Chapter 3 Linear Models for Regression

Definition. ***Introduction***

1. ***Problem*** *Given training set* $\{x_n\}$*, where* $n = 1, \cdots, N$ *and corresponding target values* $\{t_n\}$*, the goal is to predict value of* t *for a new value of* $\mathbf{x}$*.*

2. ***Approach*** *Construct a function* $y(\mathbf{x})$ *whose value is the prediction for some* $\mathbf{x}$*. More generally, we aim to model the predictive distribution* $p(t|\mathbf{x})$ *as it represent uncertainty of* t *given* $\mathbf{x}$*. We want to find a* $y(\mathbf{x})$ *in such a way so as to minimize the expected loss of a chosen loss function (i.e. for squared loss,* $y(\mathbf{x}) = \mathbb{E}_t\{t|\mathbf{x}\}$*, the mean of predictive distribution)*

3.1 Linear Basis Function Models

Definition. ***Linear Basis Function***

1. ***Model*** *A linear model with respect to parameters* $w_0, \cdots, w_D$

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi(\mathbf{x}) = \mathbf{w}^{\mathbf{T}} \phi(\mathbf{x}) \qquad \phi_0(\mathbf{x}) = 1$$

where $\phi_j(\mathbf{x})$ *are basis functions and* M *is the total number of parameters in the model, including bias.* $\mathbf{w} = (w_0, \cdots, w_{M-1})^T$ *and* $\phi = (\phi_0, \cdots, \phi_{M-1})^T$*. Here is some basis functions*

$$\phi_j(x) = x^j \text{ (polynomial)} \qquad \phi_j(x) = \exp -\frac{(x-\mu_j)^2}{2s^2} \text{ (gaussian)}$$

$$\phi_j(x) = \sigma(\frac{x-\mu_j}{s}) \quad \sigma(x) = \frac{1}{1+e^{-x}} \text{ (logistic)}$$

Definition. ***Maximum Likelihood and Least Squares*** *Here we show that fitting a linear basis function by minimizing a sum-of-square error function can be motivated as the maximum likelihood solution under an assumed Gaussian noise model. Let target variable determined by* $y(\mathbf{w}, \mathbf{x})$ *with Gaussian noise*

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

We can model the predictive distribution with

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

For squared loss, the optimal prediction for new value of $\mathbf{x}$ *is given by conditional mean of target variable,*

$$\mathbb{E}\{t|\mathbf{x}\} = \int tp(t|\mathbf{x})dt = y(\mathbf{x}, \mathbf{w}) \quad \text{i.e. } \mathbb{E}\{p(t|\mathbf{x}, \mathbf{w}, \beta)\} = y(\mathbf{x}, \mathbf{w})$$

Let $\mathbf{X} = \{\mathbf{x_1}, \cdots, \mathbf{x_N}\}$ *with target* $\mathbf{t} = \{\mathbf{t_1}, \cdots, \mathbf{t_N}\}$*, we get likelihood fucntion*

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}\left(t_n|\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x_n}), \beta^{-1}\right)$$

Now we try to minimize likelihood

$$\ln p(\mathbf{t}|\mathbf{w}, \beta) = \sum_n \ln \mathcal{N}\left(t_n|\mathbf{w}^T\phi(\mathbf{x_n}), \beta^{-1}\right) = \frac{N}{2}\ln\beta - \frac{N}{2}\ln 2\pi - \beta E_D(\mathbf{w})$$

where sum of squared error function is defined by

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(t_n - \mathbf{w^T}\boldsymbol{\phi}(\mathbf{x_n}))^2$$

Now we find maximum likelihood estimator for $\mathbf{w}$ *and* β*,*

$$\frac{\partial}{\partial \mathbf{w}} = \sum_{n=1}^{N}(t_n - \mathbf{w^T}\boldsymbol{\phi}(\mathbf{x}_n))\boldsymbol{\phi}(\mathbf{x}_n)^T \qquad \rightarrow \qquad \sum_{n=1}^{N} t_n\boldsymbol{\phi}(\mathbf{x_n})^{\mathbf{T}} = \mathbf{w}^T\left(\sum_{n=1}^{N}\boldsymbol{\phi}(\mathbf{x}_n)\boldsymbol{\phi}(\mathbf{x}_n)^T\right)$$

$$\mathbf{w}_{mle} = (\mathbf{\Phi^T\Phi})^{-1}\mathbf{\Phi^T t}$$

where $\mathbf{\Phi}$ *is called the design matrix*

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

3.13 Sequential Learning

Definition. ***Sequential/On-line Learning*** *Instead of processing the entire training set in one go, sequential algorithms consider each data points one at a time.* ***Stochastic gradient descent*** *is an online algorithm. Given error function as a sum over errors of all training samples, i.e.* $E = \sum_n E_n$*,* w *is updated according to*

