

Linear Models for Regression

Karan Nathwani

Topics in Linear Regression

- What is regression?
 - Polynomial Curve Fitting with Scalar input
 - Linear Basis Function Models $\omega_{\text{opt}} = R^{-1} P$
- Maximum Likelihood and Least Squares
- Stochastic Gradient Descent
- Regularized Least Squares

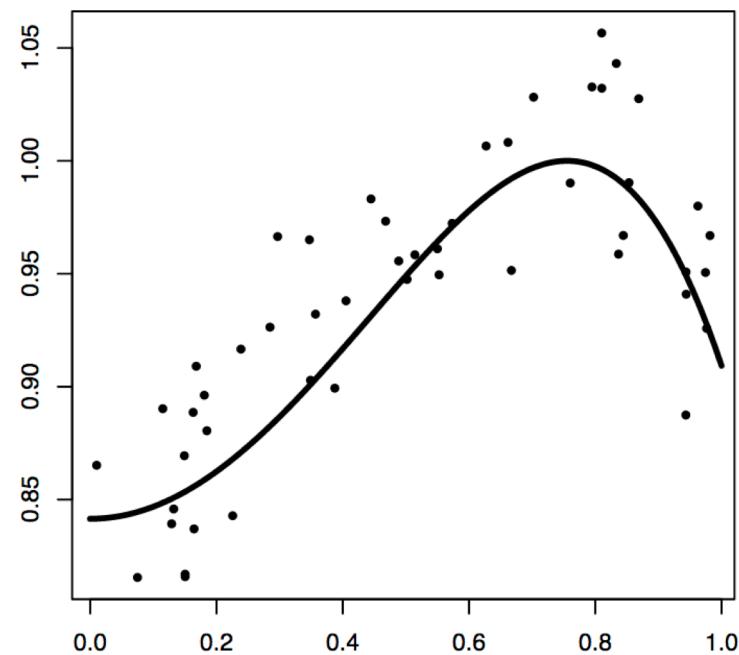
$$\begin{aligned} f &:= E[X^T X] \\ P &:= E[X^T d] \end{aligned}$$

The regression task

- It is a supervised learning task
- Goal of regression:
 - predict value of one or more target variables t
 - given d -dimensional vector \mathbf{x} of input variables
 - With dataset of known inputs and outputs
 - $(\mathbf{x}_1, t_1), \dots (\mathbf{x}_N, t_N)$
 - Where \mathbf{x}_i is an input (possibly a vector) known as the predictor
 - t_i is the target output (or response) for case i which is real-valued
 - Goal is to predict t from \mathbf{x} for some future test case
 - We are not trying to model the distribution of \mathbf{x}
 - We don't expect predictor to be a linear function of \mathbf{x}
 - So ordinary linear regression of inputs will not work
 - We need to allow for a nonlinear function of \mathbf{x}
 - We don't have a theory of what form this function to take

An example problem

- Fifty points generated (one-dimensional problem)
 - With x uniform from $(0,1)$
 - y generated from formula $y = \sin(1+x^2) + \text{noise}$
 - Where noise has $N(0, 0.03^2)$ distribution
 - Noise-free true function and data points are as shown



Applications of Regression

1. Expected claim amount an insured person will make (used to set insurance premiums) or prediction of future prices of securities
2. Also used for algorithmic trading

Policyholder age	Mean	Std. Error
60+	186.08	6.084
50-59	186.20	5.551
40-49	184.38	5.155
35-39	171.70	5.516
30-34	203.40	6.930
25-29	208.15	7.482
21-24	219.52	10.824
17-20	224.51	20.690

Vehicle group	Mean	Std. Error
D	239.87	9.513
C	200.98	6.168
B	177.71	4.663
A	178.91	5.871

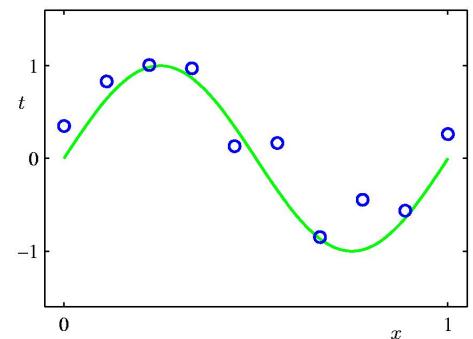
Vehicle age	Mean	Std. Error
10+	129.85	7.404
8-9	192.35	8.227
4-7	255.50	5.976
0-3	281.89	6.620

ML Terminology

- Regression
 - Predict a numerical value t given some input
 - Learning algorithm has to output function $f : \mathbb{R}^n \rightarrow \mathbb{R}$
 - where $n =$ no of input variables
- Classification
 - If t value is a label (categories): $f : \mathbb{R}^n \rightarrow \{1, \dots, k\}$
- Ordinal Regression
 - Discrete values, ordered categories

