Neural Networks: Shallow Learning

Evgeny Burnaev

Skoltech, Moscow, Russia

Outline

Problem Statement

- Let $y = f(\mathbf{x})$ be some function, which is continuous and defined on a compact $X \subset R^N$, $N \sim 5 - 50$.
- The problem is to construct an approximation $\hat{f}(\mathbf{x})$ using the train sample of size m

$$S_m = \{(\mathbf{x}_i, y_i),\, \mathbf{x}_i \in X,\, f(\mathbf{x}_i) = y_i,\, i = 1, \ldots, m\}$$

- Approximation should be accurate in some sense

$$f(\mathbf{x}) \approx \hat{f}(\mathbf{x}),\ \mathbf{x} \in X \tag{1}$$

Note that (1) should hold for all $\mathbf{x} \in X$, not only for $\mathbf{x} \in S_m$

Linear Expansion in a Functional Dictionary

- A model $\hat{f}(\mathbf{x})$ is composed of functions from some parametric dictionary
$$\hat{f}(\mathbf{x}) = \sum_{j=1}^{p} \alpha_j \psi_j(\boldsymbol{\theta}_j, \mathbf{x}) + \alpha_0$$
Or, in vector notations
$$\hat{f}(\mathbf{x}) = \boldsymbol{\psi}(\Theta, \mathbf{x})\,\boldsymbol{\alpha}, \quad \boldsymbol{\alpha} = \{\alpha_j\}_{j=0}^{p}, \quad \Theta = \{\boldsymbol{\theta}_j\}_{j=1}^{p},$$
where $\boldsymbol{\psi}(\Theta, \mathbf{x}) = (\psi_1(\boldsymbol{\theta}_1, \mathbf{x}), \ldots, \psi_p(\boldsymbol{\theta}_p, \mathbf{x}))$
- Thus $\hat{f}(\mathbf{x})$ is determined by
  - matrix Θ of dictionary functions parameters
  - vector of coefficients $\boldsymbol{\alpha}$ in the linear expansion

  $\psi_j(\mathbf{x})$ can be considered as the j-th transformation of $\mathbf{x}$

Once the dictionary $\{\psi_j(\mathbf{x})\}_{j=1}^{p}$ is determined, the model $\hat{f}(\mathbf{x})$ is linear in these new variables, and the fitting is similar to linear regression

Types of dictionary functions

- Polynomials: $\psi_j\left(\boldsymbol{\theta}_j, \mathbf{x}\right) = \prod_{k=1}^{N} x^{\theta_{j,k}}$, $\theta_j \in \{0,1\}^N$
- Indicators for some regions:

$$\psi_j\left(\boldsymbol{\theta}_j, \mathbf{x}\right) = \prod_{k=1}^{N} 1\{\theta_{k,1} \le x_k \le \theta_{k,2}\}$$

- Sigmoid function:

$$\psi_j\left(\boldsymbol{\theta}_j, \mathbf{x}\right) = \sigma\left(\mathbf{x}^{\mathrm{T}}\boldsymbol{\theta}_j^1 + \theta_j^0\right),$$

where $\sigma(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ or $\sigma(z) = \frac{1}{1+e^z}$

- Gaussian function:

$$\psi_j\left(\boldsymbol{\theta}_j, \mathbf{x}\right) = \exp\left(-\frac{\|\mathbf{x} - \boldsymbol{\theta}_j^1\|^2}{(\theta_j^0)^2}\right)$$

A dictionary can include other types of functions like trigonometric functions, etc.

Objective Measure: Error Function

- As mentioned above, the approximation should be close to the original function:

$$f(\mathbf{x}) \approx \hat{f}(\mathbf{x}),\ \mathbf{x} \in X$$

- Quantitative measure of closeness is the error function

$$\hat{R} = \hat{R}\left(S_m, \hat{f}\right) = \frac{1}{2}\sum_{i=1}^{m}\left(y_i - \hat{f}(\mathbf{x}_i)\right)^2,\ S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$$

In the sequel we denote by $\#S$ a cardinality of a set S (number of points in the sample S)

Algorithm

1. Select dictionary size p
2. Initialize dictionary functions parameters Θ and linear expansion coefficient $\boldsymbol{\alpha}$
3. Minimize the error function (Empirical Risk Minimization problem) $\hat{R}\left(S_m, \hat{f}\right) = \hat{R}\left(\Theta, \boldsymbol{\alpha}\right)$