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta\nabla E_n$$

where τ *is iteration number and* η *is a learning rate parameter. For sum-of-squares error function, this gives*

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta(t_n - \mathbf{w}^{(\tau)T}\boldsymbol{\phi}_n)\boldsymbol{\phi}_n \qquad \boldsymbol{\phi}_n = \boldsymbol{\phi}(\mathbf{x}_n)$$

Definition. ***Regularized Least Squares*** *Adding regularization term to an error function to control over-fitting,*

$$E_D(\mathbf{w}) + \lambda E_W(\mathbf{w})$$

In case of sum-of-squares error with sum-of-squares regularizer

$$\frac{1}{2}\sum_{n=1}^{N}(t_n - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n))^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

The particular choice is called ***weight decay****, an example of parameter shrinkage method because it shrinks parameter to 0. In this case, maximum likelihood estimator has closed form*

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

A more general regularizer has form

$$\frac{1}{2}\sum_{n=1}^{N}(t_n - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n))^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|^q$$

3.2 The Bias-Variance Decomposition

Definition. ***Bias-Variance Decomposition*** *Decision theory for regression consists of choosing a particular estimate $y(\mathbf{x})$ of value of t for each input $\mathbf{x}$ such that the expected loss is minimized. For squared loss function, the optimal prediction $y(\mathbf{x})$ is given by*

$$h(\mathbf{x}) = \mathbb{E}\{t|\mathbf{x}\} = \int tp(t|\mathbf{x})dt$$

We can rewrite the expected loss as

$$\mathbb{E}\{L\} = \int (y(\mathbf{x}) - h(\mathbf{x}))d\mathbf{x} + \int (h(\mathbf{x}) - t)^2 p(\mathbf{x})d\mathbf{x}$$

where the first term is minimized if $y(\mathbf{x}) = \mathbb{E}\{t|\mathbf{x}\}$ and the second term, independent of $y(\mathbf{x})$ is the variance of the distribution of t averaged over $\mathbf{x}$, representing the intrinsic variability of the data. We can decompose expectedf loss in terms of bias, variance, and noise

$$\mathbb{E}\{L\} = \int (\mathbb{E}_{\mathcal{D}}\{y(\mathbf{x};\mathcal{D})\} - h(\mathbf{x}))^2 p(\mathbf{x})d\mathbf{x} + \int \mathbb{E}_{\mathcal{D}}\left\{y(\mathbf{x}\mathcal{D}) - \mathbb{E}_{\mathcal{D}}\{y(\mathbf{x};\mathcal{D}])\}^2\right\} p(\mathbf{x})d\mathbf{x} + \int (h(\mathbf{x}) - t)^2 p(\mathbf{x},t)d\mathbf{x}dt$$

given a particular dataset $\mathcal{D}$. The first term is ***squared bias****, represents extend to which average prediction over all data sets differs from the desired regression function. The second term, called* ***variance****, measures the extend to which solutions for individual datasets vary around their average (average of $y(\mathbf{x};\mathcal{D})$s), hence measures the extent to which $y(\mathbf{x};\mathcal{D})$ is sensitive to a particular choice of dataset.*

