CS559 Machine Learning
Linear Models

Tian Han

Department of Computer Science
Stevens Institute of Technology

Week 4

Outline

- Linear Regression
- Bayesian Linear Regression
- Generalized Linear Model

Linear Regression

Basic Settings

Given:

- A training dataset comprising N observations $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, each observation is D-dimensional.
- Corresponding target values $\{t_1, t_2, \ldots, t_N\}$.

Basic Settings

Given:

- A training dataset comprising N observations $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, each observation is D-dimensional.
- Corresponding target values $\{t_1, t_2, \ldots, t_N\}$.

Goal:

- Learn the function that model the relation between $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ and $\{t_1, t_2, \ldots, t_N\}$. From probabilistic perspective, model the predictive distribution $p(t|\mathbf{x})$.

Basic Settings

Given:

- A training dataset comprising N observations $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, each observation is D-dimensional.
- Corresponding target values $\{t_1, t_2, \ldots, t_N\}$.

Goal:

- Learn the function that model the relation between $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ and $\{t_1, t_2, \ldots, t_N\}$. From probabilistic perspective, model the predictive distribution $p(t|\mathbf{x})$.
- Predict value of t for a new value of $\mathbf{x}$.

Notations 1

	input	target
1	$x_{11}, x_{12}, ..., x_{1D}$	t_1
2	$x_{21}, x_{22}, ..., x_{2D}$	t_2
...		
N	$x_{N1}, x_{N2}, ..., x_{ND}$	t_N

The model has the following form for i-th observation:

$$\begin{aligned} t_i &= w_0 + x_{i1}w_1 + x_{i2}w_2 + \cdots + x_{iD}w_D + \epsilon_i \\ &= \sum_{j=0}^{D} x_{ij}w_j + \epsilon_i \end{aligned}$$

- $\epsilon_i \sim \mathrm{N}(0, \beta^{-1})$ independently.
- w_0: intercept (or bias) term (assume $x_{i0} = 1$ for all i in above model).
- t_i: response variable, dependent variable.
  x_{ij}: predictor, independent variable.

Notations 2

input	target
$\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_D)$	$\mathbf{t}$

$\mathbf{X}_j = (x_{1j}, \ldots, x_{Nj})^T$, $\mathbf{t} = (t_1, \ldots, t_N)^T$, the model has the following form:

$$\mathbf{t} = \sum_{j=0}^{D} \mathbf{X}_j w_j + \epsilon$$

- $\epsilon = (\epsilon_1, \ldots, \epsilon_n)^T \sim \mathrm{N}(0, \beta^{-1})$
- $\mathbf{X}_0$: all 1's.

Notations 3

	input	target
1	$\mathbf{x}_1^T$	t_1
2	$\mathbf{x}_2^T$	t_2
...		
N	$\mathbf{x}_N^T$	t_N

$\mathbf{x}_i = (x_{i0}, x_{i1}, \ldots, x_{iD})^T$, the model has the following form:

$$t_i = \mathbf{x}_i^T \mathbf{w} + \epsilon_i = \mathbf{w}^T \mathbf{x}_i + \epsilon_i$$

- $\mathbf{w} = (w_0, w_1, \ldots, w_D)^T$

Purpose

The process of estimating w is called learning from the training data. The purpose is two-fold.

- Explanation: understanding the relationship between t_i and $(x_{i1}, x_{i2}, \ldots, x_{iD})$.
- Prediction: learn to predict t_i based on $\mathbf{x}_i^T = (x_{i1}, x_{i2}, \ldots, x_{iD})$, so that in the testing stage, if we are given the new predictor variables $\mathbf{x}^T$, we should be able to predict the outcome t.

	input	output
1	$\mathbf{x}_1^T$	t_1
2	$\mathbf{x}_2^T$	t_2
...		
n	$\mathbf{x}_n^T$	t_n

Linear Regression

Based on previous discussion, we assume target variable t is given by a deterministic function $\mathbf{w}^T\mathbf{x}$ with additive Gaussian noise, i.e.,

$$t = \mathbf{w}^T\mathbf{x} + \epsilon$$

where $\epsilon \sim \mathrm{N}(0, \beta^{-1})$. $\mathbf{x} = (1, x_1, \ldots, x_D)^T$ and $\mathbf{w} = (w_0, w_1, \ldots, w_D)^T$. Then the likelihood of getting target is:

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

- Gaussian noise assumption implies that the conditional distribution of t given $\mathbf{x}$ is *unimodal*.
- Extension to mixture models which permit *multimodal* conditional distributions. (Later Chapter)

Maximum Likelihood

Suppose training data $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ with corresponding target values $\mathbf{t} = \{t_1, \ldots, t_N\}$. Assume target values are *i.i.d* samples from $p(t|\mathbf{x}, \mathbf{w}, \beta)$, then we have the following likelihood:

Maximum Likelihood

Suppose training data $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ with corresponding target values $\mathbf{t} = \{t_1, \ldots, t_N\}$. Assume target values are *i.i.d* samples from $p(t|\mathbf{x}, \mathbf{w}, \beta)$, then we have the following likelihood:

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

Maximum Likelihood

Suppose training data $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ with corresponding target values $\mathbf{t} = \{t_1, \ldots, t_N\}$. Assume target values are *i.i.d* samples from $p(t|\mathbf{x}, \mathbf{w}, \beta)$, then we have the following likelihood:

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

- Typically in supervised learning (such as regression or classification), we are not seeking to model the distribution of the input variable $\mathbf{x}$.
- $\mathbf{x}$ will always appear in the set of conditioning variables, therefore drop it to make notation simpler, i.e., write $p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta)$ as $p(\mathbf{t}|\mathbf{w}, \beta)$.

Maximum Likelihood and Least Square

$$\begin{aligned}\ln p(\mathbf{t}|\mathbf{w},\beta) &= \sum_{n=1}^{N} \ln \mathrm{N}(t_n|\mathbf{w}^T\mathbf{x}_n,\beta^{-1}) \\ &= \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi) - \beta E_D(\mathbf{w})\end{aligned}$$

Maximum Likelihood and Least Square

$$\begin{aligned}\ln p(\mathbf{t}|\mathbf{w},\beta) &= \sum_{n=1}^{N} \ln \mathrm{N}(t_n|\mathbf{w}^T\mathbf{x}_n,\beta^{-1}) \\ &= \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi) - \beta E_D(\mathbf{w})\end{aligned}$$

$E_D(\mathbf{w})$, *sum-of-squares error* function is given by:

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

Maximum Likelihood and Least Square

$$\begin{aligned}\ln p(\mathbf{t}|\mathbf{w},\beta) &= \sum_{n=1}^{N} \ln \mathrm{N}(t_n|\mathbf{w}^T\mathbf{x}_n,\beta^{-1}) \\ &= \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi) - \beta E_D(\mathbf{w})\end{aligned}$$

$E_D(\mathbf{w})$, *sum-of-squares error* function is given by:

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

For linear model parameter $\mathbf{w}$:
Maximization of likelihood function under a conditional Gaussian noise distribution $p(\mathbf{t}|\mathbf{w},\beta)$ is equivalent to **minimization** a sum-of-square error function $E_D(\mathbf{w})$.

