Non-Linear Regression

Václav Šmídl

April 24, 2018

Linear regression and OLS

Fit by a linear function:

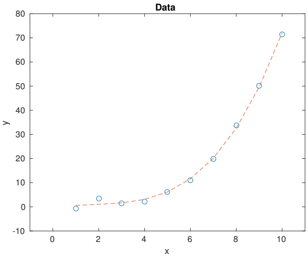
$$
\begin{aligned}
y_1 &= ax_1 &+b1, &\quad +e_1 \\
y_2 &= ax_2 &+b1 &\quad +e_2, \\
\vdots &\quad \vdots &\vdots &\quad \vdots
\end{aligned}
$$

Minimize

$$\sum_i e_i^2 = \sum_i (y_i - ax_i - b)^2$$

Solution:

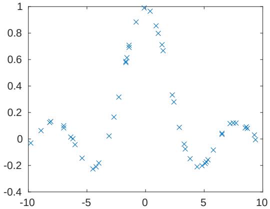
$$
\begin{aligned}
\frac{d(\sum_i e_i^2)}{d\theta} &= 0. \\
\hat{\theta} &= (X^T X)^{-1} X^T \mathbf{y}.
\end{aligned}
$$



Limit of fixed bases

Fit by a linear function:

$$y_i \quad = a\phi_1(x_i) \quad + b\phi_2(x_i) \quad + e_i$$

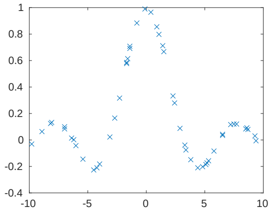


What are the basis function?

Limit of fixed bases

Fit by a linear function:

$$y_i \quad = a\phi_1(x_i) \quad +b\phi_2(x_i) \quad +e_i$$



What are the basis function?
Basis functions are functions of data:

$$y_i \quad = a\phi_1(\psi_1, x_i) \quad +b\phi_2(\psi_2, x_i) \quad +e_i$$

where ϕ_j are non-linear functions.
Estimating new set of parameters
$\theta = [a, b, \psi_1, \psi_2]$

$$\frac{d(y_i - a\phi_1(\psi_1, x_i) + b\phi_2(\psi_2, x_i))}{d\theta} = 0.$$

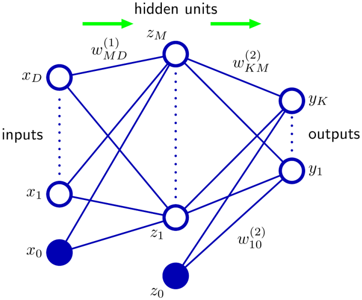
$$\hat{\theta} = ?$$

Neural networks

Feed forward NN:

$$z = \sigma_1 \left(W_1 x + b_1 \right),$$
$$y = \sigma_2 \left(w_m z_m + b_m \right)$$

with vector-valued
– **activation** functions $\sigma_j()$,
– **weights** w_j
– **biases** b_i.



For Gaussian noise, MSE (mean square error) loss function:

$$L = \sum_{i=1}^{n} \left(y_i - \sigma_1 \left(w_1 \sigma_2 \left(\cdots \right) + b_1 \right) \right)^2 .$$

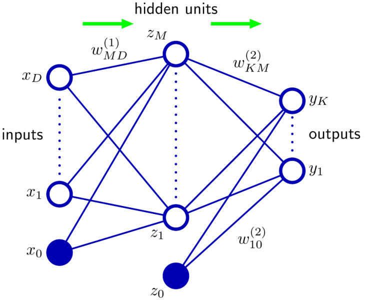
finding $\theta = [w_1, b_1, w_2, b_2, \ldots,]$ by

Neural networks

Feed forward NN:

$$z = \sigma_1 \left(W_1 x + b_1 \right),$$
$$y = \sigma_2 \left(w_m z_m + b_m \right)$$

with vector-valued
– **activation** functions $\sigma_j()$,
– **weights** w_j
– **biases** b_i.



For Gaussian noise, MSE (mean square error) loss function:

$$L = \sum_{i=1}^{n} \left(y_i - \sigma_1 \left(w_1 \sigma_2 \left(\cdots \right) + b_1 \right) \right)^2 .$$

finding $\theta = [w_1, b_1, w_2, b_2, \dots,]$ by gradient descent method

$$\hat{\theta}^{(\tau+1)} = \hat{\theta}^{(\tau)} - \eta \nabla L(\hat{\theta}^{(\tau)}),$$

Regularizationwhere η is the (small) learning rate.

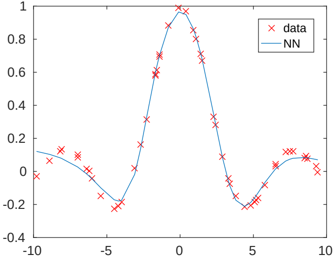
Example

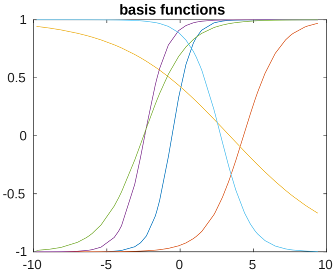
Trivial NN with one hidden layer:

$$y_i = \sum_{i=1}^{6} w_{2,i} \tanh(w_{1,j} x_i + b_{1,j}) + b_2,$$

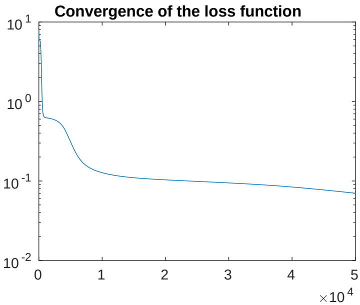
tanh activation function on hidden layer and linear activation function on output.
Training:

1. random initialization,
2. 50000 steps,
3. rate $\eta = 0.001$,





Convergence issues



- Step-size tuning, schedule, annealing, ...
- Better gradient (ADAM) [Kingma, Ba, 2014].
- Higher order methods (half-quadratic approximation, Hessian, LMBFS)

Preventing overfitting

1. ridge regression

$$L = \sum_{i=1}^{m} \left(y_i - \sigma_1\left(w_1\sigma_2\left(\cdots\right) + b_1\right)\right)^2 + \alpha \sum_{k,l} w_{k,l}^2.$$

2. Automatic relevance determination (Laplace)

$$L = \sum_{i=1}^{m} \left(y_i - \sigma_1\left(w_1\sigma_2\left(\cdots\right) + b_1\right)\right)^2 + \sum_{k,l} \alpha_{k,l} w_{k,l}^2.$$

3. Stochastic gradient descent

$$\nabla L(\hat{\theta}) = \sum_{i=1}^{n} \nabla L(\hat{\theta}, x_i, y_i) \approx \nabla L(\hat{\theta}) = \sum_{i=\mathcal{I}} \nabla L(\hat{\theta}, x_i, y_i),$$

where $\mathcal{I}$ is a random subsample of $\{1, \ldots, n\}$.

Points

	points
Ridge regression	10
ADAM (own implementation)	10
Stochastic Gradient Descent	10
Relevance determination	30