Linear Models for Regression

Sargur Srihari
srihari@cedar.buffalo.edu

Plan of Discussion

- Discuss supervised learning starting with regression
- Goal of regression:
  - predict value of one or more target variables t
  - Given d-dimensional vector $\mathbf{x}$ of input variables

Topics in Linear Regression

- Regression Terminology
- Polynomial Curve Fitting with Scalar input
- Linear Model with D inputs
- Linear Basis Function Models
  - Gaussian Radial Basis Functions
- Maximum Likelihood and Least Squares
- Geometry of Least Squares
- Sequential Learning
- Regularized Least Squares

Regression Terminology

- Regression
  - Predict a numerical value t given some input
    - Learning algorithm has to output function $f : \mathrm{R}^n \rightarrow \mathrm{R}$
      - where n =no of input variables
  - Ex: expected claim amount an insured person will make (used to set insurance premiums) or prediction of future prices of securities
    - Also used for algorithmic trading
- Classification
  - If t value is a label (categories): $f : \mathrm{R}^n \rightarrow \{1,..,k\}$
- Ordinal Regression
  - Discrete values, ordered categories

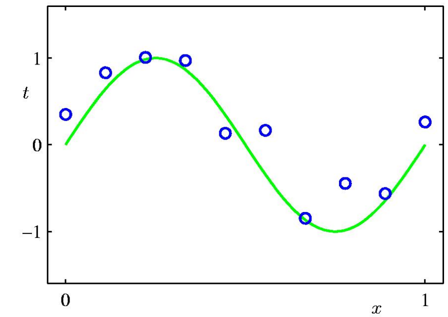
Polynomial Curve Fitting with a Scalar

- Simplest form of regression:
  - With a single input variable x
  - $y(x,\mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$

    M is the order of the polynomial,

    x^j denotes x raised to the power j,

    Coefficients $w_0,\ldots,w_M$ are collectively denoted by vector $\mathbf{w}$



Training data set
N=10, Input x, target t

  - Task: Learn $\mathbf{w}$ from training data $D = \{(x_i, t_i)\}$, $i = 1,..,N$
    - Can be done by minimizing an error function that minimizes the misfit between $y(x,\mathbf{w})$ for any given w and training data
    - One simple choice of error function is sum of squares of error between predictions $y(x_n,\mathrm{w})$ for each data point x_n and corresponding target values t_n so that we minimize $$E(\mathrm{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,\mathrm{w}) - t_n\}^2$$
    - It is zero when function $y(x,\mathrm{w})$ passes exactly through each training data point

Regression with multiple inputs

- Generalization
  - Predict value of continuous target variable t given value of d input variables $\mathrm{x}=[x_1,..x_D]$
  - t can also be a set of variables (multiple regression)
  - Linear functions of adjustable parameters
    - Specifically linear combinations of nonlinear functions of input variable
- Polynomial curve fitting is good only for:
  - Single input variable scalar x
  - It cannot be easily generalized to several variables, as we will see

Simplest Linear Model with D inputs

- Regression with D input variables

$$y(\mathrm{x},\mathrm{w}) = w_0 + w_1 x_1 + .. + w_d\, x_d = \mathrm{w}^T \mathrm{x}$$

where $\mathrm{x}=(x_1,..,x_D)^{\mathrm{T}}$ are the input variables

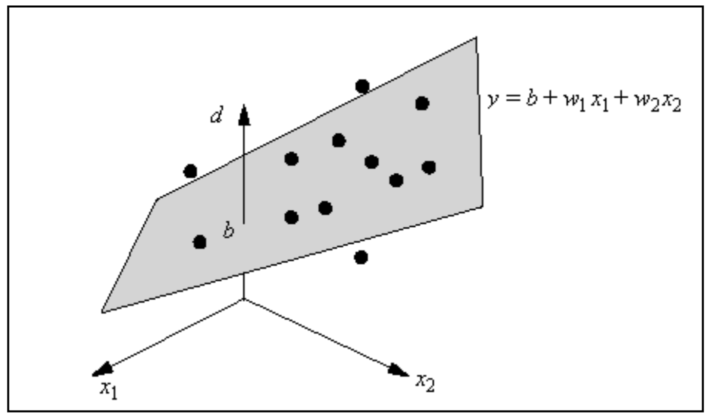
This differs from
Linear Regression with one variable
and Polynomial Reg with one variable

- Called Linear Regression since it is a linear function of
  - parameters $w_0,..,w_D$
  - input variables $x_1,..,x_D$

- Significant limitation since it is a linear function of input variables
  - In the one-dimensional case this amounts a straight-line fit (degree-one polynomial)
  - $y(x,\mathrm{w}) = w_0 + w_1 x$

Fitting a Regression Plane

- Assume t is a function of inputs $x_1,\ x_2, \ldots x_D$
  Goal: find best linear regressor of t on all inputs
  - Fitting a hyperplane through N input samples
  - For D =2:



x_1	x_2	t
1	2	2
2	5	1
2	3	2
2	2	2
3	4	1
3	5	3
4	6	2
5	5	3
5	6	4
5	7	3
6	8	4
7	6	2
8	4	4
8	9	3
9	8	4

- Being a linear function of input variables imposes limitations on the model
  - Can extend class of models by considering fixed nonlinear functions of input variables

Basis Functions

- In many applications, we apply some form of fixed-preprocessing, or feature extraction, to the original data variables
- If the original variables comprise the vector $\mathbf{x}$, then the features can be expressed in terms of basis functions $\{\phi_j(\mathbf{x})\}$
  - By using nonlinear basis functions we allow the function $y(\mathbf{x},\mathbf{w})$ to be a nonlinear function of the input vector $\mathbf{x}$
    - They are linear functions of parameters (gives them simple analytical properties), yet are nonlinear wrt input variables

Linear Regression with M Basis Functions

- Extended by considering nonlinear functions of input variables

$$y(\mathrm{x},\mathrm{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathrm{x})$$

  - where $\phi_j(\mathrm{x})$ are called Basis functions
  - We now need M weights for basis functions instead of D weights for features
  - With a dummy basis function $\phi_0(\mathrm{x})=1$ corresponding to the bias parameter w_0, we can write