MLE for $\mathbf{w}$

For $\mathbf{w}$, take gradients of log-likelihood function:

$$\nabla \ln p(\mathbf{t}|\mathbf{w}, \beta) = \beta \sum_{n=1}^{N} (t_n - \mathbf{w}^T \mathbf{x}_n) \mathbf{x}_n^T$$

MLE for $\mathbf{w}$

For $\mathbf{w}$, take gradients of log-likelihood function:

$$\nabla \ln p(\mathbf{t}|\mathbf{w}, \beta) = \beta \sum_{n=1}^{N} (t_n - \mathbf{w}^T \mathbf{x}_n) \mathbf{x}_n^T$$

Setting this gradient to 0, we get:

$$\begin{aligned} 0 &= \sum_{n=1}^{N} t_n \mathbf{x}_n^T - \mathbf{w}^T (\sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^T) \\ \hat{\mathbf{w}} &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{t} \end{aligned}$$

MLE for $\mathbf{w}$

For $\mathbf{w}$, take gradients of log-likelihood function:

$$\nabla \ln p(\mathbf{t}|\mathbf{w}, \beta) = \beta \sum_{n=1}^{N} (t_n - \mathbf{w}^T \mathbf{x}_n) \mathbf{x}_n^T$$

Setting this gradient to 0, we get:

$$\begin{aligned} 0 &= \sum_{n=1}^{N} t_n \mathbf{x}_n^T - \mathbf{w}^T (\sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^T) \\ \hat{\mathbf{w}} &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{t} \end{aligned}$$

$\mathbf{X}$ is $N \times (D+1)$ matrix, called the *design matrix*, and is given by:

$$\mathbf{X} = \begin{bmatrix} 1 & x_{11} & \dots & x_{1D} \\ 1 & x_{21} & \dots & x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & \dots & x_{ND} \end{bmatrix} = \begin{bmatrix} — & \mathbf{x}_1^T & — \\ & \vdots & \\ — & \mathbf{x}_N^T & — \end{bmatrix}$$

MLE for β

Similarly, we could get MLE estimation for β that is given by:

$$\frac{1}{\hat{\beta}} = \frac{1}{N}\sum_{n=1}^{N}(t_n - \hat{\mathbf{w}}^T\mathbf{x}_n)^2$$

- $\hat{\mathbf{w}}$ is the MLE for linear model parameter $\mathbf{w}$ defined in the previous slide.
- Inverse of the noise precision, i.e., $\frac{1}{\beta}$, is given by residual variance of the target values around the regression function.

A Step Further: Linear Basis Function Models

We discussed the simplest linear model for regression which involves the linear combination of input variables:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D = \mathbf{w}^T \mathbf{x}$$

A Step Further: Linear Basis Function Models

We discussed the simplest linear model for regression which involves the linear combination of input variables:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D = \mathbf{w}^T \mathbf{x}$$

- Such model is a *linear* function of parameters $w_0, \ldots, w_D$.
- Such model is also a *linear* function of input variables x_i (which imposes significant limitations).

A Step Further: Linear Basis Function Models

We discussed the simplest linear model for regression which involves the linear combination of input variables:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D = \mathbf{w}^T \mathbf{x}$$

- Such model is a *linear* function of parameters $w_0, \ldots, w_D$.
- Such model is also a *linear* function of input variables x_i (which imposes significant limitations).
- Extend the model: **linear** combinations of fixed **nonlinear** functions of the input variables.

Linear Basis Function Models

Considers:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1\phi_1(\mathbf{x}) + w_2\phi_2(\mathbf{x}) + \cdots + w_{M-1}\phi_{M-1}(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x})$$

Linear Basis Function Models

Considers:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1\phi_1(\mathbf{x}) + w_2\phi_2(\mathbf{x}) + \cdots + w_{M-1}\phi_{M-1}(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x})$$

- $\mathbf{w} = (w_0, \ldots, w_{M-1})^T$, and $\phi = (1, \phi_1, \ldots, \phi_{M-1})^T$
- $\phi(\mathbf{x})$ usually nonlinear functions of input variables x_i, are known as *basis functions*.
- Linear model: $y(\mathbf{x}, \mathbf{w})$ is *linear* in $\mathbf{w}$.

Linear Basis Function Models

Considers:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1\phi_1(\mathbf{x}) + w_2\phi_2(\mathbf{x}) + \cdots + w_{M-1}\phi_{M-1}(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x})$$

- $\mathbf{w} = (w_0, \ldots, w_{M-1})^T$, and $\phi = (1, \phi_1, \ldots, \phi_{M-1})^T$
- $\phi(\mathbf{x})$ usually nonlinear functions of input variables x_i, are known as *basis functions*.
- Linear model: $y(\mathbf{x}, \mathbf{w})$ is *linear* in $\mathbf{w}$.
- If original data comprises the vector $\mathbf{x}$, then features can be expressed in terms of basis functions $\{\phi_j(\mathbf{x})\}$.
- Many possible choices for basis functions: power form ($\phi_j(x) = x^j$), gaussian, sigmoidal, and even Fourier basis.

Linear Model

Again, assume target value is given by linear model $y(\mathbf{x}, \mathbf{w})$ with additive Gaussian noise:

$$\begin{aligned} t &= y(\mathbf{x}, \mathbf{w}) + \epsilon \\ &= \mathbf{w}^T \phi(\mathbf{x}) + \epsilon \end{aligned}$$

Linear Model

Again, assume target value is given by linear model $y(\mathbf{x}, \mathbf{w})$ with additive Gaussian noise:

$$\begin{aligned} t &= y(\mathbf{x}, \mathbf{w}) + \epsilon \\ &= \mathbf{w}^T \phi(\mathbf{x}) + \epsilon \end{aligned}$$

Probabilistically:

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

Linear Model

Again, assume target value is given by linear model $y(\mathbf{x}, \mathbf{w})$ with additive Gaussian noise:

$$\begin{aligned} t &= y(\mathbf{x}, \mathbf{w}) + \epsilon \\ &= \mathbf{w}^T \phi(\mathbf{x}) + \epsilon \end{aligned}$$

Probabilistically:

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

Assume target values are *i.i.d* samples, the whole likelihood function is:

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

Maximum Likelihood and Least Square

$$\begin{aligned} \ln p(\mathbf{t}|\mathbf{w}, \beta) &= \sum_{n=1}^{N} \ln \mathrm{N}(t_n|\mathbf{w}^T\phi(\mathbf{x}_n), \beta^{-1}) \\ &= \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi) - \beta E_D(\mathbf{w}) \end{aligned}$$

where $E_D(\mathbf{w})$ is *sum-of-square error function* and is given by:

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

MLE for $\mathbf{w}$

Set the gradients of the log-likelihood function to be 0, we have (similar as previous example):

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

- *Normal equations* for least squares problem
- $\mathbf{\Phi}$ is *design matrix* of size $N \times M$:

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

- $\mathbf{\Phi}^\dagger \equiv (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T$ is known as *Moore-Penrose pseudo-inverse* of matrix $\mathbf{\Phi}$ (generalization of matrix inverse).

MLE for β

Similar as the previous example, the MLE for noise precision β is given by:

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

Inverse of the noise precision is estimated by residual variance of the target values around the regression function.