Algorithm

1. Select dictionary size p
2. Initialize dictionary functions parameters Θ and linear expansion coefficient $\boldsymbol{\alpha}$
3. Minimize the error function (Empirical Risk Minimization problem) $\hat{R}\left(S_m, \hat{f}\right) = \hat{R}\left(\Theta, \boldsymbol{\alpha}\right)$

In this presentation

1. Some connections to well-known models in statistics
2. Specific features of the ERM problem
3. Incorporate these features into the ERM algorithm to
   - Increase accuracy of approximation
   - Reduce training time

Additive Models I

- We try to fit a regression function

$$\hat{f}(\mathbf{x}) = \mathbb{E}(y|\mathbf{x}) = f(x_1, \dots, x_N),$$

  in which every level of interaction is potentially present
- It is natural to consider analysis-of-variance (ANOVA) decompositions of the form

$$\hat{f}(x_1, \dots, x_N) = \alpha_0 + \sum_j g_j(x_j) + \sum_{k<r} g_{kr}(x_k, x_r) + \dots$$

- In order to restrict a model class we consider additive models, containing only main effect terms

$$\hat{f}(\mathbf{x}) = \alpha_0 + \sum_{j=1}^{N} g_j(\mathbf{x}_j)$$

Additive Models II

Usually backfitting procedure is used:

1. Initialize $g_1(x_1), \dots, g_N(x_N)$
2. For $j = 1, \dots, N$
   (A) Calculate residuals $\epsilon_{j,i} = y_i - \sum_{s \neq j} g_s(x_{s,i})$
   (B) Fit $g_j(x_j)$ using the sample $S_j = \{(x_{j,i}, \epsilon_{j,i})\}_{i=1}^m$
3. If converged STOP. Else, go back to step 2

Projection Pursuit Regression I

- Let $\{\boldsymbol{\theta}_j\}_{j=1}^p$ be unit N-vectors of unknown parameters
- The projection pursuit regression model has the form

$$\hat{f}(\mathbf{x}) = \sum_{j=1}^{p} g_j(\boldsymbol{\theta}_j^{\mathrm{T}}\mathbf{x})$$

- This is an additive model, but in the derived features $z_j = \boldsymbol{\theta}_j^{\mathrm{T}}\mathbf{x}$
- The functions g_j are unspecified and are estimated. Since

$$g(\boldsymbol{\theta}^{\mathrm{T}}\mathbf{x}) \approx g(\boldsymbol{\theta}_{\mathrm{old}}^{\mathrm{T}}\mathbf{x}) + g'(\boldsymbol{\theta}_{\mathrm{old}}^{\mathrm{T}}\mathbf{x})(\boldsymbol{\theta} - \boldsymbol{\theta}_{\mathrm{old}}^{\mathrm{T}})\mathbf{x},$$

then

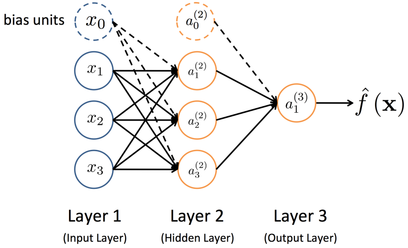
$$\sum_{i=1}^{m}[y_i - g(\boldsymbol{\theta}^{\mathrm{T}}\mathbf{x}_i)]^2 \approx$$

$$\sum_{i=1}^{m} g'(\boldsymbol{\theta}_{\mathrm{old}}^{\mathrm{T}}\mathbf{x}_i)^2 \left[\left(\boldsymbol{\theta}_{\mathrm{old}}^{\mathrm{T}}\mathbf{x}_i + \frac{y_i - g(\boldsymbol{\theta}_{\mathrm{old}}^{\mathrm{T}}\mathbf{x}_i)}{g'(\boldsymbol{\theta}_{\mathrm{old}}^{\mathrm{T}}\mathbf{x}_i)}\right) - \boldsymbol{\theta}^{\mathrm{T}}\mathbf{x}_i\right]^2 \tag{2}$$

Projection Pursuit Regression II

- We fit the model by an iterative process:
  - a) Optimize (2) to update $\boldsymbol{\theta}$ (quadratic optimization!)
  - b) Tune $g(\cdot)$ by smoothing current residuals
  - c) Repeat steps a)-b) until convergence
- Fit a new term $g(\boldsymbol{\theta}_{\mathrm{new}}^{\mathrm{T}}\mathbf{x})$ to the residuals, etc.