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

  - where $\mathrm{w}=(w_0,w_1,..,w_{M-1})$ and $\Phi=(\phi_0,\phi_1,..,\phi_{M-1})^{\mathrm{T}}$

- Basis functions allow non-linearity with D input variables

Choice of Basis Functions

- Many possible choices for basis function:
  1. Polynomial regression
     - Good only if there is only one input variable
  2. Gaussian basis functions
  3. Sigmoidal basis functions
  4. Fourier basis functions
  5. Wavelets

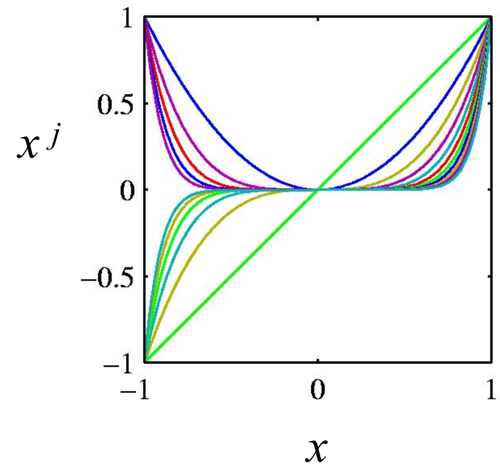
1. Polynomial Basis for one variable

- Linear Basis Function Model

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

- Polynomial Basis (for single variable x)

  $\phi_j(x) = x^j$ with degree M-1 polynomial



- Disadvantage
  - Global:
    - changes in one region of input space affects others
  - Difficult to formulate
    - Number of polynomials increases exponentially with M
  - Can divide input space into regions
    - use different polynomials in each region:
    - equivalent to spline functions

Can we use Polynomial with D variables? (Not practical!)

- Consider (for a vector $\mathbf{x}$) the basis: $\phi_j(\mathbf{x}) = \|\mathbf{x}\|^j = \left[\sqrt{x_1^2 + x_2^2 + .. + x_d^2}\right]^j$
  - $\mathbf{x}=(2,1)$ and $\mathbf{x}=(1,2)$ have the same squared sum, so it is unsatisfactory
  - Vector is being converted into a scalar value thereby losing information
- Better polynomial approach:
  - Polynomial of degree M-1 has terms with variables taken none, one, two… M-1 at a time.
  - Use multi-index $j=(j_1,j_2,..j_D)$ such that $j_1+j_2+..j_D \leq M$-1
  - For a quadratic (M=3) with three variables (D=3)

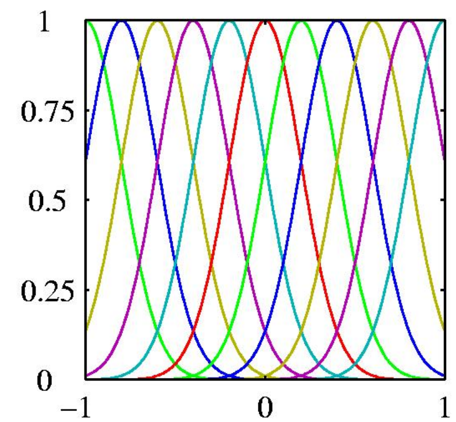
$$y(\mathbf{x},\mathbf{w}) = \sum_{(j_1,j_2,j_3)} w_j \phi_j(\mathbf{x}) = w_0 + w_{1,0,0}x_1 + w_{0,1,0}x_2 + w_{0,0,1}x_3 + w_{1,1,0}x_1x_2 + w_{1,0,1}x_1x_3 +$$

$$w_{0,1,1}x_2x_3 + w_{2,0,0}x_1^2 + w_{0,2,0}x_2^2 + w_{0,0,2}x_3^2$$

  - Number of quadratic terms is $1+D+D(D\text{-}1)/2+D$
  - For D=46, it is 1128
- Better to use Gaussian kernel, discussed next

2. Gaussian Radial Basis Functions

- Gaussian $$\phi_j(x) = \exp\left(\frac{(x-\mu_j)^2}{2\sigma^2}\right)$$
  - Does not necessarily have a probabilistic interpretation
  - Usual normalization term is unimportant
    - since basis function is multiplied by weight w_j

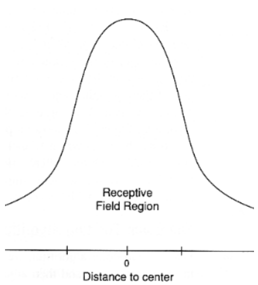


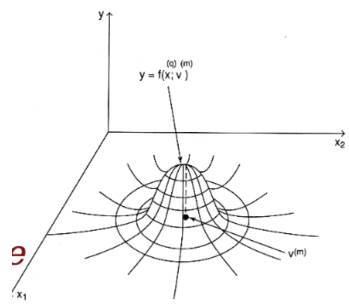
- Choice of parameters
  - μ_j govern the locations of the basis functions
    - Can be an arbitrary set of points within the range of the data
      - Can choose some representative data points
  - σ governs the spatial scale
    - Could be chosen from the data set e.g., average variance
- Several variables
  - A Gaussian kernel would be chosen for each dimension
  - For each j a different set of means would be needed– perhaps chosen from the data

$$\phi_j(\mathrm{x}) = \exp\left(-\frac{1}{2}(\mathrm{x}-\boldsymbol{\mu}_j)^t\Sigma^{-1}(\mathrm{x}-\boldsymbol{\mu}_j)\right)$$

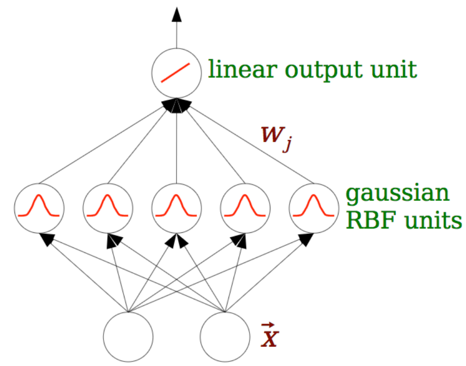
Biological Inspiration for RBFs

- Nervous system contains many examples
  - Local receptive fields in visual cortex