Geometric Interpretation

MLE for $\mathbf{w}$ is equivalent to considering least square:

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

Take gradients, we have:

$$\beta\sum_{n=1}^{N}(t_n - \mathbf{w}^T\phi(\mathbf{x}_n))\phi(\mathbf{x}_n)^T = 0$$

Consider j-th basis function $\phi_j(\mathbf{x})$ over N observation φ_j, i.e.,j-th column of design matrix $\mathbf{\Phi}$, then we have:

$$(\mathbf{t} - \mathbf{\Phi w})\varphi_j = 0$$

- $\mathbf{t} = (t_1, \ldots, t_N)^T$ and $\mathbf{\Phi} = (\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_N))^T$
- Residual $\mathbf{t} - \mathbf{\Phi w}$ *orthogonal* to basis functions which viewed as vector φ_j of length N with elements $\phi_j(\mathbf{x}_n)$

Geometric interpretation

The least square solution for $\mathbf{w}$ corresponds to choice of $\mathbf{y}$ that lies in subspace $\mathcal{S}$ and is closest to $\mathbf{t}$.

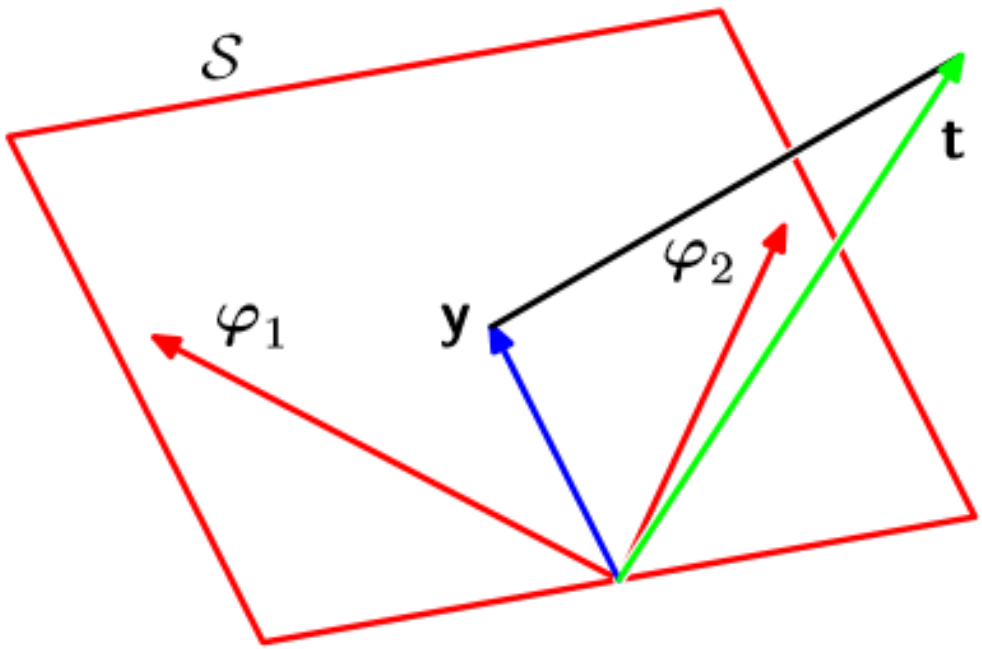

Figure: Geometric interpretation of the least-squares solutions. φ_j is the j-th column of design matrix $\mathbf{\Phi}$ represents the j-th basis function on N observations. [C. Bishop 2006]

Prediction

After learning parameters, we use the learned model $\hat{\mathbf{w}}$ to do prediction. Given new observation $\phi(\mathbf{x}_{new})$, the prediction from the linear model would be:

$$t_{new} = y(\mathbf{x}_{new}, \hat{\mathbf{w}}) = \phi(\mathbf{x}_{new})^T \hat{\mathbf{w}}$$

Regularized Least Squares

To control over-fitting, we often consider **regularized least squares** that adds a regularization term to error function:

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

Regularized Least Squares

To control over-fitting, we often consider **regularized least squares** that adds a regularization term to error function:

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

Again consider sum-of-square error function:

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

Regularized Least Squares

To control over-fitting, we often consider **regularized least squares** that adds a regularization term to error function:

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

Again consider sum-of-square error function:

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

Simple quadratic regularizer (l_2 regularizer):

$$E_W(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

MLE for l_2 regularized least squares

The total error function:

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

Take gradient w.r.t $\mathbf{w}$ and set it to be 0, we obtain:

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

In case $\mathbf{\Phi}^T\mathbf{\Phi} = \mathbf{I}$, then $\mathbf{w}_{rls} = \frac{\mathbf{w}_{ls}}{1+\lambda}$, with $\mathbf{w}_{ls}$ to be the solution for least square problem (also the solution for MLE), and $\mathbf{w}_{rls}$ is a *shrinkage estimator*.

Different Regularizers

$$E_W(\mathbf{w}) = \frac{1}{2}\sum_{j=1}^{M} |w_j|^q$$

- If $q = 2$: quadratic regularizer (l_2 regularizer), encourage the weight value to decay towards 0.
- If $q = 1$: lasso regularizer (l_1 regularizer), encourage *sparse* model in which large proportion of basis functions play no role.

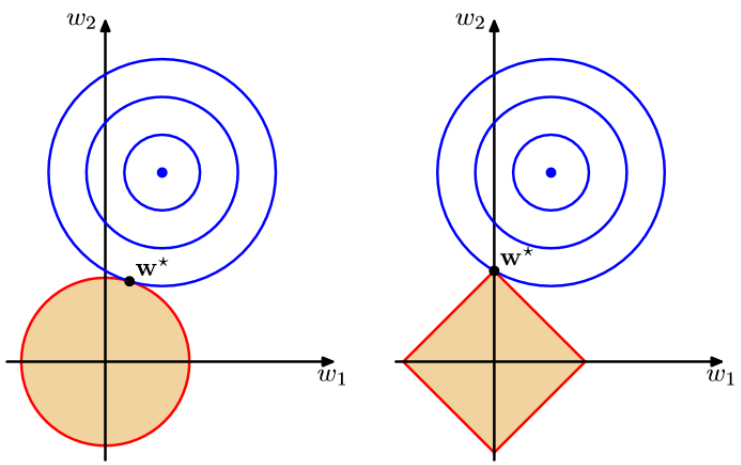

Figure: Left: l_2 regularizer. Right: l_1 regularizer. [C. Bishop 2006]

Sequential Learning: Stochastic Gradient Descent

In practice, the dataset is sufficiently large, it maybe worthwhile to use *sequential* algorithms. The data points are considered one (or small batch) at a time.
Suppose the error function comprises the sum over data points $E = \sum_n E_n$, then after presentation of pattern n, we could use Robbins-Monro like procedure to update the parameter $\mathbf{w}$:

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

- τ is the iteration number, η is the learning rate.
- The $\mathbf{w}$ is initialized to some starting $\mathbf{w}^{(0)}$.

Bayesian Linear Regression

Bayesian Treatment

- The model decided by simply maximizing likelihood function would always lead to complex model and over-fitting.
- Model complexity can be controlled by regularization.
- Bayesian treatment will avoid the over-fitting of MLE and will automatically determine the model complexity.