Neural Network with Two-Layers I



- $a_i^{(j)}$ = "activation" of unit i in layer j
- $\Theta^{(j)}$ = weight matrix controlling function mapping from layer j to layer $j+1$

$$a_1^{(2)}(\mathbf{x}) = \sigma\left(\theta_{10}^{(1)} + \theta_{11}^{(1)}x_1 + \theta_{12}^{(1)}x_2 + \theta_{13}^{(1)}x_3\right) = \sigma((\boldsymbol{\theta}_1^{(1)})^{\mathrm{T}}\mathbf{x})$$

$$a_2^{(2)}(\mathbf{x}) = \sigma\left(\theta_{20}^{(1)} + \theta_{21}^{(1)}x_1 + \theta_{22}^{(1)}x_2 + \theta_{23}^{(1)}x_3\right) = \sigma((\boldsymbol{\theta}_2^{(1)})^{\mathrm{T}}\mathbf{x})$$

$$a_3^{(2)}(\mathbf{x}) = \sigma\left(\theta_{30}^{(1)} + \theta_{31}^{(1)}x_1 + \theta_{32}^{(1)}x_2 + \theta_{33}^{(1)}x_3\right) = \sigma((\boldsymbol{\theta}_3^{(1)})^{\mathrm{T}}\mathbf{x})$$

$$\hat{f}(\mathbf{x}) = a_1^{(3)}(a_1^{(2)}, a_2^{(2)}, a_3^{(2)}) = \theta_{10}^{(2)} + \theta_{11}^{(2)}a_1^{(2)} + \theta_{12}^{(2)}a_2^{(2)} + \theta_{13}^{(2)}a_3^{(2)}$$

Here we assume that bias units $a_0^{(2)} = 1$ and $x_0 = 1$, and that $\sigma(\cdot)$ is a sigmoid function

Neural Network with Two-Layers II

- NN with two layers and p hidden units **coincides** with a basis expansion in a dictionary with p basis functions in case

$$\psi(\boldsymbol{\theta}_j, \mathbf{x}) = \sigma(\boldsymbol{\theta}_j^{\mathrm{T}} \mathbf{x})$$

  and $\alpha_j = \theta_{1,j}^{(2)}$, $j = 0, 1, \ldots, p$

- NN with two layers and p hidden units is similar to the projection pursuit regression when

$$g_j(\boldsymbol{\theta}_j^{\mathrm{T}} \mathbf{x}) = \sigma(\boldsymbol{\theta}_j^{\mathrm{T}} \mathbf{x}),$$

  i.e. we consider pre-determined $g_j(\cdot) = \sigma(\cdot)$

Learning of Basis Expansions

Further

- We consider approximation models, defined by basis expansions in nonlinear parametric dictionaries

$$\hat{f}(\mathbf{x}) = \sum_{j=1}^{p} \alpha_j \psi_j(\boldsymbol{\theta}_j, \mathbf{x}) + \alpha_0$$

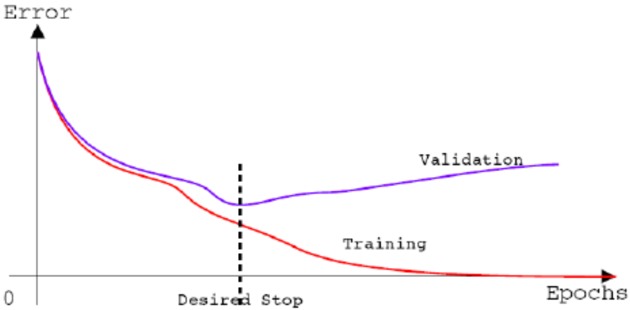
- We provide hints how to take a structure of the ERM problem into account in order to perform minimization efficiently

$$\hat{R}\left(S_m, \hat{f}\right) = \hat{R}(\Theta, \boldsymbol{\alpha}) = \frac{1}{2}\sum_{i=1}^{m}\left(y_i - \hat{f}(\mathbf{x}_i)\right)^2,\ S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$$

