint $\Rightarrow$ $x = 0$ trivially unique

Instead, consider the problem





Constraint

$c_i(\underline{x}) \leq 0$

The optimization is over the choices of $\underline{x}$ over the objective where the constraints are satisfied

1) A vector $\underline{x}^*$ is a local solution to problem (I) if $\underline{x}^* \in \Omega$ and there is a neighborhood N of $\underline{x}^*$ such that $f(\underline{x}) \geq f(\underline{x}^*)$ $\forall\, \underline{x} \in N \cap \Omega$

2) A vector $\underline{x}^*$ is a strict local solution if $\underline{x}^* \in \Omega$ and there is a neighborhood N of $\underline{x}^*$ / $f(\underline{x}) > f(\underline{x}^*)$ $\forall\, \underline{x} \in N \cap \Omega$ with $\underline{x} \neq \underline{x}^*$

3) A point $\underline{x}^*$ is an isolated local solution if $\underline{x}^* \in \Omega$ and there is a neighborhood N of $\underline{x}^*$ / $\underline{x}^*$ is the only minimizer in $N \cap \Omega$

Smoothness

Smoothness of objective fns & constraints can help algorithms to make better choices during gradient search towards the soln. (local soln)

Active / Inactive Constraints

At a feasible point $\underline{x}$, the inequality constraint $c_i \in I$ is active if $c_i(\underline{x}) = 0$ & inactive if $c_i(\underline{x}) > 0$