Likelihood and Prior

Consider $t = \mathbf{w}^T \phi(\mathbf{x}) + \epsilon$, and $\epsilon \sim \mathrm{N}(0, \beta^{-1})$. Assume noise precision β is *known* constant. Then likelihood function is defined as:

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

Likelihood and Prior

Consider $t = \mathbf{w}^T\phi(\mathbf{x}) + \epsilon$, and $\epsilon \sim \mathrm{N}(0, \beta^{-1})$. Assume noise precision β is *known* constant. Then likelihood function is defined as:

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

Define the prior distribution on $\mathbf{w}$:

$$p(\mathbf{w}) = \mathrm{N}(\mathbf{w}|\mathbf{m}_0, \mathbf{S}_0)$$

Likelihood and Prior

Consider $t = \mathbf{w}^T\phi(\mathbf{x}) + \epsilon$, and $\epsilon \sim \mathrm{N}(0, \beta^{-1})$. Assume noise precision β is *known* constant. Then likelihood function is defined as:

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

Define the prior distribution on $\mathbf{w}$:

$$p(\mathbf{w}) = \mathrm{N}(\mathbf{w}|\mathbf{m}_0, \mathbf{S}_0)$$

Consider the simplified Gaussian prior:

$$p(\mathbf{w}|\alpha) = \mathrm{N}(\mathbf{w}|0, \alpha^{-1}\mathbf{I})$$

Posterior

Posterior distribution of $\mathbf{w}$ is proportional to the product of the likelihood function and the prior and is given by:

$$p(\mathbf{w}|\mathbf{t}) = \mathrm{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N)$$

where

$$\begin{aligned} \mathbf{m}_N &= \beta \mathbf{S}_N \mathbf{\Phi}^T \mathbf{t} \\ \mathbf{S}_N^{-1} &= \alpha \mathbf{I} + \beta \mathbf{\Phi}^T \mathbf{\Phi} \end{aligned}$$

Equivalent to Regularized Least Square

Given posterior $p(\mathbf{w}|\mathbf{t})$, **maximization** of this posterior w.r.t $\mathbf{w}$ is equivalent to **minimization** of sum-of-square error function with additional regularization term:

Equivalent to Regularized Least Square

Given posterior $p(\mathbf{w}|\mathbf{t})$, **maximization** of this posterior w.r.t $\mathbf{w}$ is equivalent to **minimization** of sum-of-square error function with additional regularization term:

$$\ln p(\mathbf{w}|\mathbf{t}) = -\frac{\beta}{2}\sum_{n=1}^{N}(t_n - \mathbf{w}^T\phi(\mathbf{x}_n))^2 - \frac{\alpha}{2}\mathbf{w}^T\mathbf{w} + const$$

Equivalent to Regularized Least Square

Given posterior $p(\mathbf{w}|\mathbf{t})$, **maximization** of this posterior w.r.t $\mathbf{w}$ is equivalent to **minimization** of sum-of-square error function with additional regularization term:

$$\ln p(\mathbf{w}|\mathbf{t}) = -\frac{\beta}{2}\sum_{n=1}^{N}(t_n - \mathbf{w}^T\phi(\mathbf{x}_n))^2 - \frac{\alpha}{2}\mathbf{w}^T\mathbf{w} + const$$

- Adding quadratic regularizer to least square corresponds to defining zero-mean isotropic Gaussian prior on parameter $\mathbf{w}$.
- Since the posterior distribution is Gaussian, so maximum posterior parameter coincides with mean of posterior, i.e., $\mathbf{m}_N$

Simple Example

Consider $y(x, \mathbf{w}) = w_0 + w_1 x$. The target values t_n are simulated using following procedure:

- The underlying true model is assumed to be $f(x, \mathbf{a}) = a_0 + a_1 x$, where $a_0 = -0.3$ and $a_1 = 0.5$.
- Choose values x_n from uniform distribution $U(x| -1, 1)$, then evaluating $f(x_n, \mathbf{a})$.
- Add Gaussian noise $\mathrm{N}(0, 0.2)$ to $f(x_n, \mathbf{a})$ to get target values t_n.
- Goal: recover the a_0 and a_1 from the generated targets.
- Sequential update of posterior distribution. Recall $p(\mathbf{w}|\mathcal{D}^N) \propto p(\mathbf{w}|\mathcal{D}^{N-1})p(\mathbf{x}_N|\mathbf{w})$

Initial Stage

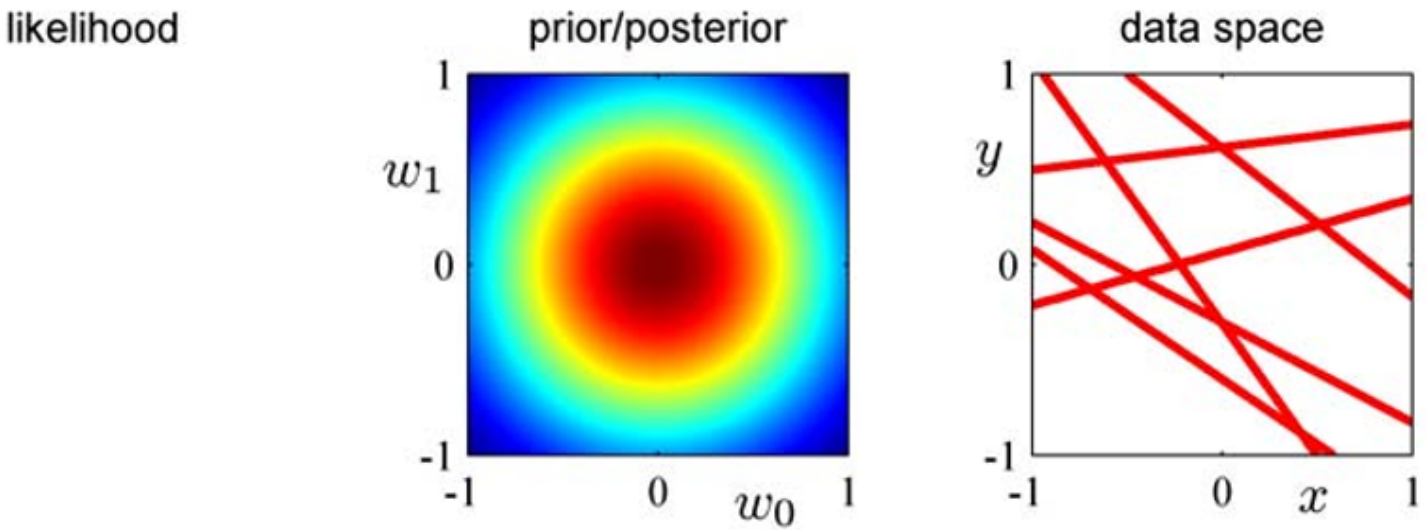

Figure: No data seen yet. $\mathbf{w}$ draw from Gaussian prior. [C. Bishop 2006]