Early Stopping I

Let S_{test} be some test sample, generated in X. Empirical indication of the overfitting is

$$\hat{R}\left(S_{test}, \hat{f}\right) \gg \hat{R}\left(S_m, \hat{f}\right)$$



Early Stopping II

Let S_{test} be some test sample, generated in X. Empirical indication of the overfitting is

$$\hat{R}\left(S_{test}, \hat{f}\right) \gg \hat{R}\left(S_m, \hat{f}\right)$$

Early stooping prevents overfitting:

Adopted algorithm

1. Divide S_m into S_{train} and S_{val}
2. Select dictionary size p
3. Initialize dictionary functions parameters Θ and linear expansion coefficients $\boldsymbol{\alpha}$
4. Minimize the error function $\hat{R}\left(S_{train}, \hat{f}\right) = \hat{R}\left(\Theta, \boldsymbol{\alpha}\right)$.
5. Stop the optimization process when $\hat{R}\left(S_{val}, \hat{f}\right)$ stops to decrease

Separability of Variables

Let us consider the error function $\hat{R}(\Theta, \boldsymbol{\alpha})$:

$$\hat{R}(\Theta, \boldsymbol{\alpha}) = (\boldsymbol{\Psi}(\Theta)\boldsymbol{\alpha} - \mathbf{y})^{\mathrm{T}} (\boldsymbol{\Psi}(\Theta)\boldsymbol{\alpha} - \mathbf{y}),$$

where

- $\mathbf{y} = \{y_1, \dots, y_m\}$,
- $\boldsymbol{\Psi}(\Theta) = (\boldsymbol{\psi}(\Theta, \mathbf{x}_1), \dots, \boldsymbol{\psi}(\Theta, \mathbf{x}_m))$. Here and further $m = \#(S_{train})$ is a size of the training set

We can easily see that

- Dependence of $\hat{R}$ on Θ is nonlinear
- Dependence of $\hat{R}$ on $\boldsymbol{\alpha}$ is quadratic

We can find the optimal value of $\boldsymbol{\alpha}$ using least squares approach:

$$\boldsymbol{\alpha} = \boldsymbol{\alpha}(\Theta)$$

Possible explicit formulas for calculating optimal $\boldsymbol{\alpha}(\Theta)$

1. Least squares estimate

$$\boldsymbol{\alpha}(\Theta) = \left(\Psi(\Theta)^{\mathrm{T}} \Psi(\Theta)\right)^{-1} \Psi(\Theta)^{\mathrm{T}} \mathbf{y}$$

2. Ridge regression

$$\boldsymbol{\alpha}(\Theta) = \left(\Psi(\Theta)^{\mathrm{T}} \Psi(\Theta) + \lambda \mathrm{I}_p\right)^{-1} \Psi(\Theta)^{\mathrm{T}} \mathbf{y}$$

3. Ridge regression with smoothness penalty:

$$\boldsymbol{\alpha}(\Theta) = \left(\Psi(\Theta)^{\mathrm{T}} \Psi(\Theta) + \lambda \Gamma(\Theta)^{\mathrm{T}} \Gamma(\Theta)\right)^{-1} \Psi(\Theta)^{\mathrm{T}} \mathbf{y},$$

where $\Gamma(\Theta)$ is a some functional of derivatives of the approximation

Using of separability of variables

1. Calculate derivatives $\hat{R}_\Theta$ and $\hat{R}_{\Theta\Theta}$ of the error function
2. Calculate step size $\boldsymbol{d}_k$ by some optimization algorithm using gradient $\hat{R}_\Theta$ and hessian $\hat{R}_{\Theta\Theta}$
3. Update the matrix of dictionary functions parameters: $\Theta_{k+1} = \Theta_k + \boldsymbol{d}_k$
4. Update linear expansion coefficients $\boldsymbol{\alpha}\left(\Theta_{k+1}\right)$, applying e.g. LSQ formula

We should take into account dependence of $\boldsymbol{\alpha}\left(\Theta\right)$ on Θ in step 1!