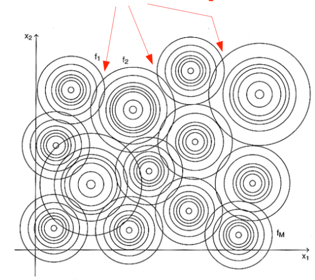
- RBF Network



$$Output = \sum_j w_j \cdot \exp\left(\frac{-\|\vec{x}-\vec{\mu}_j\|^2}{\sigma_j^2}\right)$$

- Receptive fields overlap
- So there is usually more than one unit active
- But for given input, total no. of active units is small

Tiling the input space



- Determining centers
  - k-means clustering
    - Choose k cluster centers
    - Mark each training point as captured by cluster to which it is closest
    - Move each cluster center to mean of points it captured
- Determining variance σ^2

  Global: σ =mean distance between each unit j and its closest neighbor

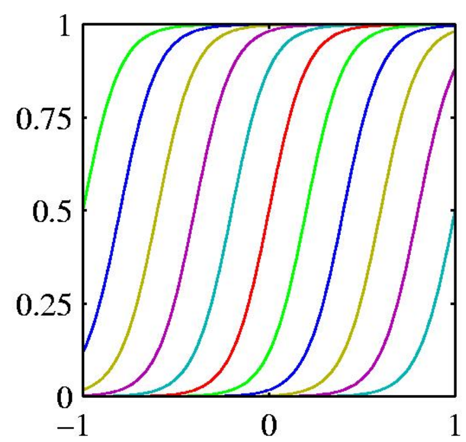
  P-nearest neighbor: set each σ_j so that there is certain overlap with P closest neighbors of unit j

3. Sigmoidal Basis Function

- Sigmoid $$\phi_j(x) = \sigma\left(\frac{x-\mu_j}{s}\right) \quad \text{where} \quad \sigma(a) = \frac{1}{1+\exp(-a)}$$
- Equivalently, $\tanh$ because it is related to logistic sigmoid by

$$\tanh(a) = 2\sigma(a) - 1$$

Logistic Sigmoid
For different μ_j



4. Other Basis Functions

- Fourier
  - Expansion in sinusoidal functions
  - Infinite spatial extent
- Signal Processing
  - Functions localized in time and frequency
  - Called *wavelets*
    - Useful for lattices such as images and time series
- Further discussion independent of choice of basis including $\phi(\mathrm{x}) = \mathrm{x}$

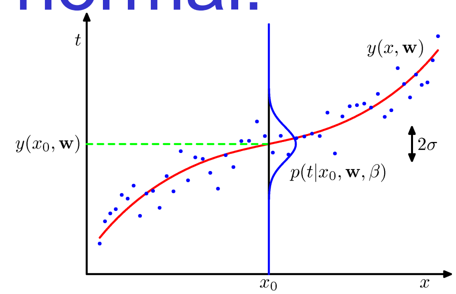
Relationship between Maximum Likelihood and Least Squares

- Will show that Minimizing sum-of-squared errors is the same as maximum likelihood solution under a Gaussian noise model
- Target variable is a scalar t given by deterministic function $y(\mathrm{x},\mathrm{w})$ with additive Gaussian noise ε

  $t = y(\mathrm{x},\mathrm{w}) + \varepsilon$

  – which is a *zero-mean* Gaussian with *precision* β
- Thus distribution of t is univariate normal:

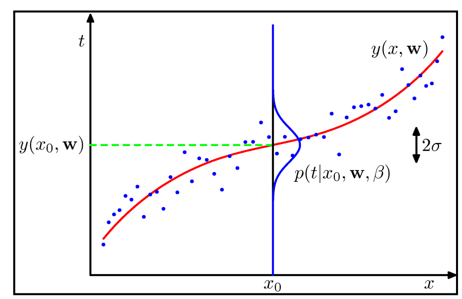
$$p(t|\mathrm{x},\mathrm{w},\beta)=N(t \mid \underbrace{y(\mathrm{x},\mathrm{w})}_{\text{mean}},\underbrace{\beta^{-1}}_{\text{variance}})$$



Likelihood Function

- Data set:
  - Input $X=\{\mathrm{x}_1,..\mathrm{x}_N\}$ with target $\mathrm{t} = \{t_1,..t_N\}$
  - Target variables t_n are scalars forming a vector of size N
- Likelihood of the target data
  - It is the probability of observing the data assuming they are independent
  - since $p(t|\mathrm{x},\mathrm{w},\beta)=N(t \mid y(\mathrm{x},\mathrm{w}),\beta^{-1})$
  - and $y(\mathrm{x},\mathrm{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathrm{x}) = \mathrm{w}^T \phi(\mathrm{x})$

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



Log-Likelihood Function

- Likelihood

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

- Log-likelihood

$$\ln p(\mathrm{t} \mid \mathrm{w}, \beta) = \sum_{n=1}^{N} \ln N\left(t_n \mid \mathrm{w}^T \phi(\mathrm{x}_n), \beta^{-1}\right)$$

  – Using standard univariate Gaussian

$$N(x \mid \mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$$

$$\ln p(\mathrm{t} \mid \mathrm{w}, \beta) = \frac{N}{2} \ln \beta - \frac{N}{2} \ln 2\pi - \beta E_D(\mathrm{w})$$

  - Where

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

Sum-of-squares Error Function

  - With Gaussian basis functions

$$\phi_j(\mathrm{x}) = \exp\left(\frac{(\mathrm{x} - \boldsymbol{\mu}_j)^t (\mathrm{x} - \boldsymbol{\mu}_j)}{2s^2} \right)$$

Maximizing Log-Likelihood Function

- Log-likelihood

$$\ln p(\mathrm{t} \mid \mathrm{w},\beta) = \sum_{n=1}^{N} \ln N\left(t_n \mid \mathrm{w}^T \phi(\mathrm{x}_n), \beta^{-1}\right)$$
$$= \frac{N}{2}\ln\beta - \frac{N}{2}\ln 2\pi - \beta E_D(\mathrm{w})$$