Polynomial Curve Fitting with a Scalar

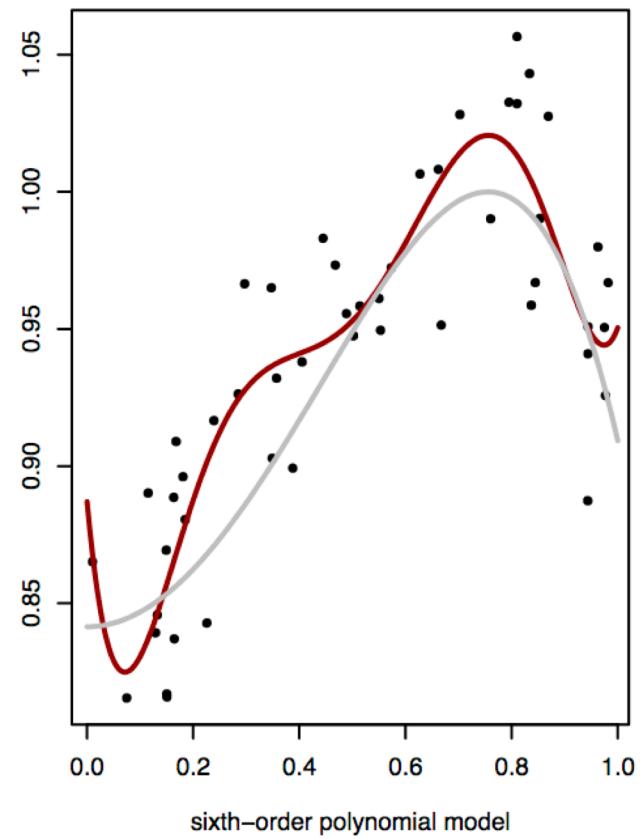
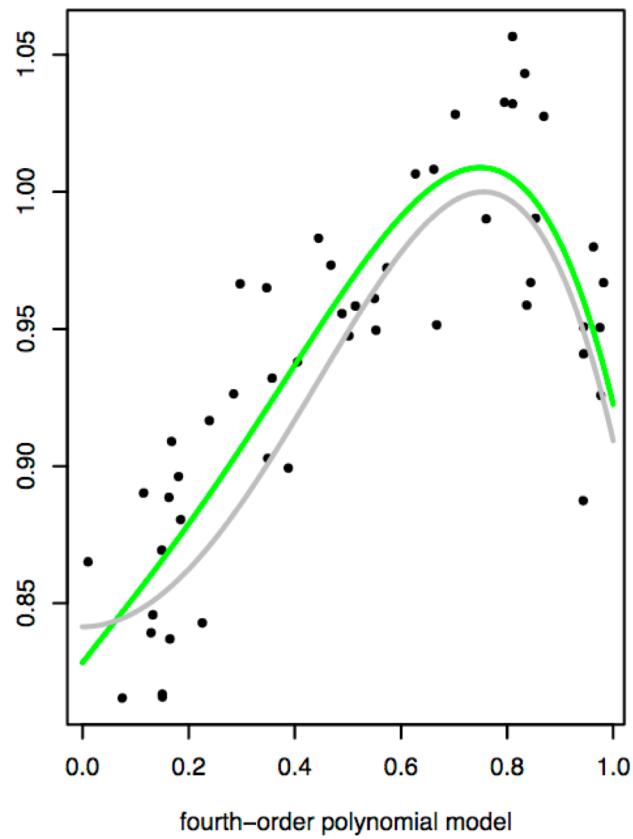
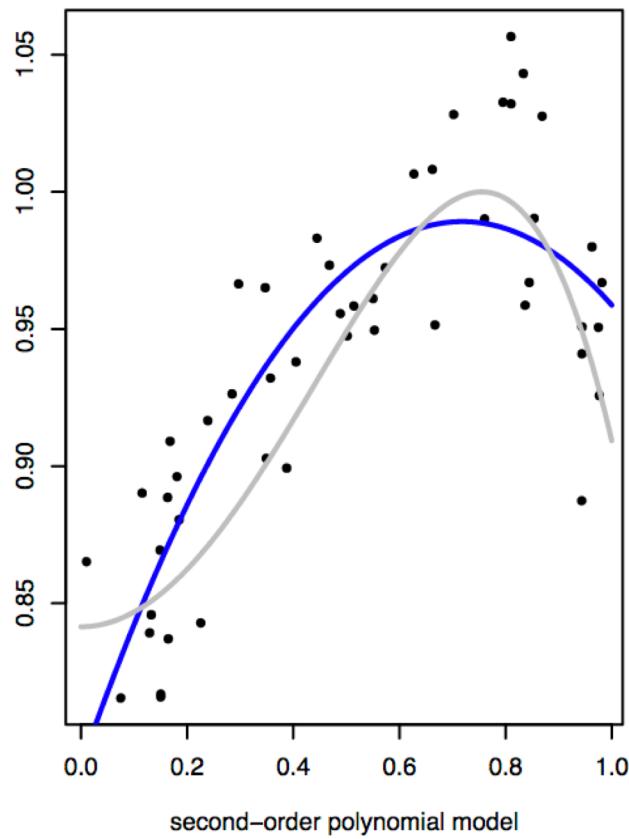
- With a single input variable x
- $y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \dots + w_M x^M = \sum_{j=0}^M w_j x^j$
 M is the order of the polynomial,
 x^j denotes x raised to the power j ,
Coefficients w_0, \dots, w_M are collectively denoted by vector \mathbf{w}
- Task: Learn \mathbf{w} from training data $D = \{(x_i, t_i)\}, i = 1, \dots, N$
 - Can be done by minimizing an error function that minimizes the misfit between $y(x, \mathbf{w})$ for any given \mathbf{w} and training data
 - One simple choice of error function is sum of squares of error between predictions $y(x_n, \mathbf{w})$ for each data point x_n and corresponding target values t_n so that we minimize
- It is zero when function $y(x, \mathbf{w})$ passes exactly through each training data point