Separability of Variables

Consider a new objective function

$$\hat{R}(\Theta) = \hat{R}(\Theta, \boldsymbol{\alpha}(\Theta)), \; \boldsymbol{\alpha}(\Theta) = \left(\Psi(\Theta)^{\mathrm{T}} \Psi(\Theta)\right)^{-1} \Psi(\Theta)^{\mathrm{T}} \mathbf{y}$$

Let us calculate derivatives of this objective function

- By the definition of the least squares method

$$\hat{R}_\alpha = \mathbf{0}$$

- Gradient of $\hat{R}(\Theta)$

$$\hat{R}_\Theta = \hat{R}_\Theta + \hat{R}_\alpha \alpha_\Theta = \hat{R}_\Theta + \mathbf{0}\alpha_\Theta = \hat{R}_\Theta$$

- Hessian

$$\hat{R}_{\Theta\Theta} = \hat{R}_{\Theta\Theta} + \hat{R}_{\Theta\alpha}\, \alpha_\Theta$$

1. We can obtain $\hat{R}_{\Theta\alpha}$ and $\hat{R}_{\Theta\Theta}$ straightforwardly

$$\hat{R}_{\Theta\alpha} = \left(\mathbf{e}^{\mathrm{T}}\Psi\right)_{\Theta} = \mathbf{e}^{\mathrm{T}} \odot \Psi_{\Theta} + \mathbf{e}_{\Theta}^{\mathrm{T}}\Psi = \mathbf{e}^{\mathrm{T}} \odot \Psi_{\Theta} + J^{\mathrm{T}}\Psi$$

$$\hat{R}_{\Theta\Theta} = J^{\mathrm{T}}J + \mathbf{e} \odot \hat{f}_{\Theta\Theta}(\mathbf{X})$$

where
- $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$
- $J \overset{\mathrm{def}}{=} \mathbf{e}_{\Theta} = \hat{f}_{\Theta}(\mathbf{X})$
- $\odot$ is pseudo-multiplication of $3D$ matrix:
  $\boldsymbol{e}^{\mathrm{T}} \odot \hat{f}_{\theta\theta}(\mathbf{X}) = \sum_{i=1}^{m} e_i \hat{f}_{\theta\theta}(\mathbf{x}_i)$

2. Direct computation of α_{Θ} is rather lengthy

$$\hat{R}_{\alpha} = 0 \;\Rightarrow\; d\hat{R}_{\alpha} = \hat{R}_{\alpha\alpha}d\alpha + \hat{R}_{\alpha\Theta}d\Theta \equiv \mathbf{0},$$

$$\alpha_{\Theta} = \frac{d\alpha}{d\Theta} = -(\hat{R}_{\alpha\alpha})^{-1}\hat{R}_{\alpha\Theta},$$

$$\left(\text{and } \hat{R}_{\alpha\alpha} = \Psi^{\mathrm{T}}\Psi\right)$$

Final formula for the hessian of $\hat{R}$

- Hessian of $\hat{R}$
$$\hat{R}_{\Theta\Theta} = \hat{R}_{\Theta\Theta} + \hat{R}_{\Theta\alpha}\alpha_{\Theta}$$
- After substitution:
$$\hat{R}_{\Theta\Theta} = J^{\mathrm{T}}J + \mathbf{e} \odot \hat{f}_{\Theta\Theta}(\mathbf{X}) - \left(\mathbf{e}^{\mathrm{T}} \odot \Psi_{\Theta} + J^{\mathrm{T}}\Psi\right)\left(\Psi^{\mathrm{T}}\Psi\right)^{-1}\left(\mathbf{e}^{\mathrm{T}} \odot \Psi_{\Theta} + J^{\mathrm{T}}\Psi\right)^{\mathrm{T}}$$
- Let us neglect the terms with residuals $\mathbf{e}$ ($\mathbf{e} \approx \mathbf{0}$)
$$\hat{R}_{\Theta\Theta} \approx J^{\mathrm{T}}J - \left(J^{\mathrm{T}}\Psi\right)\left(\Psi^{\mathrm{T}}\Psi\right)^{-1}\left(J^{\mathrm{T}}\Psi\right)^{\mathrm{T}}$$

LSQ isn't a good idea!