  – where

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

- Therefore, maximizing likelihood is equivalent to minimizing $E_D(\mathrm{w})$

Determining max likelihood solution

- The likelihood function has the form

$$\ln p(\mathrm{t} \mid \mathrm{w}, \beta) = \frac{N}{2} \ln \beta - \frac{N}{2} \ln 2\pi - \beta E_D(\mathrm{w})$$

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

- We will show that the maximum likelihood solution has a closed form

– Take derivative of $\ln p(\mathrm{t} \mid \mathrm{w}, \beta)$ wrt w and set equal to zero and solve for w

  – or equivalently just the derivative of $E_D(\mathrm{w})$

Gradient of Log-likelihood wrt w

$$\boxed{\nabla \ln p(\mathrm{t} \mid \mathrm{w}, \beta) = \beta \sum_{n=1}^{N} \left\{ \ t_n - \mathrm{w}^T \phi(\mathrm{x}_n) \ \right\} \ \phi(\mathrm{x}_n)^T}$$

-which is obtained from log-likelihood expression and by using calculus result

$$\nabla_w [-\frac{1}{2}\left(a - wb\right)^2] = (a - wb)b$$

- Gradient is set to zero and we solve for w

$$\boxed{0 = \sum_{n=1}^{N} t_n \phi(\mathrm{x}_n) - \mathrm{w}^{\mathrm{T}} \left(\sum_{n=1}^{N} \phi(\mathrm{x}_n) \phi(\mathrm{x}_n)^T \right)}$$

as shown in next slide

- Second derivative will be negative making this a maximum

Max Likelihood Solution for $\mathbf{w}$

- Solving for $\mathbf{w}$ we obtain:

$$\boxed{\mathbf{w}_{ML} = \Phi^{+}\mathbf{t}}$$

$X=\{\mathbf{x}_1,..\mathbf{x}_N\}$ are samples (vectors of d variables)
$\mathbf{t} = \{t_1,..t_N\}$ are targets (scalars)

where $\Phi^{+} = (\Phi^T\Phi)^{-1}\Phi^T$ is the Moore-Penrose pseudo inverse of the N x M Design Matrix Φ whose elements are given by $\Phi_{nj}=\phi_j(\mathbf{x}_n)$

- Known as the normal equations for the least squares problem

Design Matrix: Rows correspond to N samples, Columns to M basis functions

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

Pseudo inverse: generalization of notion of matrix inverse to non-square matrices
If design matrix is square and invertible. then pseudo-inverse is same as inverse

$\phi_i(\mathbf{x}_n)$ are M basis functions, e.g., Gaussians centered on M data points

$$\boxed{\phi_j(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_j)^t\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_j)\right)}$$

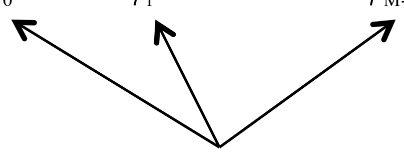
Design Matrix

$\longleftarrow M$ Basis functions $\longrightarrow$

$$\Phi = \begin{pmatrix} \phi_0(\mathrm{x}_1) & \phi_1(\mathrm{x}_1) & \dots & \phi_{M-1}(\mathrm{x}_1) \\ \phi_0(\mathrm{x}_2) & & & \\ & & & \\ \phi_0(\mathrm{x}_N) & & & \phi_{M-1}(\mathrm{x}_N) \end{pmatrix} \quad \updownarrow N \text{ Data}$$

Represented as N-dim vectors

φ_0 φ_1 $\varphi_{M\text{-}1}$



What is the role of Bias parameter w_0?

- Sum-of squares error function is: $E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^T\phi(\mathbf{x}_n)\}^2$
  - Substituting: $y(\mathbf{x},\mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j\phi_j(\mathbf{x})$ we get:

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\{t_n - w_0 - \sum_{j=1}^{M-1} w_j\phi_j(\mathbf{x}_n)\right\}^2$$

  - Setting derivatives wrt w_0 equal to zero and solving for w_0

$$w_0 = \bar{t} - \sum_{j=1}^{M-1} w_j\overline{\phi_j}$$ where $$\bar{t} = \frac{1}{N}\sum_{n=1}^{N} t_n$$ and $$\bar{\phi}_j = \frac{1}{N}\sum_{n=1}^{N}\phi_j(\mathbf{x}_n)$$

    - First term is average of the N values of t
    - Second term is weighted sum of the average basis function values over N samples
- Thus bias w_0 compensates for difference between average target values and weighted sum of averages of basis function values

Maximum Likelihood for precision β

- We have determined m.l.e. solution for w using a probabilistic formulation
  - $p(t|\mathbf{x},\mathbf{w},\beta)=N(t|y(\mathbf{x},\mathbf{w}),\beta^{-1})$
  - With log-likelihood

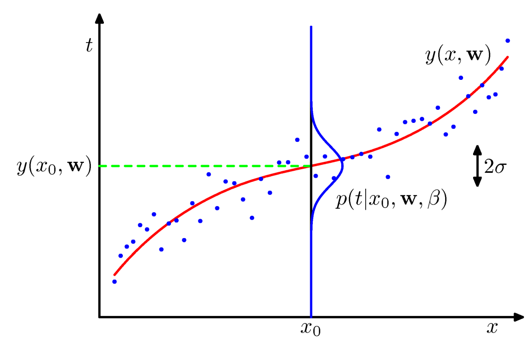
$$\boxed{\mathbf{w}_{ML} = \Phi^{+}\mathbf{t}}$$

$$\ln p(\mathbf{t}\,|\,\mathbf{w},\beta) = \frac{N}{2}\ln\beta - \frac{N}{2}\ln 2\pi - \beta E_D(\mathbf{w})$$

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

- Taking gradient wrt β gives

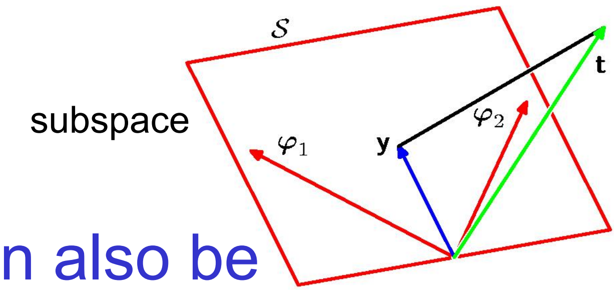
$$\boxed{\frac{1}{\beta_{ML}} = \frac{1}{N}\sum_{n=1}^{N}\left\{ t_n - \mathbf{w}_{ML}^T\phi(\mathbf{x}_n)\right\}^2}$$