Observe one point

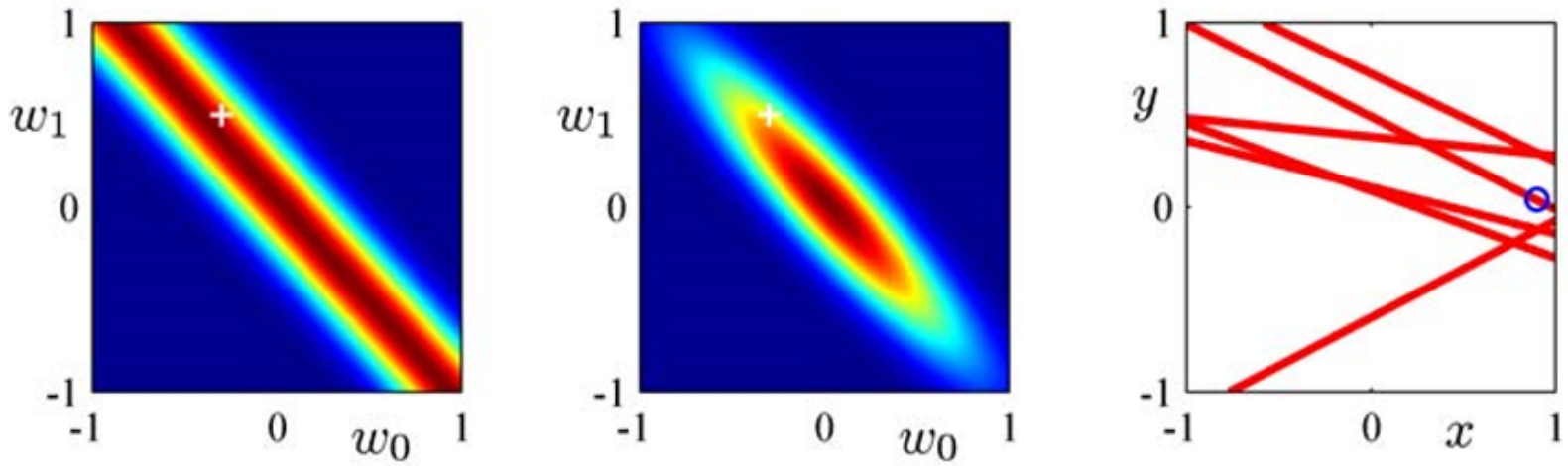

Figure: Observe one point (blue circle). The true parameters $(-0.3, 0.5)$ are marked as white cross. $\mathbf{w}$ draw from posterior. [C. Bishop 2006]

Observe two points

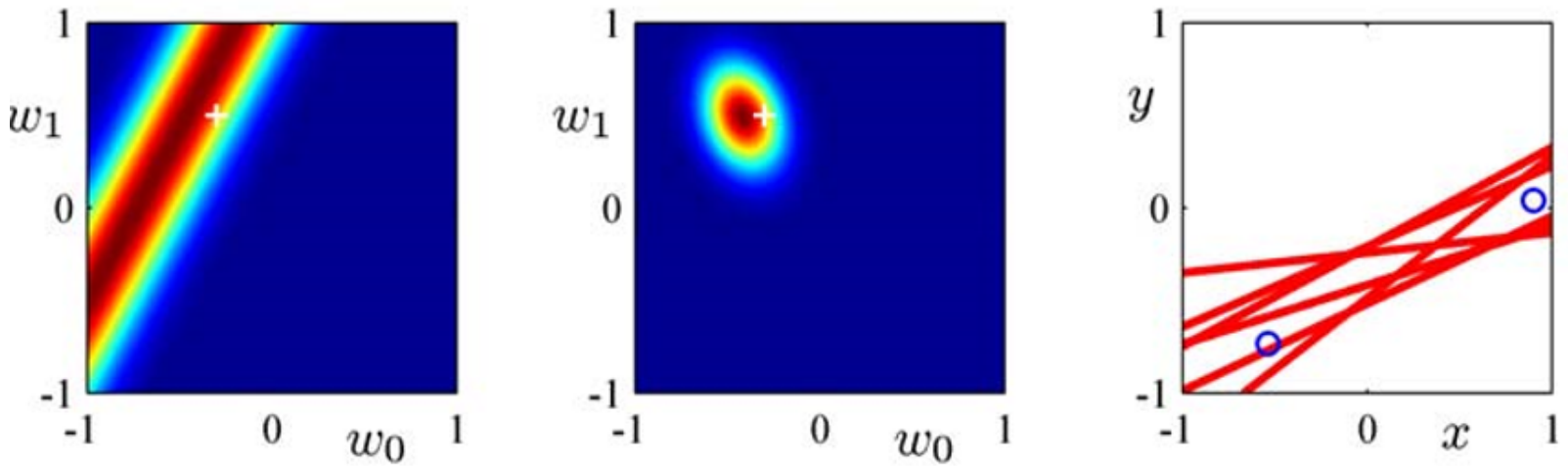

Figure: Observe two points (blue circle). The true parameters $(-0.3, 0.5)$ are marked as white cross. $\mathbf{w}$ draw from posterior. [C. Bishop 2006]

Observe 20 points

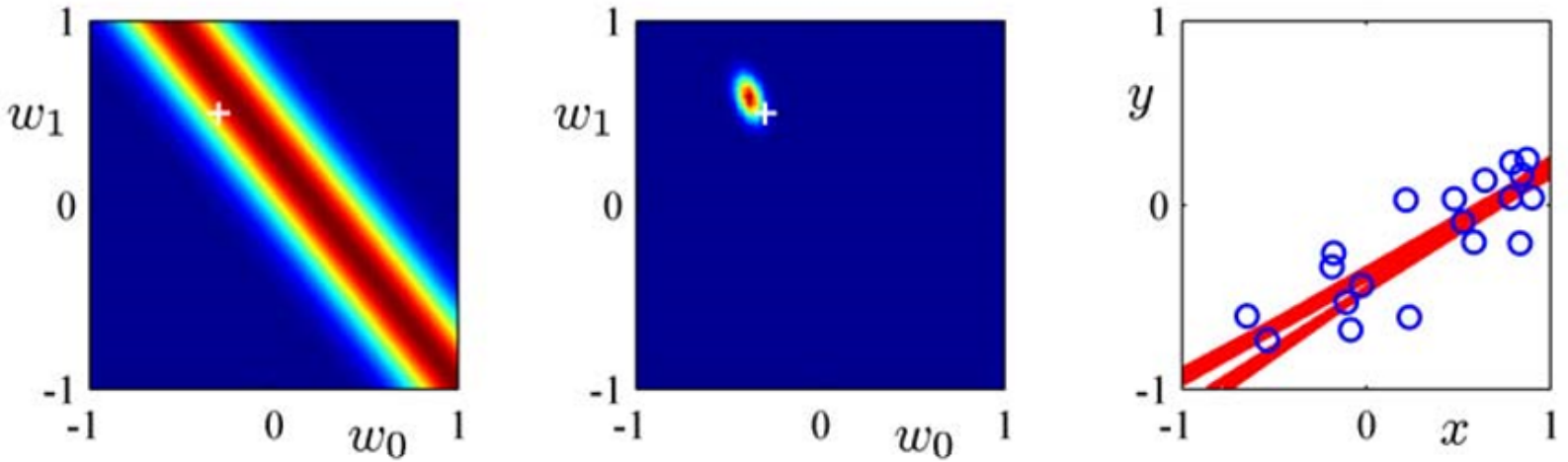

Figure: Observe 20 points (blue circle). The true parameters $(-0.3, 0.5)$ are marked as white cross. $\mathbf{w}$ draw from posterior. [C. Bishop 2006]

Predictive Distribution

Given new values of $\mathbf{x}$, making prediction for t which requires we evaluate the predictive distribution:

$$p(t|\mathbf{t}, \alpha, \beta) = \int p(t|\mathbf{w}, \beta)p(\mathbf{w}|\mathbf{t}, \alpha, \beta)d\mathbf{w}$$