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

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \left\{ y(x_n, \mathbf{w}) - t_n \right\}^2$$

Results with polynomial basis



Regression with multiple inputs

- Generalization
 - Predict value of continuous target variable t given value of D input variables $x = [x_1, \dots x_D]$
 - 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

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

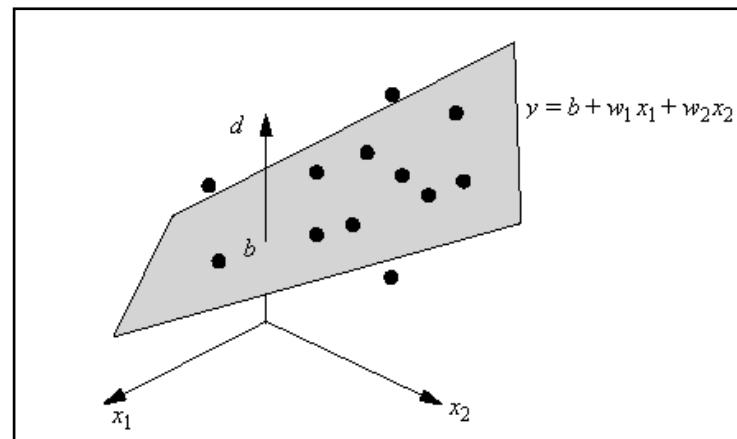
This differs from
Linear Regression with one variable
and Polynomial Reg with one variable

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

- Called Linear Regression since it is a linear function of
 - parameters w_0, \dots, w_D
 - input variables x_1, \dots, 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)
 - $y(x, \mathbf{w}) = w_0 + w_1 x$

Fitting a Regression Plane

- Assume t is a function of inputs x_1, x_2, \dots, 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 $\{\varphi_j(\mathbf{x})\}$
 - 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

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

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

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

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

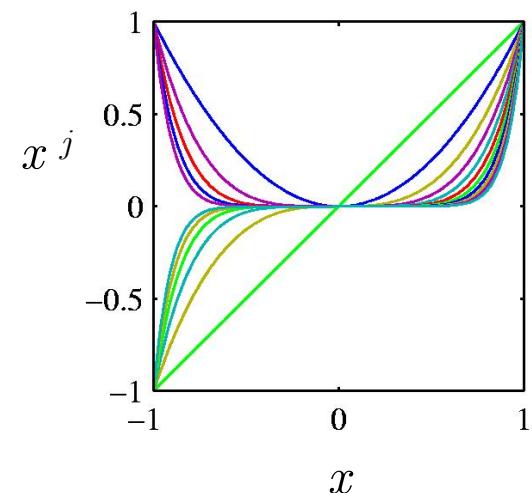
1. Polynomial Basis for one variable

- Linear Basis Function Model

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

- Polynomial Basis (for single variable x)

