Deep Learning - Foundations and Concepts

Chapter 4. Single-layer Networks: Regression

nonlineark@github

February 9, 2025

Outline

Basis functions

Consider the linear combinations of fixed nonlinear functions of the input variables:

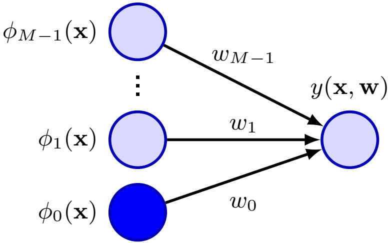
$$y(x;w) = w_0 + \sum_{m=1}^{M-1} w_m \phi_m(x)$$

where $\phi_m(x)$ are known as basis functions. The parameter w_0 allows for any fixed offset in the data and is sometimes called a bias parameter. If we define $\phi_0(x) = 1$ then $y(x;w)$ becomes:

$$y(x;w) = \sum_{m=0}^{M-1} w_m \phi_m(x) = w^T \phi(x)$$

Basis function

Figure: The linear regression model as a single-layer network



Basis function

Here are some possible choices of basis functions:

- Polynomial: $\phi_m(x) = x^m$.
- Gaussian: $\phi_m(x) = \exp(-\frac{(x-\mu_m)^2}{2s^2})$.
- Sigmoidal: $\phi_m(x) = \frac{1}{1+\exp(-\frac{x-\mu_m}{s})}$.

Maximum likelihood

Consider a data set of inputs $\{x^1, \ldots, x^N\}$ with corresponding target values $t_1, \ldots, t_N$. Assume that given the value of x^n, the corresponding value of t_n has a Gaussian distribution. The likelihood function takes the form:

$$p(t_1, \ldots, t_N | x^1, \ldots, x^N; w, \sigma^2) = \prod_{n=1}^{N} \mathcal{N}(t_n; w^T \phi(x^n), \sigma^2 I)$$

The negative log of the likelihood function is given by:

$$\begin{aligned} L &= -\log p(t_1, \ldots, t_N | x^1, \ldots, x^N; w, \sigma^2) \\ &= \frac{N}{2} \log(2\pi) + \frac{N}{2} \log \sigma^2 + \frac{1}{2\sigma^2} \sum_{n=1}^{N} (t_n - w^T \phi(x^n))^2 \end{aligned}$$

Maximum likelihood

Let's maximize the likelihood function (for simplicity, we will denote $\phi(x^n)$ by ϕ_n):

$$\frac{\partial L}{\partial w} = \frac{1}{\sigma^2}(w^T \sum_{n=1}^{N} \phi_n \phi_n^T - \sum_{n=1}^{N} t_n \phi_n^T) = \frac{1}{\sigma^2}(w^T \Phi^T \Phi - t^T \Phi)$$

where:

$$\Phi = \begin{pmatrix} \phi_1^T \\ \phi_2^T \\ \vdots \\ \phi_N^T \end{pmatrix}$$

Maximum likelihood

We see that:

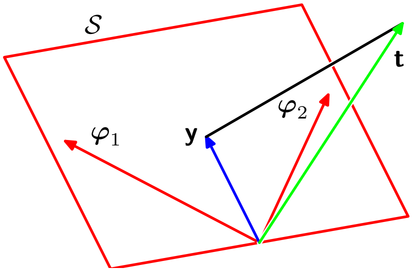
$$w_{ML} = (\Phi^T \Phi)^{-1} \Phi^T t$$

The quantity $(\Phi^T \Phi)^{-1} \Phi^T$ is known as the Moore-Penrose pseudo-inverse of the matrix Φ. It's easy to calculate σ^2_{ML} as well:

$$\sigma^2_{ML} = \frac{1}{N} \sum_{n=1}^{N} (t_n - w^T_{ML} \phi_n)^2$$

Geometry of least squares

Figure: Geometrical interpretation of the least squares solution



Geometry of least squares

Let Φ_m be the mth column of the matrix Φ, and let $y_{ML} \in \mathbb{R}^N$ be the best approximation to t we obtained by maximizing the likelihood function:

$$y_{ML} = \begin{pmatrix} w_{ML}^T \phi_1 \\ w_{ML}^T \phi_2 \\ \vdots \\ w_{ML}^T \phi_N \end{pmatrix} = \Phi w_{ML} = \sum_{m=0}^{M-1} (w_{ML})_m \Phi_m$$