- Thus Inverse of the noise precision gives *Residual variance of the target values around the regression function*

Geometry of Least Squares

- Geometrical Interpretation of Least Squares Solution instructive
- Consider N-dim space with axes t_n
  so that $\mathrm{t} = (t_1,\ldots,t_\mathrm{N})^\mathrm{T}$ is a vector in this space



- Each basis $\boldsymbol{\phi}_j(\mathrm{x}_\mathrm{n})$ evaluated at N points can also be represented as a vector in the same space
- ϕ_j corresponds to j^{th} column of $\boldsymbol{\Phi}$, whereas $\phi(\mathrm{x}_\mathrm{n})$ corresponds to the the n^{th} row of $\boldsymbol{\Phi}$
- If the no of basis functions is smaller than the no of data points
  - i.e., $M<N$ then the M vectors $\phi_j(\mathrm{x}_n)$ will span linear subspace S of dim M
- Define $\boldsymbol{y}$ to be an N-dim vector whose n^{th} element is $y(\mathrm{x}_n,\mathrm{w})$
- Sum-of-squares error is equal to squared Euclidean distance between $\boldsymbol{y}$ and t
- Solution w corresponds to $\boldsymbol{y}$ that lies in subspace S that is closest to t
  - Corresponds to orthogonal projection of t onto S

Difficulty of Direct solution

- Direct solution of normal equations

$$\boxed{\mathbf{w}_{ML} = \Phi^{+}\mathbf{t}} \qquad \boxed{\Phi^{+} = (\Phi^{T}\Phi)^{-1}\Phi^{T}}$$

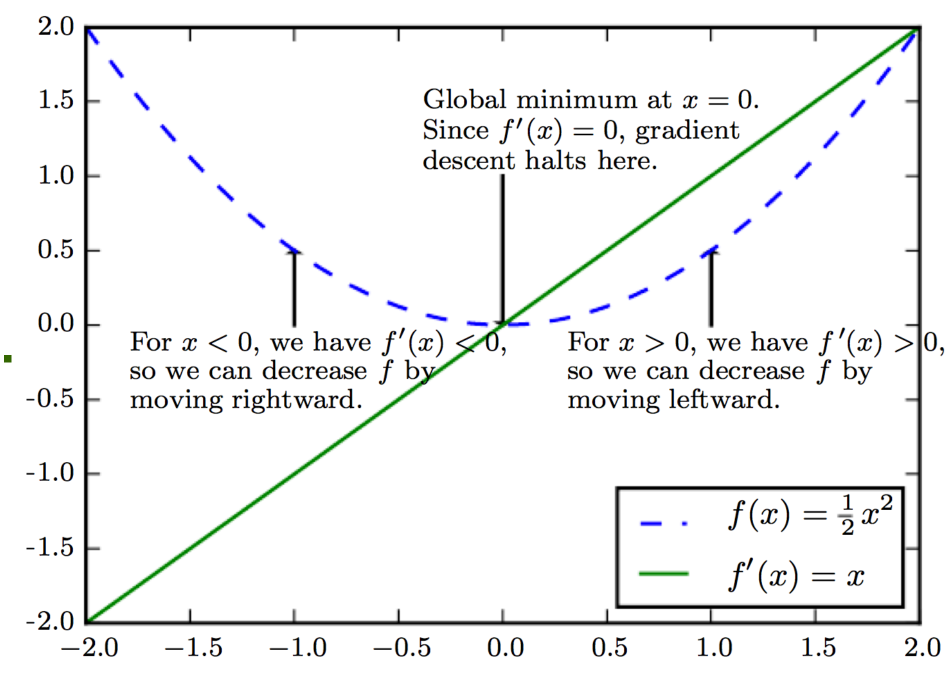
- This direct solution can lead to numerical difficulties
  - When $\Phi^{T}\Phi$ is close to singular (determinant=0)
  - When two basis functions are collinear parameters can have large magnitudes
- Not uncommon with real data sets
- Can be addressed using
  - Singular Value Decomposition
  - Addition of regularization term ensures matrix is non-singular