$\varphi_j(x) = x^j$ 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 x) the basis: $\phi_j(x) = \|x\|^j = \left[\sqrt{x_1^2 + x_2^2 + \dots + x_d^2} \right]^j$
 - $x=(2,1)$ and $x=(1,2)$ 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 $j=(j_1, j_2, \dots, j_D)$ such that $j_1 + j_2 + \dots + j_D \leq M-1$
 - For a quadratic ($M=3$) with three variables ($D=3$)

$$y(x, w) = \sum_{(j_1, j_2, j_3)} w_j \phi_j(x) = w_0 + w_{1,0,0}x_1 + w_{0,1,0}x_2 + w_{0,0,1}x_3 + w_{1,1,0}x_1x_2 + w_{1,0,1}x_1x_3 + w_{0,1,1}x_2x_3 + w_{2,0,0}x_1^2 + w_{0,2,0}x_2^2 + w_{0,0,2}x_3^2$$

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

Disadvantage of Polynomial

- Polynomials are *global* basis functions
 - Each affecting the prediction over the whole input space
- Often local basis functions are more appropriate

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

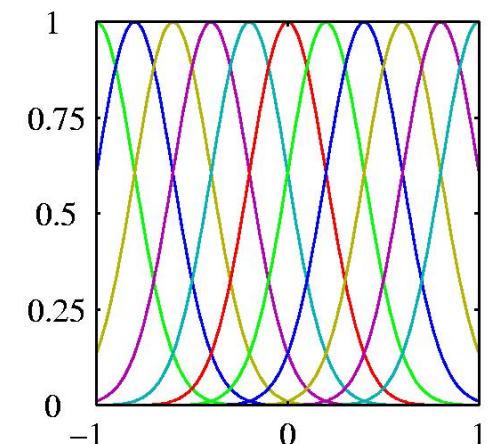
- **Choice of parameters**

- 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

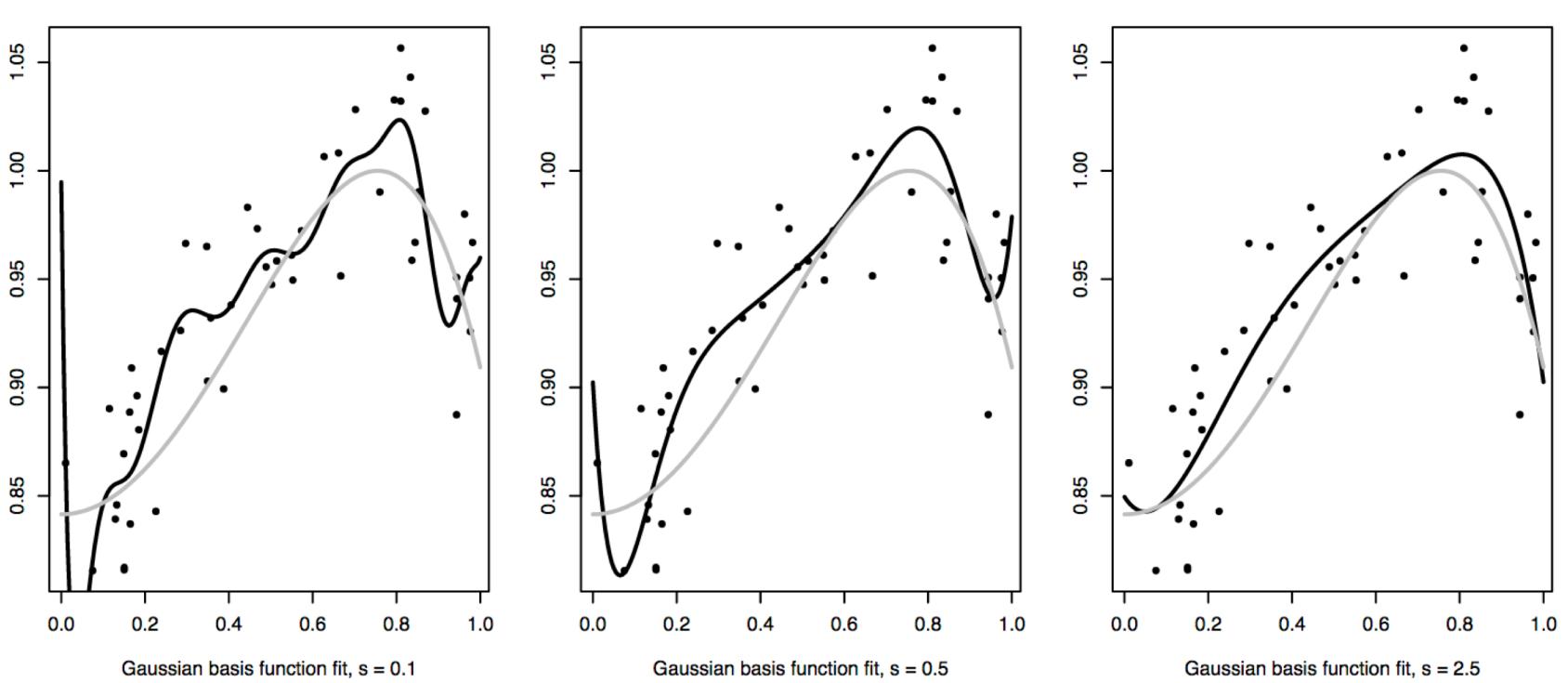
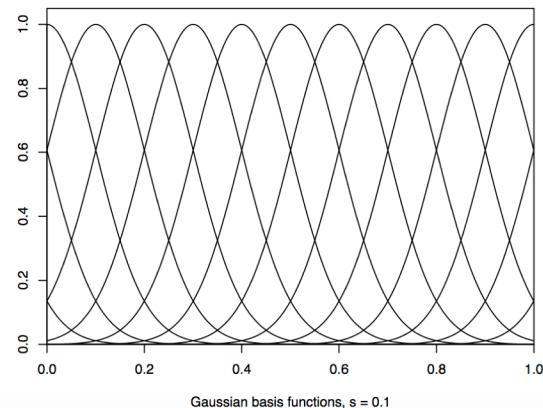
$$\phi_j(x) = \exp\left(-\frac{1}{2}(x - \mu_j)^t \Sigma^{-1} (x - \mu_j)\right)$$



