

CS-567 Machine Learning

Nazar Khan

PUCIT

Lectures 20-21
Jan 7, 12 2016

Regression

- ▶ The previous topic, density estimation, was an unsupervised learning problem.
 - ▶ The goal was to model the distribution $p(\mathbf{x})$ of input variables $\mathbf{x}$.
- ▶ We now turn to supervised learning where we model the *predictive distribution* $p(t|\mathbf{x})$.
- ▶ We start by studying the problem of *regression*.
 - ▶ Predict *continuous* target variable(s) t given input variables vector $\mathbf{x}$.
- ▶ Given training data $\{(\mathbf{x}_1, t_1), \dots, (\mathbf{x}_N, t_N)\}$, learn a function $y(\mathbf{x}, \mathbf{w})$ that maps the inputs to the targets.
- ▶ Regression corresponds to finding the optimal parameters $\mathbf{w}^*$.

Linear Regression

- ▶ The simplest regression model is *linear regression*.
- ▶ Linear in parameters $\mathbf{w}$ and linear in inputs $\mathbf{x}$.

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \mathbf{x} = w_0 + w_1 x_1 + \cdots + w_D x_D$$

- ▶ Parameter w_0 accounts for a fixed offset in the data and is called the *bias* parameter.
- ▶ Note that for $\mathbf{x} \in \mathbb{R}^D$, $\mathbf{w} \in \mathbb{R}^{D+1}$.

Linear Regression

- ▶ Linear models are significantly limited for practical problems – especially for high dimensional inputs.
- ▶ However, they have nice analytical properties and they form the foundation for more sophisticated machine learning approaches.

Linear Regression

- ▶ A more powerful model is linear in parameters $\mathbf{w}$ but non-linear in inputs $\mathbf{x}$.

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \phi(\mathbf{x}) = w_0 \phi_0(\mathbf{x}) + w_1 \phi_1(\mathbf{x}) + \cdots + w_M \phi_M(\mathbf{x})$$

- ▶ $\phi_0(\mathbf{x})$ is usually set to 1 to make w_0 the bias parameter.
- ▶ Note that now $\mathbf{w} \in \mathbb{R}^{M+1}$ where M is not necessarily equal to D .
- ▶ The input $\mathbf{x}$ -space is non-linearly mapped to ϕ -space and learning takes place in this new ϕ -space.
- ▶ While the learning remains linear, the learned mapping is actually non-linear in $\mathbf{x}$ -space.

Linear Regression

Probabilistic perspective

- ▶ We have already covered linear regression in our polynomial fitting example.
- ▶ As before, we assume that target t is given by a deterministic function $y(\mathbf{x}, \mathbf{w})$ with additive Gaussian noise. That is

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon$$

where $\epsilon \sim \mathcal{N}(0, \beta^{-1})$.

- ▶ Therefore, we can write

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1})$$

Linear Regression

Probabilistic perspective

- ▶ Likelihood for i.i.d data $\{(\mathbf{x}_1, t_1), \dots, (\mathbf{x}_N, t_N)\}$ can be written as

$$\prod_{n=1}^N \mathcal{N}(t_n | \mathbf{w}^T \phi(\mathbf{x}_n), \beta^{-1})$$

- ▶ Log-likelihood becomes

$$\frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \underbrace{\beta \frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2}_{\text{SSE}}$$

- ▶ Therefore, maximisation of log-likelihood with respect to $\mathbf{w}$ is equivalent to minimisation of SSE function.

Linear Regression

Probabilistic perspective

- ▶ Gradient with respect to $\mathbf{w}$ is $\sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\} \phi(\mathbf{x}_n)^T$.
- ▶ Equating gradient to the $\mathbf{0}$ vector

$$\mathbf{0} = \sum_{n=1}^N t_n \phi(\mathbf{x}_n)^T - \mathbf{w}_{\text{ML}}^T \left(\sum_{n=1}^N \phi(\mathbf{x}_n) \phi(\mathbf{x}_n)^T \right)$$

- ▶ To convert to a pure matrix-vector notation without summations, let us define the following $N \times M$ matrix

$$\Phi = \begin{bmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{bmatrix}$$

known as the *design matrix*.

Linear Regression

Probabilistic perspective

- ▶ It can be verified that the second term in Equation (1) $\sum_{n=1}^N \phi(\mathbf{x}_n)\phi(\mathbf{x}_n)^T = \Phi^T \Phi$. (H.W. Verify this.)
- ▶ By placing the target values in a vector $\mathbf{t} = (t_1, \dots, t_N)^T$ we can also write the first term as $\Phi^T \mathbf{t}$. (H.W. Verify this.)
- ▶ Now we can solve for $\mathbf{w}_{\text{ML}}$ as

$$\mathbf{w}_{\text{ML}} = \underbrace{(\Phi^T \Phi)^{-1} \Phi^T}_{\Phi^\dagger} \mathbf{t} \text{ (this was your answer to Exercise 1.1)}$$

- ▶ The $M \times N$ matrix $\Phi^\dagger$ is known as the *Moore-Penrose pseudo-inverse* or simply *pseudo-inverse* of matrix Φ .
- ▶ It is a generalisation of matrix inverse to non-square matrices.
- ▶ For a square, invertible matrix Φ , it can be verified that $\Phi^\dagger = \Phi^{-1}$. (H.W. Verify this.)

Linear Regression

Regularisation

- ▶ MAP estimation using a zero-mean Gaussian prior on $\mathbf{w}$ leads to regularised linear regression

$$\mathbf{w}_{\text{ML}} = (\lambda \mathbf{I} + \Phi^T \Phi)^{-1} \Phi^T \mathbf{t} \text{ (this was your answer to Exercise 1.2)}$$

where λ is the *regularisation coefficient* that controls the trade-off between fitting and regularisation.

- ▶ This is also known as *regularised least squares*.
- ▶ Such regularisation is also called *weight decay* or *parameter shrinkage* because it encourages weight/parameter values to remain close to 0.
- ▶ Regularisation allows more complex models to be trained on small datasets without severe over-fitting.
- ▶ However, parameter λ needs to be set appropriately.

Linear Regression

Multivariate targets

- ▶ For the case of multivariate target vectors $\mathbf{t}_n \in \mathbb{R}^K$, we are interested in the multivariate mapping $\mathbf{y}(\mathbf{x}, \mathbf{W}) = \mathbf{W}^T \Phi(\mathbf{x})$.
- ▶ Column k of the $M \times K$ matrix $\mathbf{W}$ determines the mapping from $\phi(\mathbf{x})$ to the k_{th} output component.
- ▶ Under isotropic Gaussian noise assumption, we can write the *multivariate* predictive distribution

$$p(\mathbf{t}|\mathbf{x}, \mathbf{W}, \beta) = \mathcal{N}(\mathbf{t}|\mathbf{y}(\mathbf{x}, \mathbf{W}), \beta^{-1}\mathbf{I}) = \mathcal{N}(\mathbf{t}|\mathbf{W}^T \Phi(\mathbf{x}), \beta^{-1}\mathbf{I})$$

- ▶ The ML solution for i.i.d. data $\{\mathbf{x}_n, \mathbf{t}_n\}_{n=1}^N$ can then be computed as

$$\mathbf{W}_{\text{ML}} = \Phi^\dagger \mathbf{T}$$

where $\mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}$ is the $N \times K$ matrix of target vectors.