Method of Gradient Descent

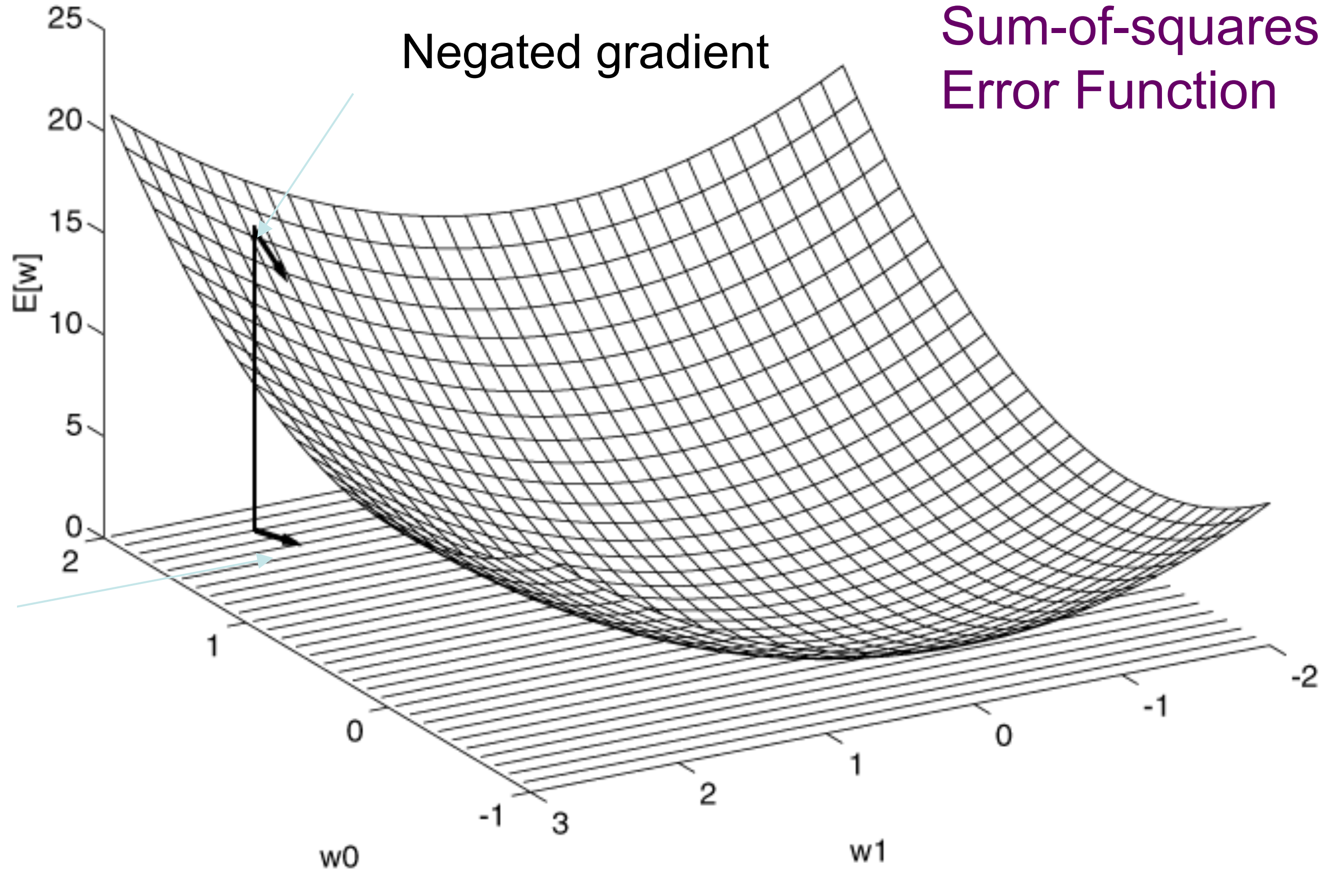
- Criterion $f(\boldsymbol{x})$ is minimized by moving from the current solution in direction of the negative of gradient
- Steepest descent proposes a new point

$$x' = x - \varepsilon \nabla_x f(x)$$

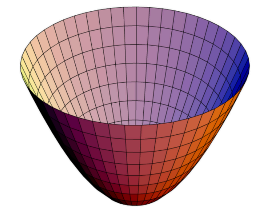
  – where ε is the learning
  – rate, a positive scalar.
  – Set to a small constant.



Direction of Steepest Descent



Sum-of-squares Error Function



Gradient Descent for Regression

- Error function $E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\{ t_n - \mathbf{w}^T\phi(\mathbf{x}_n) \right\}^2$ sums over data
  - Denoting $E_D(\mathbf{w}) = \Sigma_n E_n$, update parameter vector $\mathbf{w}$ using

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

    - where τ is the iteration number and η is a learning rate parameter
- Substituting for the derivative $\nabla E_n = -\sum_{n=1}^{N}\left\{ t_n - \mathbf{w}^T\phi(\mathbf{x}_n) \right\}\phi(\mathbf{x}_n)^T$

where $\phi_n = \phi(\mathbf{x}_n)$

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \eta(t_n - \mathbf{w}^{(\tau)T}\phi_n)\phi_n$$

    - $\mathbf{w}$ is initialized to some starting vector $\mathbf{w}^{(0)}$
    - η chosen with care to ensure convergence
- Known as *Least Mean Squares* Algorithm

Sequential (On-line) Learning

- Maximum likelihood solution is

$$\mathrm{w}_{ML} = (\Phi^T \Phi)^{-1} \Phi^T \mathrm{t}$$

- It is a batch technique
  - Processing entire training set in one go
- It is computationally expensive for large data sets
  - Due to huge $N \times M$ Design matrix Φ
- Solution is to use a sequential algorithm where samples are presented one at a time (or a minibatch at a time)
- Called stochastic gradient descent

Regularized Least Squares

- As model complexity increases, e.g., degree of polynomial or no.of basis functions, then it is likely that we overfit
- One way to control overfitting is not to limit complexity but to add a regularization term to the error function
- Error function to minimize takes the form

$$E(\mathrm{w}) = E_D(\mathrm{w}) + \lambda E_W(\mathrm{w})$$

  - where λ is the *regularization coefficient* that controls relative importance of data-dependent error $E_D(\mathrm{w})$ and regularization term $E_W(\mathrm{w})$

Simplest Regularizer is weight decay

- Regularized least squares is

$$\boxed{E(\mathrm{w}) = E_D(\mathrm{w}) + \lambda E_W(\mathrm{w})}$$

- Simple form of regularization term is

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

  – Thus total error function becomes

$$\boxed{E(\mathrm{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\{ t_n - \mathrm{w}^T\phi(x_n) \right\}^2 + \frac{\lambda}{2}\mathrm{w}^T\mathrm{w}}$$

- This regularizer is called *weight decay*
  – because in sequential learning weight values decay towards zero unless supported by data
- Also, the error function remains a *quadratic function* of w, so exact minimizer found in closed form

Closed-form Solution with Regularizer

- Error function with *quadratic regularizer is*,

$$\boxed{E(\mathrm{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\{ t_n - \mathrm{w}^T\phi(x_n) \right\}^2 + \frac{\lambda}{2}\mathrm{w}^T\mathrm{w}}$$