Result with Gaussian Basis Functions

$$\phi_j(x) = \exp(-(x - \mu_j)^2 / 2s^2)$$

Basis functions for $s=0.1$, with the μ_j on a grid with spacing s



w_j for middle model:

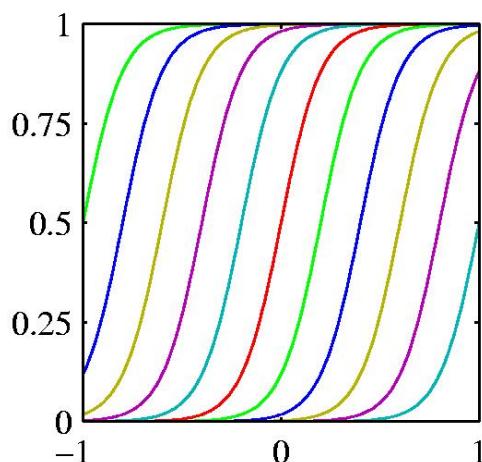
6856.5
-3544.1
-2473.7
-2859.8
-2637.7
-2861.5
-2468.0
-3558.4

3. Sigmoidal Basis Function

- Sigmoid $\phi_j(x) = \sigma\left(\frac{x - \mu_j}{s}\right)$ where $\sigma(a) = \frac{1}{1 + \exp(-a)}$
- 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 $\varphi(x) = x$

Relationship between Maximum Likelihood and Least Squares

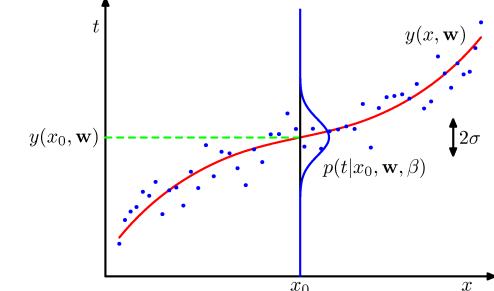
- Will show that Minimizing sum-of-squared errors is the same as maximum likelihood solution under a Gaussian noise model
- Target variable is a scalar t given by deterministic function $y(x, w)$ with additive Gaussian noise ε

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

– which is a *zero-mean Gaussian with precision β*

- Thus distribution of t is univariate normal:

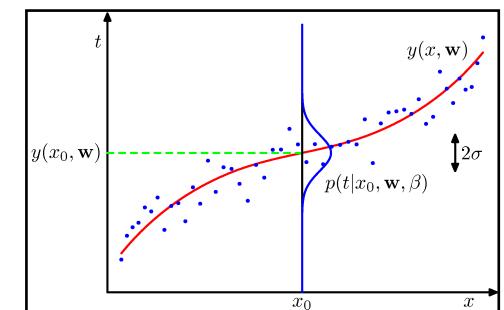
$$p(t|x, w, \beta) = N(t \mid \underbrace{y(x, w)}_{\text{mean}}, \underbrace{\beta^{-1}}_{\text{variance}})$$



Likelihood Function

- Data set:
 - Input $X = \{x_1, \dots, x_N\}$ with target $t = \{t_1, \dots, t_N\}$
 - 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 $p(t|x, w, \beta) = N(t | y(x, w), \beta^{-1})$
 - and $y(x, w) = \sum_{j=0}^{M-1} w_j \phi_j(x) = w^T \phi(x)$