- Matrix $\Psi\left(\Theta\right)^{\mathrm{T}}\Psi\left(\Theta\right)$ can be ill-conditioned
- LSQ estimates are unbiased but can have high variance
- In our case high variance $\equiv$ unstable optimization process

Let us use a ridge regression to estimate linear expansion coefficients $\boldsymbol{\alpha}$

$$\boldsymbol{\alpha}\left(\Theta\right) = \left(\Psi\left(\Theta\right)^{\mathrm{T}}\Psi\left(\Theta\right) + \lambda \mathrm{I}_p\right)^{-1}\Psi\left(\Theta\right)^{\mathrm{T}}\mathbf{y}$$

Estimating $\boldsymbol{\alpha}$ with this formula is equivalent to minimization of

$$\widetilde{R}\left(\Theta, \boldsymbol{\alpha}\right) = \frac{1}{2}\Big(\left(\mathbf{y} - \Psi\boldsymbol{\alpha}\right)\left(\mathbf{y} - \Psi\boldsymbol{\alpha}\right) + \lambda\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{\alpha}\Big)$$

w.r.t. $\boldsymbol{\alpha}$

Let us apply the same approach to the new objective function

- We consider the modified error function $\widetilde{R}$

$$\widetilde{R}\left(\Theta\right) = \widetilde{R}\left(\Theta, \boldsymbol{\alpha}\left(\Theta\right)\right)$$

- Note that $\widetilde{R}_{\alpha} = \mathbf{0}$, therefore we can use a general representation of the hessian matrix

$$\widetilde{R}_{\Theta\Theta} = \widetilde{R}_{\Theta\Theta} + \widetilde{R}_{\Theta\alpha}\alpha_{\Theta}$$

- Partial derivatives with respect to Θ are the same

$$\widetilde{R}_{\alpha\Theta} = \hat{R}_{\alpha\Theta}, \quad \widetilde{R}_{\Theta\Theta} = \hat{R}_{\Theta\Theta}$$

- A final formula

$$\widetilde{R}_{\Theta\Theta} \approx J^{\mathrm{T}} J - \left(J^{\mathrm{T}}\Psi\right)\left(\Psi^{\mathrm{T}}\Psi + \lambda \mathrm{I}_{p}\right)^{-1}\left(J^{\mathrm{T}}\Psi\right)^{\mathrm{T}}$$

Let us compare two objective functions:

1. Initial error function:

$$\hat{R}\left(\Theta, \boldsymbol{\alpha}\right) = \frac{1}{2}\left(\mathbf{y} - \Psi\boldsymbol{\alpha}\right)^{\mathrm{T}}\left(\mathbf{y} - \Psi\alpha\right)$$

2. Modified error function, based on explicit ridge regression estimate

$$\widetilde{R}\left(\Theta\right) = \frac{1}{2}\left(\mathbf{y} - \Psi\alpha\left(\Theta\right)\right)^{\mathrm{T}}\left(\mathbf{y} - \Psi\alpha\left(\Theta\right)\right) + \frac{1}{2}\lambda\alpha\left(\Theta\right)^{\mathrm{T}}\alpha\left(\Theta\right),$$

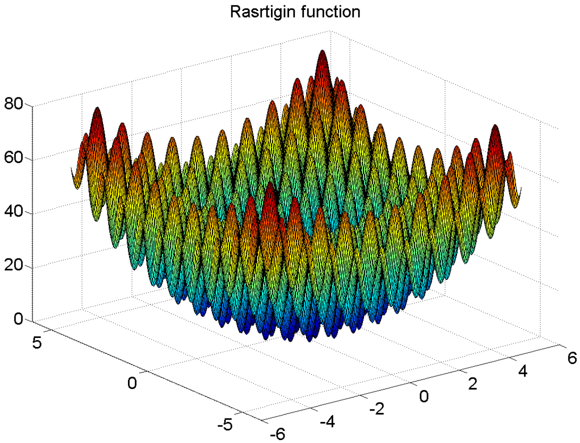
$$\alpha\left(\Theta\right) = \left(\Psi\left(\Theta\right)^{\mathrm{T}}\Psi\left(\Theta\right) + \lambda\mathrm{I}_p\right)^{-1}\Psi\left(\Theta\right)^{\mathrm{T}}\mathbf{y}$$

Also we assume that $\mathbf{e} \approx \mathbf{0}$ in this case

We use the trust region method with a modified More-Sorensen method to solve a restricted quadratic minimization problem