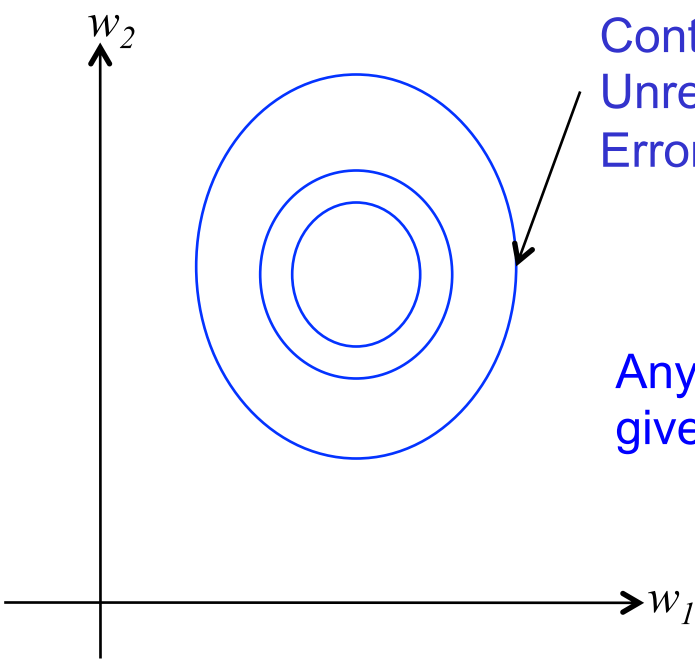
- Its exact minimizer can be found in closed form
  - By setting gradient wrt w to zero and solving for w

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

- This is a simple extension of the least squared solution

$$\boxed{\mathrm{w}_{ML} = (\Phi^T\Phi)^{-1}\Phi^T \mathrm{t}}$$

Geometric Interpretation of Regularizer



Contours of Unregularized Error function

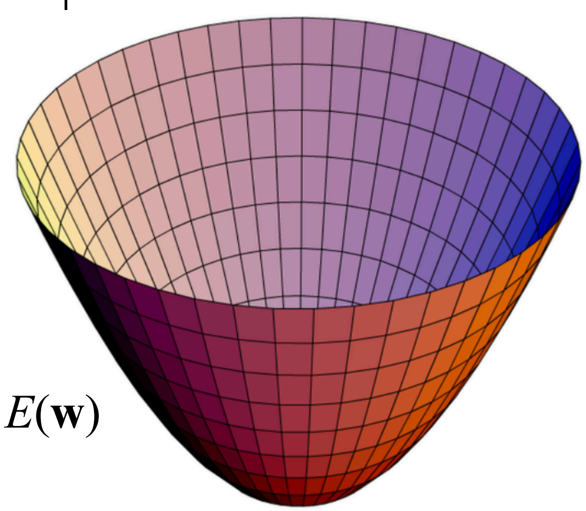
In *unregularized* case:
we are trying to find w that minimizes

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

Any value of w on contour gives same error

In *regularized* case:
choose that value of w subject to the constraint

$$\sum_{j=1}^{M} | w_j |^2 \leq \eta$$



$E(\mathbf{w})$

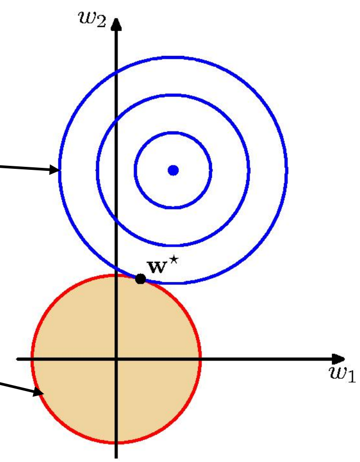
We don't want the weights to become too large

The two approaches related by Lagrange multipliers

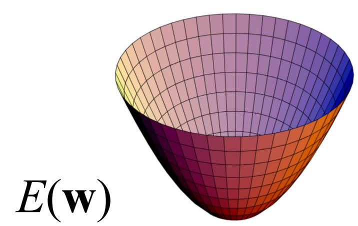
$$E(\mathrm{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\{ t_n - \mathrm{w}^T\phi(x_n) \right\}^2 + \frac{\lambda}{2}\mathrm{w}^T\mathrm{w}$$

Minimization of Unregularized Error subject to constraint

- Blue: Contours of unregularized error function
- Constraint region
- w^* is optimum value



Minimization With quadratic Regularizer, $q=2$



A more general regularizer

- Regularized Error

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

- Where $q=2$ corresponds to the *quadratic* regularizer

  $q=1$ is known as *lasso*

- Lasso has the property that if λ is sufficiently large some of the coefficients w_j are driven to zero leading to a sparse model in which the corresponding basis functions play no role

Contours of regularization term

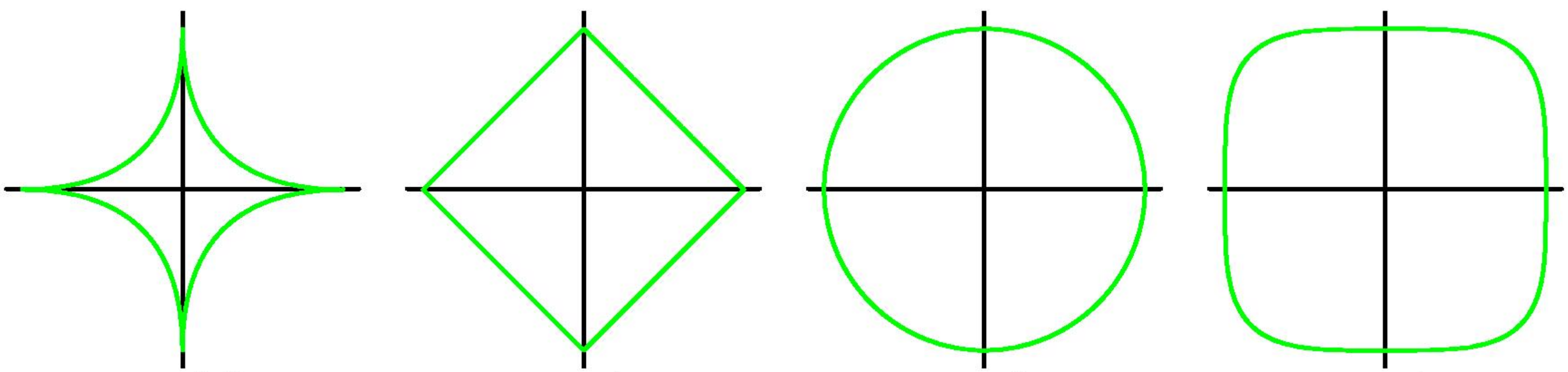
$$\frac{1}{2}\sum_{n=1}^{N}\{\, t_n - \mathrm{w}^T\phi(\mathrm{x}_n)\,\}^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|\,w_j\,|^q$$