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



Log-Likelihood Function

- Likelihood

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

- Log-likelihood

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

- Using standard univariate Gaussian

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

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

- Where

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

Sum-of-squares Error Function

- With Gaussian basis functions

$$\boldsymbol{\phi}_j(\mathbf{x}) = \exp\left(\frac{(\mathbf{x} - \boldsymbol{\mu}_j)^T (\mathbf{x} - \boldsymbol{\mu}_j)}{2s^2}\right)$$

Maximizing Log-Likelihood Function

- Log-likelihood

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

– where

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

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

Determining max likelihood solution

- The likelihood function has the form

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

- where
- We will show that the maximum likelihood solution has a closed form
- Take derivative of $\ln p(t | w, \beta)$ wrt w and set equal to zero and solve for w
 - or equivalently just the derivative of $E_D(w)$

Gradient of Log-likelihood wrt w

$$\nabla \ln p(t | w, \beta) = \beta \sum_{n=1}^N \left\{ t_n - w^T \phi(x_n) \right\} \phi(x_n)^T$$

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

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

- Gradient is set to zero and we solve for w

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

as shown in next slide

- Second derivative will be negative making this a maximum

Max Likelihood Solution for w

- Solving for w we obtain:

$$w_{ML} = \Phi^+ t$$

$X = \{x_1, \dots, x_N\}$ are samples
 (vectors of d variables)
 $t = \{t_1, \dots, t_N\}$ are targets (scalars)

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

- Known as the normal equations for the least squares problem

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

$$\Phi = \begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) & \dots & \phi_{M-1}(x_1) \\ \phi_0(x_2) & & & \\ \vdots & & & \\ \phi_0(x_N) & & & \phi_{M-1}(x_N) \end{pmatrix}$$

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

$\phi_i(x_n)$ are M basis functions, e.g., Gaussians centered on M data points

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

Design Matrix Φ

$\leftarrow M \text{ Basis functions} \rightarrow$

$$\Phi = \begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) & \dots & \phi_{M-1}(x_1) \\ \phi_0(x_2) & & & \\ \vdots & & & \\ \phi_0(x_N) & & & \phi_{M-1}(x_N) \end{pmatrix}$$

↑ N Data

Represented as N -dim vectors

The diagram shows three vectors originating from a single point (the origin). The first vector is labeled ϕ_0 , the second ϕ_1 , and the third ϕ_{M-1} . They are represented by arrows pointing upwards and to the right.

Note that
 ϕ_0 corresponds
to bias, which is set
to 1

Φ is an $N \times M$ matrix

Thus Φ^T is an $M \times N$ matrix

Thus, $\Phi^T \Phi$ is $M \times M$, and so is $[\Phi^T \Phi]^{-1}$

So we have $[\Phi^T \Phi]^{-1} \times \Phi^T$ is $M \times N$

Since t is $N \times 1$, we have that $w_{ML} = [\Phi^T \Phi]^{-1} \times \Phi^T t$ is $M \times 1$.
which consists of the M weights (including bias).

What is the role of Bias parameter w_0 ?

- Sum-of squares error function is:

– Substituting:

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

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

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

$$w_0 = \bar{t} - \sum_{j=1}^{M-1} w_j \bar{\phi}_j$$

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

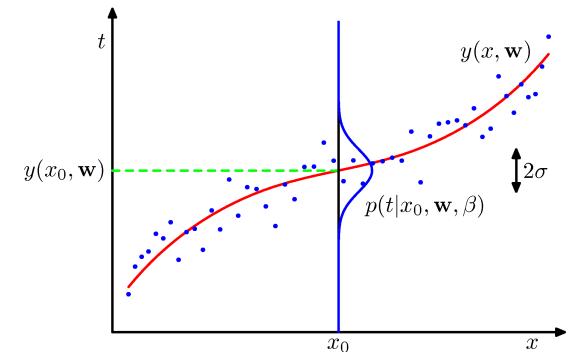
- We have determined m.l.e. solution for w using a probabilistic formulation
 - $p(t|x,w,\beta) = N(t|y(x,w), \beta^{-1})$
 - With log-likelihood

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

$$\nabla \ln p(t | w, \beta) = \beta \sum_{n=1}^N \left\{ t_n - w^T \phi(x_n) \right\} \phi(x_n)^T$$

- Taking gradient wrt β gives

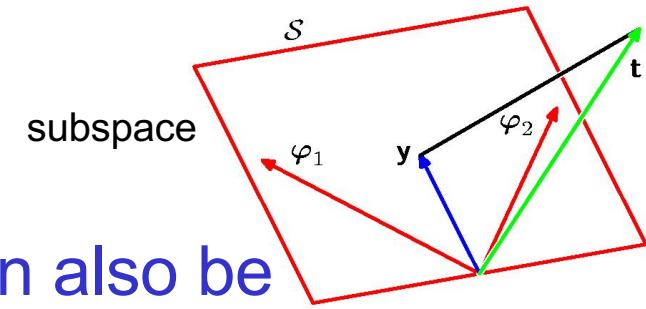
$$\frac{1}{\beta_{ML}} = \frac{1}{N} \sum_{n=1}^N \left\{ t_n - w_{ML}^T \phi(x_n) \right\}^2$$