Predictive Distribution

Given new values of $\mathbf{x}$, making prediction for t which requires we evaluate the predictive distribution:

$$p(t|\mathbf{t}, \alpha, \beta) = \int p(t|\mathbf{w}, \beta)p(\mathbf{w}|\mathbf{t}, \alpha, \beta)d\mathbf{w}$$

The predictive distribution has the following form:

$$p(t|\mathbf{t}, \alpha, \beta) = \mathrm{N}(t|\mathbf{m}_N^T\phi(\mathbf{x}), \sigma_N^2(\mathbf{x}))$$

Predictive Distribution

Given new values of $\mathbf{x}$, making prediction for t which requires we evaluate the predictive distribution:

$$p(t|\mathbf{t}, \alpha, \beta) = \int p(t|\mathbf{w}, \beta)p(\mathbf{w}|\mathbf{t}, \alpha, \beta)d\mathbf{w}$$

The predictive distribution has the following form:

$$p(t|\mathbf{t}, \alpha, \beta) = \mathrm{N}(t|\mathbf{m}_N^T\phi(\mathbf{x}), \sigma_N^2(\mathbf{x}))$$

where variance of the prediction distribution is given by:

$$\sigma_N^2(\mathbf{x}) = \frac{1}{\beta} + \phi(\mathbf{x})^T\mathbf{S}_N\phi(\mathbf{x})$$

Predictive Distribution

Given new values of $\mathbf{x}$, making prediction for t which requires we evaluate the predictive distribution:

$$p(t|\mathbf{t}, \alpha, \beta) = \int p(t|\mathbf{w}, \beta)p(\mathbf{w}|\mathbf{t}, \alpha, \beta)d\mathbf{w}$$

The predictive distribution has the following form:

$$p(t|\mathbf{t}, \alpha, \beta) = \mathrm{N}(t|\mathbf{m}_N^T\phi(\mathbf{x}), \sigma_N^2(\mathbf{x}))$$

where variance of the prediction distribution is given by:

$$\sigma_N^2(\mathbf{x}) = \frac{1}{\beta} + \phi(\mathbf{x})^T\mathbf{S}_N\phi(\mathbf{x})$$

- First term represents the noise of the data.
- Second term represents the uncertainty associated with parameter $\mathbf{w}$.

Prediction Function

Given predictive distribution $p(t|\mathbf{t}, \alpha, \beta) = \mathrm{N}(t|\mathbf{m}_N^T \phi(\mathbf{x}), \sigma_N^2(\mathbf{x}))$, we make prediction by using posterior mean:

$$y(\mathbf{x}, \mathbf{m}_N) = \mathbf{m}_N^T \phi(\mathbf{x})$$

where $\mathbf{m}_N$ is given by:

$$\begin{aligned} \mathbf{m}_N &= \beta \mathbf{S}_N \mathbf{\Phi}^T \mathbf{t} \\ \mathbf{S}_N^{-1} &= \alpha \mathbf{I} + \beta \mathbf{\Phi}^T \mathbf{\Phi} \end{aligned}$$

Use Kernel

We have:

$$\begin{aligned}\mathbf{m}_N &= \beta(\alpha\mathbf{I} + \beta\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{t} \\ &= \beta\mathbf{\Phi}^T(\alpha\mathbf{I} + \beta\mathbf{\Phi}\mathbf{\Phi}^T)^{-1}\mathbf{t}(?)\end{aligned}$$

Use Kernel

We have:

$$\begin{aligned}\mathbf{m}_N &= \beta(\alpha\mathbf{I} + \beta\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{t} \\ &= \beta\mathbf{\Phi}^T(\alpha\mathbf{I} + \beta\mathbf{\Phi}\mathbf{\Phi}^T)^{-1}\mathbf{t}(?)\end{aligned}$$

The prediction function becomes:

$$\begin{aligned}y(\mathbf{x}, \mathbf{m}_N) &= \mathbf{m}_N^T\phi(\mathbf{x}) = \phi(\mathbf{x})^T\mathbf{m}_N \\ &= \beta\phi(\mathbf{x})^T\mathbf{\Phi}^T(\alpha\mathbf{I} + \beta\mathbf{\Phi}\mathbf{\Phi}^T)^{-1}\mathbf{t}\end{aligned}$$

Use Kernel

We have:

$$\begin{aligned}\mathbf{m}_N &= \beta(\alpha\mathbf{I} + \beta\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{t} \\ &= \beta\mathbf{\Phi}^T(\alpha\mathbf{I} + \beta\mathbf{\Phi}\mathbf{\Phi}^T)^{-1}\mathbf{t}(?)\end{aligned}$$

The prediction function becomes:

$$\begin{aligned}y(\mathbf{x}, \mathbf{m}_N) &= \mathbf{m}_N^T\phi(\mathbf{x}) = \phi(\mathbf{x})^T\mathbf{m}_N \\ &= \beta\phi(\mathbf{x})^T\mathbf{\Phi}^T(\alpha\mathbf{I} + \beta\mathbf{\Phi}\mathbf{\Phi}^T)^{-1}\mathbf{t}\end{aligned}$$

Define Kernel $K = \mathbf{\Phi}\mathbf{\Phi}^T$, where
$K_{ij} = K(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)) =< \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) >$

Representer Results

Recall for posterior mean, we have:

$$\mathbf{m}_N = \beta \mathbf{\Phi}^T (\alpha \mathbf{I} + \beta \mathbf{\Phi}\mathbf{\Phi}^T)^{-1} \mathbf{t}$$

Using defined kernel K, we have:

$$\mathbf{m}_N = \beta \mathbf{\Phi}^T \mathbf{c} = \beta \sum_{n=1}^{N} c_n \phi(\mathbf{x}_n)$$

where

$$\begin{aligned} \mathbf{c} &= (\alpha \mathbf{I} + \beta \mathbf{\Phi}\mathbf{\Phi}^T)^{-1} \mathbf{t} \\ &= (\alpha \mathbf{I} + \beta K)^{-1} \mathbf{t} \\ &= (c_1, c_2, \ldots, c_N) \end{aligned}$$

Representer Results

Recall for posterior mean, we have:

$$\mathbf{m}_N = \beta \mathbf{\Phi}^T (\alpha \mathbf{I} + \beta \mathbf{\Phi}\mathbf{\Phi}^T)^{-1} \mathbf{t}$$

Using defined kernel K, we have:

$$\mathbf{m}_N = \beta \mathbf{\Phi}^T \mathbf{c} = \beta \sum_{n=1}^{N} c_n \phi(\mathbf{x}_n)$$

where

$$\begin{aligned} \mathbf{c} &= (\alpha \mathbf{I} + \beta \mathbf{\Phi}\mathbf{\Phi}^T)^{-1} \mathbf{t} \\ &= (\alpha \mathbf{I} + \beta K)^{-1} \mathbf{t} \\ &= (c_1, c_2, \ldots, c_N) \end{aligned}$$

The posterior mean of parameter distribution can be represented as *linear combinations* of training examples.