Here we clearly see that $y_{ML} \in \operatorname{span}(\Phi_0, \ldots, \Phi_{M-1})$. In addition, we have:

$$\Phi^T y_{ML} = \Phi^T \Phi w_{ML} = (\Phi^T \Phi)(\Phi^T \Phi)^{-1} \Phi^T t = \Phi^T t$$
$$(t - y_{ML})^T \Phi = 0 \qquad (t - y_{ML})^T \Phi_m = 0$$

That is, $t - y_{ML}$ is orthogonal to each Φ_m, or put another way, y_{ML} is the orthogonal projection of t.

Sequential learning

The maximum likelihood estimator for w involves processing the entire training set in one go. Sometimes we want the data points to be considered one at a time and the model parameters updated after each such presentation. The technique of stochastic (sequential) gradient descent:

- The error function comprises a sum over data points: $E = \sum_n E_n$.
- After presentation of data point n, updates the parameter w using: $w^{(\tau+1)} = w^{(\tau)} - \eta \nabla E_n$.
- τ denotes the iteration number, and η is a training rate parameter.

Sequential learning

For the sum-of-squares error function:

$$E_n = \frac{1}{2}(t_n - w^T\phi_n)^2$$
$$\nabla E_n = -(t_n - w^T\phi_n)\phi_n$$
$$w^{(\tau+1)} = w^{(\tau)} + \eta(t_n - (w^{(\tau)})^T\phi_n)\phi_n$$

Regularized least squares

Adding a regularization term to an error function to control over-fitting:

$$E_D(w) + \lambda E_W(w)$$

For example, if we use the sum-of-squares error function, the total error function becomes:

$$\frac{1}{2}\sum_{n=1}^{N}(t_n - w^T\phi_n)^2 + \frac{\lambda}{2}w^T w$$

Minimizing this total error function, we obtain:

$$w_{ML} = (\lambda I + \Phi^T\Phi)^{-1}\Phi^T t$$

Multiple outputs

We have considered situations with a single target variable. In some applications, we may wish to predict $K > 1$ target variables. Let's first get the dimensions right:

- There are N input data: $x^1, \ldots, x^N$, where $x^n \in \mathbb{R}^D$.
- There are N target data: $t^1, \ldots, t^N$, where $t^n \in \mathbb{R}^K$.
  - Let $T = \begin{pmatrix} t_1 & t_2 & \ldots & t_N \end{pmatrix}^T \in \mathbb{R}^{N \times K}$
- There is a basis ϕ: $\mathbb{R}^D \to \mathbb{R}^M$, $x \to \phi(x)$. For simplicity, we denote $\phi(x^n)$ by ϕ_n.
  - Let $\Phi = \begin{pmatrix} \phi_1 & \phi_2 & \ldots & \phi_N \end{pmatrix}^T \in \mathbb{R}^{N \times M}$
- There is a matrix of parameters: $W \in \mathbb{R}^{M \times K}$.

Multiple outputs

Now, let's maximize the likelihood for $y(x; W) = W^T \phi(x)$:

$$
\begin{aligned}
L &= -\log p(t^1, \ldots, t^N | x^1, \ldots, x^N; W, \sigma^2) \\
&= -\log \prod_{n=1}^{N} \mathcal{N}(t^n; W^T \phi_n, \sigma^2 I) \\
&= \frac{NK}{2} \log(2\pi\sigma^2) + \frac{1}{2\sigma^2} \sum_{n=1}^{N} ||t^n - W^T \phi_n||^2 \\
\frac{\partial L}{\partial W}(W)H &= \frac{1}{\sigma^2} \sum_{n=1}^{N} (\mathrm{tr}(W^T \phi_n \phi_n^T H) - \mathrm{tr}(t^n \phi_n^T H)) \\
&= \frac{1}{\sigma^2} \mathrm{tr}((W^T \Phi^T \Phi - T^T \Phi) H) \\
W_{ML} &= (\Phi^T \Phi)^{-1} \Phi^T T
\end{aligned}
$$