– 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 $t = (t_1, \dots, t_N)^T$ is a vector in this space
- Each basis $\phi_j(x_n)$ evaluated at N points can also be represented as a vector in the same space
- φ_j corresponds to j^{th} column of Φ , whereas $\phi(x_n)$ corresponds to the n^{th} row of Φ
- If the no of basis functions is smaller than the no of data points
—i.e., $M < N$ then the M vectors $\varphi_j(x_n)$ will span linear subspace S of dim M
- Define y to be an N -dim vector whose n^{th} element is $y(x_n, w)$
- Sum-of-squares error is equal to squared Euclidean distance between y and t
- Solution w corresponds to 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

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

$$\Phi^+ = (\Phi^T \Phi)^{-1} \Phi^T$$

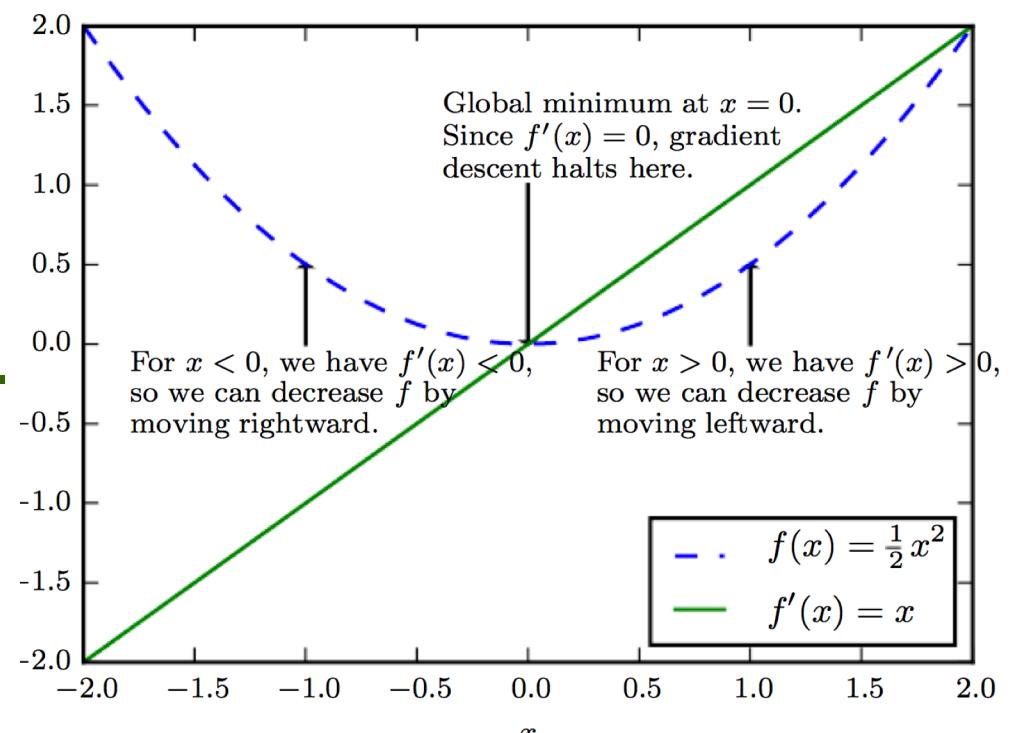
- This direct solution can lead to numerical difficulties
 - When $\Phi^T \Phi$ is close to singular (determinant=0)
 - When two basis functions are collinear parameters can have large magnitudes
- Not uncommon with real data sets
- Can be addressed using
 - Singular Value Decomposition
 - Addition of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

Method of Gradient Descent

- Criterion $f(x)$ minimized by moving from current solution in direction of negative of gradient $f'(x)$
- Steepest descent proposes a new point

$$x' = x - \eta f'(x)$$

- where η is the learning rate, a positive scalar.
- Set to a small constant.