Kernel Regression

We have $\mathbf{m}_N = \beta \sum_{n=1}^{N} c_n \phi(\mathbf{x}_n)$

Kernel Regression

We have $\mathbf{m}_N = \beta \sum_{n=1}^{N} c_n \phi(\mathbf{x}_n)$

The prediction function:

$$\begin{aligned} y(\mathbf{x}, \mathbf{m}_N) &= \phi(\mathbf{x})^T \mathbf{m}_N \\ &= \beta \phi(\mathbf{x})^T \sum_{n=1}^{N} c_n \phi(\mathbf{x}_n) \\ &= \beta \sum_{n=1}^{N} c_n < \phi(\mathbf{x}), \phi(\mathbf{x}_n) > \\ &= \sum_{n=1}^{N} \beta c_n K(\phi(\mathbf{x}), \phi(\mathbf{x}_n)) \end{aligned}$$

Kernel Regression

We have $\mathbf{m}_N = \beta \sum_{n=1}^{N} c_n \phi(\mathbf{x}_n)$

The prediction function:

$$\begin{aligned} y(\mathbf{x}, \mathbf{m}_N) &= \phi(\mathbf{x})^T \mathbf{m}_N \\ &= \beta \phi(\mathbf{x})^T \sum_{n=1}^{N} c_n \phi(\mathbf{x}_n) \\ &= \beta \sum_{n=1}^{N} c_n < \phi(\mathbf{x}), \phi(\mathbf{x}_n) > \\ &= \sum_{n=1}^{N} \beta c_n K(\phi(\mathbf{x}), \phi(\mathbf{x}_n)) \end{aligned}$$

We do not need explicitly define the feature function $\phi(\mathbf{x})$, only need to know the kernel K which defines the relation/distance between two features.

Generalized Linear Model

Classification

- Classification goal: given input vector $\mathbf{x}$, assign it to one of K discrete classes $\mathcal{C}_k$, where $k = 1, \ldots, K$.

Classification

- Classification goal: given input vector $\mathbf{x}$, assign it to one of K discrete classes $\mathcal{C}_k$, where $k = 1, \ldots, K$.
- The input space of $\mathbf{x}$ is divided into *decision regions* whose boundaries are called *decision boundaries* or *decision surfaces*.

Classification

- Classification goal: given input vector $\mathbf{x}$, assign it to one of K discrete classes $\mathcal{C}_k$, where $k = 1, \ldots, K$.
- The input space of $\mathbf{x}$ is divided into *decision regions* whose boundaries are called *decision boundaries* or *decision surfaces*.
- We consider *linear model* for classification. The decision surfaces are *linear function* of input vector $\mathbf{x}$.

Classification

- Classification goal: given input vector $\mathbf{x}$, assign it to one of K discrete classes $\mathcal{C}_k$, where $k = 1, \ldots, K$.
- The input space of $\mathbf{x}$ is divided into *decision regions* whose boundaries are called *decision boundaries* or *decision surfaces*.
- We consider *linear model* for classification. The decision surfaces are *linear function* of input vector $\mathbf{x}$.
- Data sets whose classes can be separated exactly by linear decision surfaces are said to be *linearly separable*.

Three Approaches for Classification

- Construct **discriminant function** that directly assign $\mathbf{x}$ to a specific class.
- Model posterior $p(\mathcal{C}_k|\mathbf{x})$:
  - **Probabilistic discriminative approach**: directly model $p(\mathcal{C}_k|\mathbf{x})$.
  - **Probabilistic generative approach**: model class-conditional density $p(\mathbf{x}|\mathcal{C}_k)$, together with prior $p(\mathcal{C}_k)$.

Generalized Linear Models

For regression, the simplest linear model would be:

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$$

Generalized Linear Models

For regression, the simplest linear model would be:

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$$

The classification, however, we need to predict discrete class labels or posterior probabilities that lie in the range $(0, 1)$.

Generalized Linear Models

For regression, the simplest linear model would be:

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$$

The classification, however, we need to predict discrete class labels or posterior probabilities that lie in the range $(0, 1)$.

$$y(\mathbf{x}) = f(\mathbf{w}^T\mathbf{x} + w_0)$$

Generalized Linear Models

For regression, the simplest linear model would be:

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$$

The classification, however, we need to predict discrete class labels or posterior probabilities that lie in the range $(0, 1)$.

$$y(\mathbf{x}) = f(\mathbf{w}^T\mathbf{x} + w_0)$$

- $f(.)$ is known as *activation function*.
- The decision surface correspond to $y(\mathbf{x}) =$ constant, so that $\mathbf{w}^T\mathbf{x} + w_0 =$ constant, thus the decision surfaces are linear function of $\mathbf{x}$. Thus *generalized linear model*.
- The model used is no longer linear in the parameter $\mathbf{w}$ because of nonlinear function $f(.)$.

Fixed Basis Functions

- Similarly as the previous discussed linear regression model, we could use nonlinear transformation of the inputs using a vector of basis functions $\phi(\mathbf{x})$.
- The resulting decision boundaries will be linear in the feature space ϕ, and these correspond to nonlinear decision boundaries in the original $\mathbf{x}$ space.
- Classes that are linearly separable in the feature space $\phi(\mathbf{x})$ need not be linearly separable in the original observation space $\mathbf{x}$.