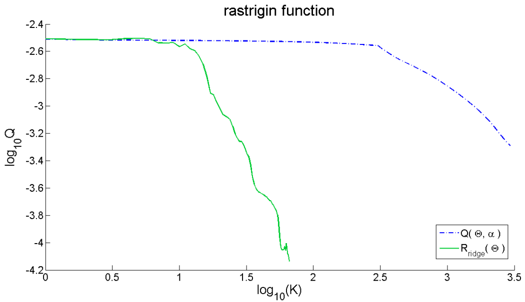
Toy example: Rastrigin function I

$$f(\mathbf{x}) = An + \sum_{i=1}^{N} \left(x_i^2 - A\cos\left(2\pi x_i\right)\right), \ A = 10, \ \mathbf{x} \in [-5.12, 5.12]^N$$



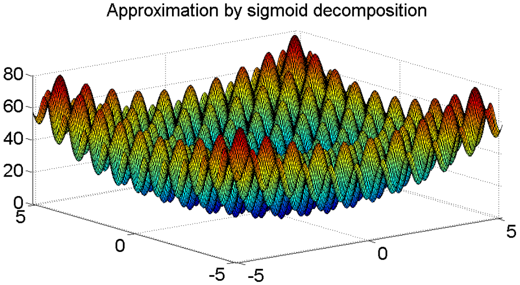
Toy example: Rastrigin function II

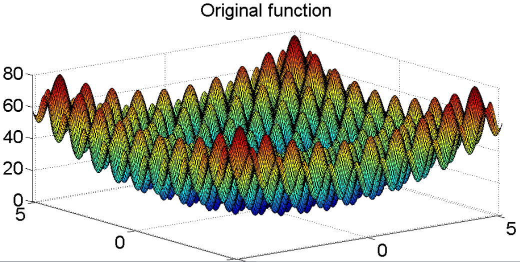
$m = 1000$ points, a dictionary consists of $p = 128$ sigmoids



	$\hat{R}(\Theta, \boldsymbol{\alpha})$	$\widetilde{R}(\Theta)$
$\hat{R}\left(S_{test}, \hat{f}\right)$	2,99E-02	5,48E-05
$Iterations$	1532	87

Toy example: Rastrigin function III





Toy example: Rastrigin function IV

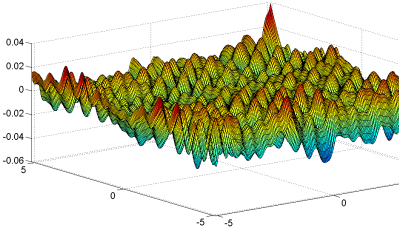


Figure : Residuals

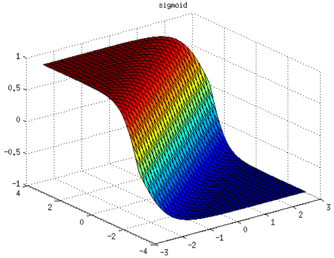


Figure : Sigmoid

Range of the original function is [0, 80], range of residuals is [-0.06, 0.04].

Average results

- "Time" provides values of learning time for approximation construction
- We calculate a square normalized error on some test sample S_{test}
- Ratio (for one task):

$$timeRatio(task) = \frac{\text{basic training algorithm}}{\text{improved training algoritms}}$$

- We consider 26 artificial tasks for each sample size m

m	median timeRatio	median errorRatio
160	1.9	1.45
320	2.6	1.39
1000	3.2	1.54

Hessian additivity

Let us consider the error function $\widetilde{R}(\Theta)$ and its hessian

$$\widetilde{R}(\Theta) = \widetilde{R}(\Theta, \boldsymbol{\alpha}(\Theta)) = \frac{1}{2}\Big((\mathbf{y} - \Psi\boldsymbol{\alpha}(\Theta))^{\mathrm{T}}(\mathbf{y} - \Psi\boldsymbol{\alpha}(\Theta)) \\ + \lambda\boldsymbol{\alpha}(\Theta)^{\mathrm{T}}\boldsymbol{\alpha}(\Theta)\Big)$$

$$\widetilde{R}_{\Theta\Theta}(\Theta) \approx \mathcal{H} \stackrel{\mathrm{def}}{=} J^{\mathrm{T}}J - \left(J^{\mathrm{T}}\Psi\right)\left(\Psi^{\mathrm{T}}\Psi + \lambda \mathrm{I}_p\right)^{-1}\left(J^{\mathrm{T}}\Psi\right)^{\mathrm{T}},$$