- Contours of regularization term $|w_j|^q$ for values of q

Space of w_1, w_2

Any choice along the contour has the same value of w

$\sqrt{w_1} + \sqrt{w_2} = const$ $\quad w_1 + w_2 = const$ $\quad w_1^{\,2} + w_2^{\,2} = const$ $\quad w_1^{\,4} + w_2^{\,4} = const$



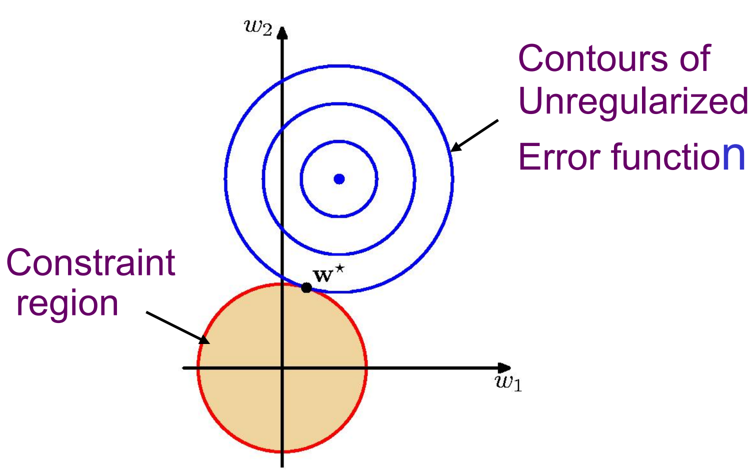
$q = 0.5$ $\quad q = 1$ $\quad q = 2$ $\quad q = 4$

Lasso Quadratic

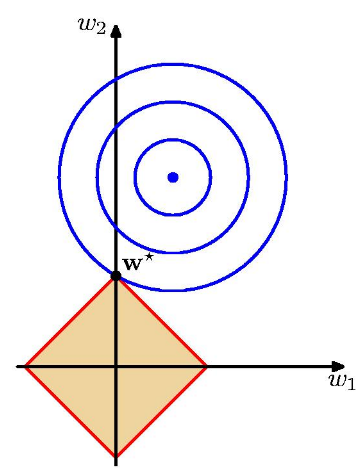
Sparsity with Lasso constraint

- With q=1 and λ is sufficiently large, some of the coefficients w_j are driven to zero
- Leads to a sparse model
  - where corresponding basis functions play no role
- Origin of sparsity is illustrated here:

Quadratic solution where w_1* and w_0* are nonzero



Minimization with Lasso Regularizer A sparse solution with w_1*=0



Regularization: Conclusion

- Regularization allows
  - complex models to be trained on small data sets
  - without severe over-fitting
- It limits model complexity
  - i.e., how many basis functions to use?
- Problem of limiting complexity is shifted to
  - one of determining suitable value of regularization coefficient

Linear Regression Summary

- Linear Regression with M basis functions:

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

$$\phi_j(\mathrm{x}) = \exp\left(-\frac{1}{2}(\mathrm{x}-\mu_j)^t \Sigma^{-1}(\mathrm{x}-\mu_j)\right)$$

- Objective Function *without/with* regularization is

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

$$E_D(\mathrm{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\{ t_n - \mathrm{w}^T \phi(x_n) \right\}^2 + \frac{\lambda}{2}\mathrm{w}^T\mathrm{w}$$

- Closed-form ML solution is:

$$\mathrm{w}_{ML} = (\Phi^T\Phi)^{-1}\Phi^T \mathrm{t}$$

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

$$\Phi = \begin{pmatrix} \phi_0(\mathrm{x}_1) & \phi_1(\mathrm{x}_1) & \ldots & \phi_{M-1}(\mathrm{x}_1) \\ \phi_0(\mathrm{x}_2) & & & \\ & & & \\ \phi_0(\mathrm{x}_N) & & & \phi_{M-1}(\mathrm{x}_N) \end{pmatrix}$$

- Gradient Descent: $\mathrm{w}^{(\tau+1)} = \mathrm{w}^{(\tau)} - \eta \nabla E_n$

$$\nabla E_n = -\sum_{n=1}^{N}\left\{ t_n - \mathrm{w}^T \phi(\mathrm{x}_n) \right\} \phi(\mathrm{x}_n)^T$$

$$\nabla E_n = \left[-\sum_{n=1}^{N}\left\{ t_n - \mathrm{w}^T \phi(\mathrm{x}_n) \right\} \phi(\mathrm{x}_n)^T \right] + \lambda \mathrm{w}$$

Returning to LeToR Problem

- Try:
- Several Basis Functions
- Quadratic Regularization
- Express results as E_{RMS}
  - rather than as squared error $E(\mathrm{w}^*)$ or as Error Rate with thresholded results

$$E_{RMS} = \sqrt{2E(\mathrm{w}^*)/N}$$

Multiple Outputs

- Several target variables $\mathbf{t} = (t_1, .., t_K)$ $K > 1$
- Can be treated as multiple (K) independent regression problems
  - Different basis functions for each component of $\mathbf{t}$
- More common solution: same set of basis functions to model all components of target vector $\mathbf{y}(\mathbf{x},\mathbf{w}) = \mathbf{W}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x})$
  - where $\mathbf{y}$ is a K-dim column vector, $\mathbf{W}$ is a M **x** K matrix of weights and $\phi(\mathbf{x})$ is a M-dimensional column vector with with elements $\phi_j(\mathbf{x})$

Solution for Multiple Outputs

- Set of observations $\mathbf{t}_1,..,\mathbf{t}_N$ are combined into a matrix T of size N x K such that the n^{th} row is given by $\mathrm{t}_n^{\mathrm{T}}$
- Combine input vectors $\mathrm{x}_1,..,\mathrm{x}_N$ into matrix X
- Log-likelihood function is maximized
- Solution is similar: $\mathrm{W}_{\mathrm{ML}}=(\Phi^{\mathrm{T}}\Phi)^{-1}\Phi^{\mathrm{T}}\mathrm{T}$