Example

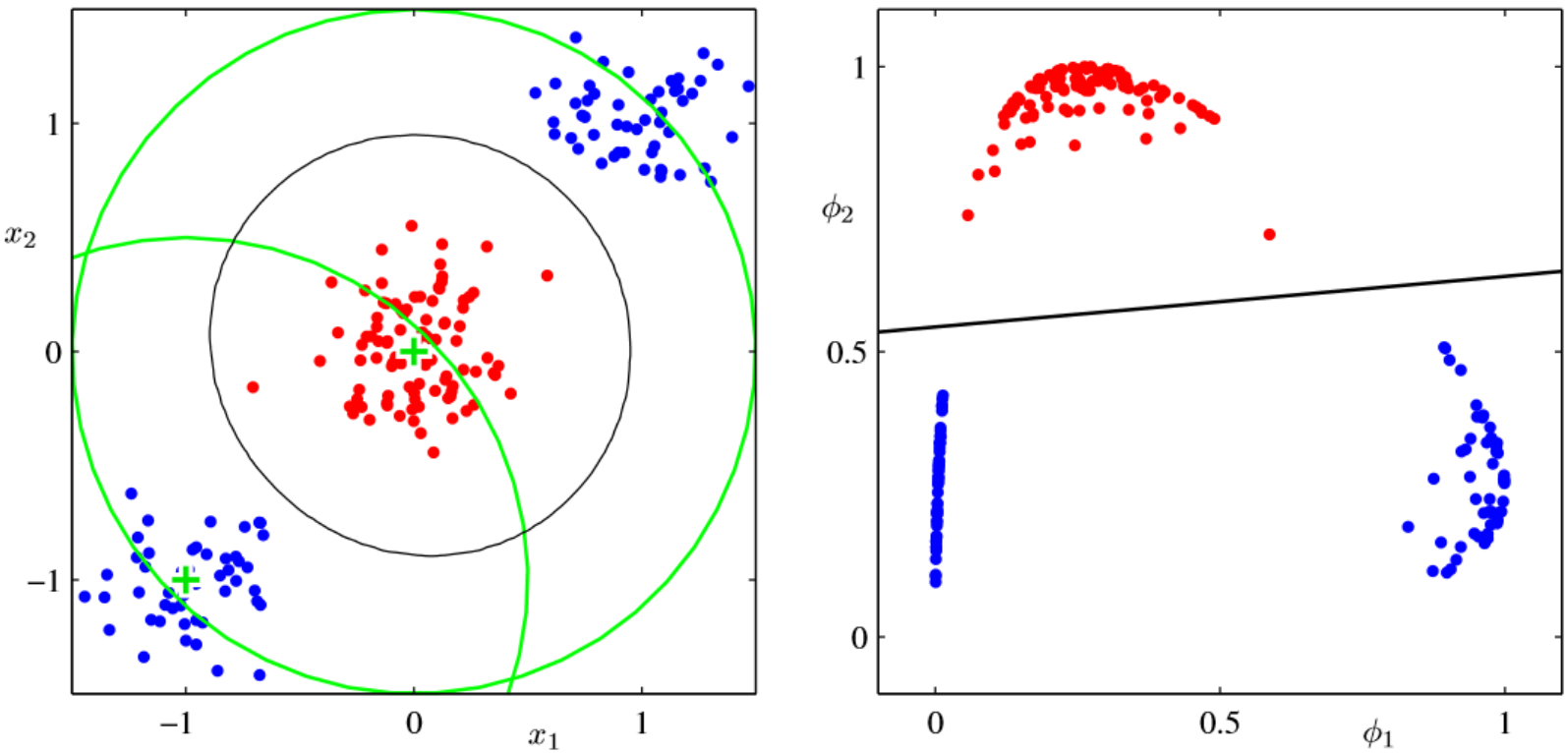

Figure: Left: original $\mathbf{x}$ space with nonlinear decision boundary. Right: feature space $\phi(\mathbf{x})$ with linear decision boundary. [C. Bishop 2006]

Sigmoid Function as f

The sigmoid function is widely used nonlinear function that maps all real input variable to range $(0, 1)$.

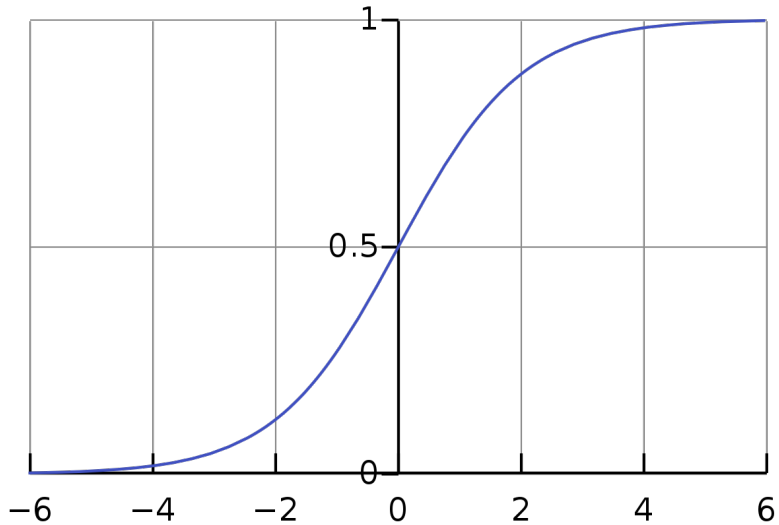

Figure: Sigmoid function. (Wiki)

- $\sigma(a) = \frac{1}{1+\exp(-a)}$
- $\frac{d\sigma}{da} = \sigma(1-\sigma)$

Logistic Regression

Consider problem of two-class (i.e., $\mathcal{C}_1$ and $\mathcal{C}_2$) classification. Using *discriminative approach*, we tend to maximize a likelihood function defined through the posterior $p(\mathcal{C}_k|\mathbf{x})$.

Logistic Regression

Consider problem of two-class (i.e., $\mathcal{C}_1$ and $\mathcal{C}_2$) classification. Using *discriminative approach*, we tend to maximize a likelihood function defined through the posterior $p(\mathcal{C}_k|\mathbf{x})$.

We define:

$$\begin{aligned} p(\mathcal{C}_1|\phi) &= y(\phi) = \sigma(\mathbf{w}^T\phi) \\ p(\mathcal{C}_2|\phi) &= 1 - p(\mathcal{C}_1|\phi) \end{aligned}$$

Logistic Regression

Consider problem of two-class (i.e., $\mathcal{C}_1$ and $\mathcal{C}_2$) classification. Using *discriminative approach*, we tend to maximize a likelihood function defined through the posterior $p(\mathcal{C}_k|\mathbf{x})$.

We define:

$$\begin{aligned} p(\mathcal{C}_1|\phi) &= y(\phi) = \sigma(\mathbf{w}^T\phi) \\ p(\mathcal{C}_2|\phi) &= 1 - p(\mathcal{C}_1|\phi) \end{aligned}$$

- $\sigma(.)$: sigmoid function
- The above model is called *logistic regression*, note that the model is actually used for classification.

MLE for Logistic Regression

For dataset $\{\phi_n, t_n\}$, where $t_n \in \{0, 1\}$, $\phi_n = \phi(\mathbf{x}_n)$ and $n = 1, \ldots, N$. The likelihood function is:

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n}(1 - y_n)^{1-t_n}$$

where $\mathbf{t} = (t_1, \ldots, t_N)^T$ and $y_n = p(\mathcal{C}_1|\phi_n) = \sigma(\mathbf{w}^T\phi_n)$.

MLE for Logistic Regression

For dataset $\{\phi_n, t_n\}$, where $t_n \in \{0, 1\}$, $\phi_n = \phi(\mathbf{x}_n)$ and $n = 1, \ldots, N$. The likelihood function is:

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n} (1 - y_n)^{1-t_n}$$

where $\mathbf{t} = (t_1, \ldots, t_N)^T$ and $y_n = p(\mathcal{C}_1|\phi_n) = \sigma(\mathbf{w}^T \phi_n)$.

Get *cross entropy error function* by taking negative log:

$$E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^{N} \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\}$$

MLE for Logistic Regression

For dataset $\{\phi_n, t_n\}$, where $t_n \in \{0, 1\}$, $\phi_n = \phi(\mathbf{x}_n)$ and $n = 1, \ldots, N$. The likelihood function is:

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n}(1 - y_n)^{1-t_n}$$

where $\mathbf{t} = (t_1, \ldots, t_N)^T$ and $y_n = p(\mathcal{C}_1|\phi_n) = \sigma(\mathbf{w}^T\phi_n)$.

Get *cross entropy error function* by taking negative log:

$$E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^{N}\{t_n \ln y_n + (1 - t_n)\ln(1 - y_n)\}$$

Taking gradient (making use of derivative of sigmoid function):

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N}(y_n - t_n)\phi_n$$

Implement MLE for Logistic Regression

For homework assignment, you need to implement and build the classifier by training logistic regression using MLE.

- For Iris dataset, it has three classes: Iris Setosa, Iris Versicolour and Virginica. But we consider classification on two-classes, i.e., Virginica and non-Virginica
- For Iris dataset, it has four features. But we only use the first two.
- $p(\mathcal{C}_1|\mathbf{w}) = \sigma(\mathbf{w}^T\mathbf{x})$ and $p(\mathcal{C}_2|\mathbf{w}) = 1 - p(\mathcal{C}_1|\mathbf{w})$, use MLE to learn $\mathbf{w}$.
- Use (stochastic) gradient descent: $\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta\nabla E(\mathbf{w}^{(\tau)})$.
- After you learned optimal $\hat{\mathbf{w}}$, predict the label to be 1 if $\sigma(\hat{\mathbf{w}}^T\mathbf{x}) > 0.5$, otherwise, predict the label to be 0.