$J = J(\Theta)$ are derivatives $\hat{f}_{\Theta}(\mathbf{x})$ in sample points $\mathbf{x} \in S_m$
Note, that $\mathcal{H}$ is a sum of "sub-hessians" for training sample points:

$$\mathcal{H} = \sum_{i=1}^{m} J_i^{\mathrm{T}}J_i - \left(J_i^{\mathrm{T}}\Psi\right)\left(\Psi^{\mathrm{T}}\Psi + \lambda \mathrm{I}_p\right)^{-1}\left(J_i^{\mathrm{T}}\Psi\right)^{\mathrm{T}} = \sum_{i=1}^{m} h(\mathbf{x}_i, \Theta)$$

Computational complexity of

- $\mathcal{H}$ calculation is $\sim m\left(pN\right)^2$,
- $\mathcal{H}$ inversion (or Cholesky decomposition) is $\sim\left(pN\right)^3$.

where

- m is the training sample size,
- N is a dimensionality of $\mathbf{x}$,
- p is a number of functions in the dictionary

If $m \gg pN$ then computational complexity of the hessian calculation is significantly higher than of the hessian inversion

Let us define subsample $S_{subtrain} \subset S_{train}$ whith size $\hat{m} \ll m$
Now we can calculate approximation of the true hessian $\mathcal{H}$ using only points from $S_{subtrain}$

$$\widehat{\mathcal{H}} = \sum_{i=1}^{\hat{m}} h\left(\mathbf{x}_i, \Theta\right), \mathbf{x}_i \in S_{subtrain},$$

$$\widetilde{R}_{\Theta,\Theta} \approx \mathcal{H} \approx \widehat{\mathcal{H}}$$

How to select $S_{subtrain}$ from S_{train}?

1. In order to approximate diagonal of the hessian with a maximal accuracy:

$$\operatorname{diag}\left(\widehat{\mathcal{H}}\right) \approx \operatorname{diag}\left(\mathcal{H}\right)$$

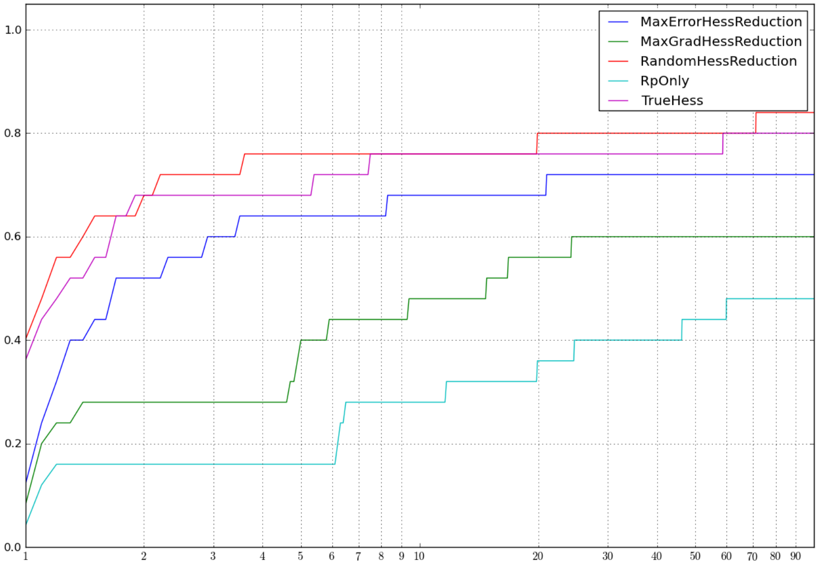
2. Include points with maximal residuals in $S_{subtrain}$
3. Uniformly randomly select $S_{subtrain}$ (we use a new subsample for each iteration of the learning process)

Dolan-More curves

- Let $\{r_1, \dots, r_n\}$ be the set of compared methods, $\{S_1, \dots S_T\}$ be the set of tasks (datasets), q_{ti} be the quality of the method i on the dataset t
- For each method i we introduce $p_i(\beta)$, a fraction of datasets, on which the method i is worse than the best one not more than β times:

$$p_i(\beta) = \frac{1}{T} \left| \left\{ t : q_{ti} \geq \frac{1}{\beta} \max_i q_{ti} \right\} \right|, \; \beta \geq 1$$

- For example, $p_i(1)$ is a fraction of datasets where the method i is the best
- A graph of $p_i(\beta)$ is called Dolan-More curve for the method i
- This definition implies that the higher the curve, the better the method



- Random approximation works better than others