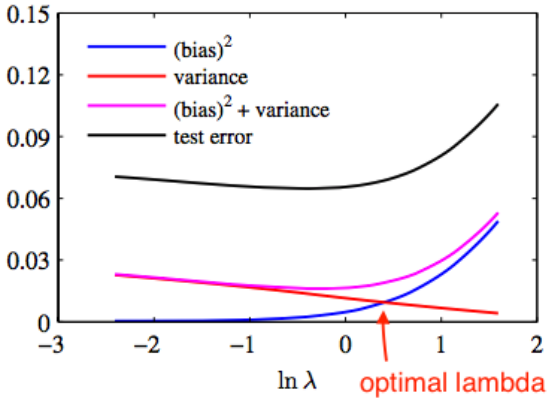
As an example, assuming we sample L datasets and fit a model by minimizing regularized squared error function to give L prediction function $y^{(l)}(x)$. The average prediction is estimated by

$$\bar{y}(x) = \frac{1}{L}\sum_{l=1}^{L} y^{(l)}(x)$$

and the integrated squared bias and integrated variance is approximated by finite sums

$$\begin{aligned} (bias)^2 &= \frac{1}{N}\sum_{n=1}^{N}(\bar{y}(x_n) - h(x_n))^2 \\ variance &= \frac{1}{N}\sum_{n=1}^{N}\frac{1}{L}\sum_{l=1}^{L}(y^{(l)}(x_n) - \bar{y}(x_n))^2 \end{aligned}$$

Note values of these terms depend on choice of regularization parameter, changes to λ adjusts for the bias-variance tradeoff.



Definition. ***Convex Function*** *A function f is convex if and only if for all θ_1, β_2, and for all $\alpha \in [0,1]$, we have*

$$f(\alpha\theta_1 + (1-\alpha)\theta_2) \leq \alpha f(\theta_1) + (1-\alpha)f(\theta_2)$$

Definition. ***Newton's Method*** *is a method for finding successively better approximations to the roots of a real-valued function. The algorithm starts with a function f, the derivative f' and the initial guess x_0 for the root of f, a better approximation is given by updating x_n until convergence*

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

$(x_n, 0)$ is the intersection of x-axis and the tangent of the graph of f at $(x_n, f(x_n))$. Note the equation of tangen line is given by

$$y = f'(x_n)(x - x_n) + f(x_n) \qquad \rightarrow \qquad 0 = f'(x_n)(x_{n+1} - x_n) + f(x_n)$$

We can use Newton's Method to find a miminum or maximum of a function f by applying Newton's method to the derivative

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}$$

In the multivariate case, we have

$$y = f(\mathbf{x} + \triangle\mathbf{x}) \approx f(\mathbf{x}) + \nabla f(\mathbf{x})^T \triangle\mathbf{x} + \frac{1}{2}\triangle\mathbf{x}^T \mathbf{H}(\mathbf{x})\triangle\mathbf{x}$$

Take the gradient

$$\nabla_{\triangle\mathbf{x}} f(\mathbf{x} + \triangle\mathbf{x}) \approx \nabla f(\mathbf{x}) + \mathbf{H}\triangle\mathbf{x}$$

Setting gradient to zero provides the Newton step

$$\triangle\mathbf{x} = -H^{-1} y(\mathbf{x}) \qquad \rightarrow \qquad \mathbf{x}_{n+1} = \mathbf{x}_n + \triangle\mathbf{x}$$

Computing and storing hessian matrix takes $\Theta(n^2)$ memory, infeasible for high dimensional functions.

1. ***Quasi-Newton Method*** *Replaces the exact Hessian with an approximation.*

Definition. ***Stochastic Gradient Descent*** *is a stochastic approximation of the greadient descent optimization and iterative method for minimizing an objective function that is written as a sum of differentiable functions. Sum-minimization problems arise in least squares and in maximum likelihood estimation*

$$Q(w) = \frac{1}{n}\sum_{i=1}^{N} Q_i(w)$$

where we want to estimate w such that $Q(w)$ is minimized. A batch gradient descent updates weight with

$$w = w - \eta\nabla Q(w) = w - \frac{\eta}{N}\sum_{i=1}^{N} \nabla Q_i(w)$$

Ther idea is that the summand functions have a simple form that enables inexpensive evalaution of sum and gradient operations for objective function with a good form. In stochastic gradient descent, the gradient $Q(w)$ is approximated by a gradient at a single example

$$w = w - \eta\nabla Q_i(w)$$

which perfoms update for each training example, in multiple epochs, such that the algorithm converges. To choose an appropriate setp size by picking a sequence of η_t such that

$$\sum_t \eta_t = \infty \qquad\qquad \sum_t \eta_t < \infty$$

satisfied by $\eta_T \propto 1/t$