Gradient with multiple inputs

- For multiple inputs we need partial derivatives:

$$\boxed{\frac{\partial}{\partial x_i} f(x)}$$

is how f changes as only x_i increases

–Gradient of f is a vector of partial derivatives

$$\boxed{\nabla_x f(x)}$$

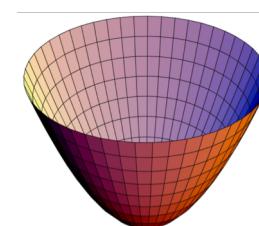
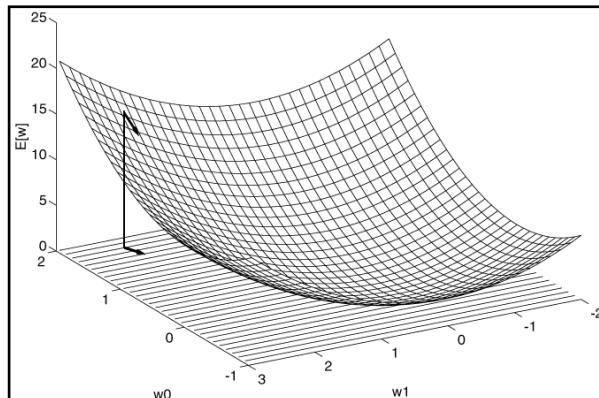
- Gradient descent proposes a new point

$$\boxed{x' = x - \eta \nabla_x f(x)}$$

–where η is the learning rate, a positive scalar

- Set to a small constant

Direction in
 w_0-w_1 plane
producing
steepest
descent



Stochastic Gradient Descent

- Error function $E_D(w) = \frac{1}{2} \sum_{n=1}^N \left\{ t_n - w^T \phi(x_n) \right\}^2$ sums over data
 - Denoting $E_D(w) = \sum_n E_n$, SGD updates w using

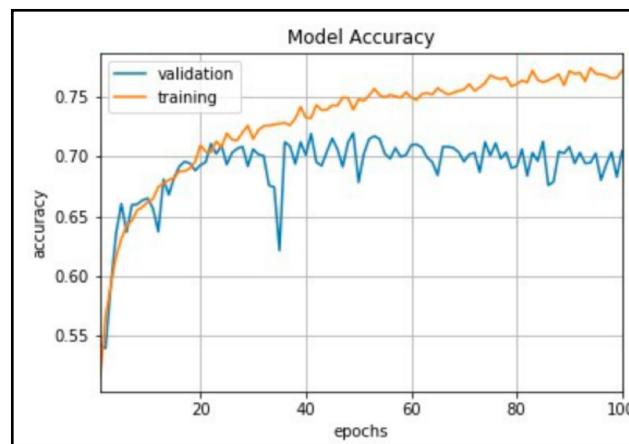
$$w^{(\tau+1)} = w^{(\tau)} - \eta \nabla E_n$$
 - where τ is the iteration no., η is a learning rate parameter and we are updating after presenting pattern n
- Substituting for the derivative $\nabla E_n = -\sum_{n=1}^N \left\{ t_n - w^T \phi(x_n) \right\} \phi(x_n)^T$

$$w^{(\tau+1)} = w^{(\tau)} + \eta(t_n - w^{(\tau)T} \phi_n) \phi_n$$
 - w is initialized to some starting vector $w^{(0)}$
 - η chosen with care to ensure convergence
 - Known as *Least Mean Squares* Algorithm

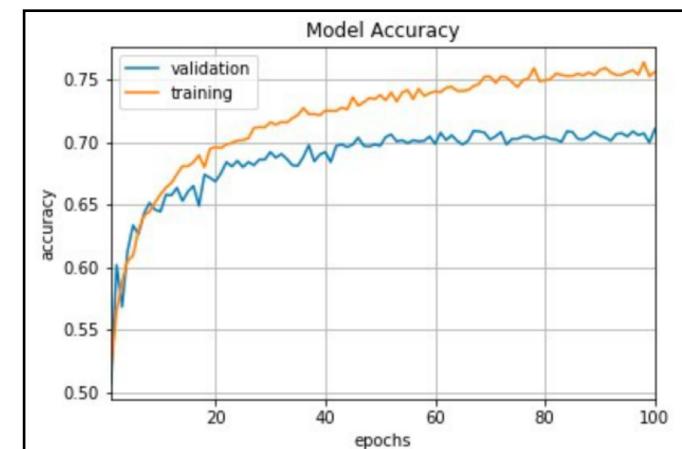
Choosing the Learning rate

- Useful to reduce η as training progresses
- Constant learning rate is default in Keras
 - Momentum and decay are set to 0 by default
 - `keras.optimizers.SGD(lr=0.1, momentum=0.0, decay=0.0, nesterov=False)`

Constant learning rate



Time-based decay: `decay_rate=learning_rate/epochs`
`SGD(lr=0.1, momentum=0.8, decay=decay_rate,`
`Nesterov=False)`



Sequential (On-line) Learning

- Disadvantage of ML solution, which is

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

- 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*

Computational bottleneck

- A recurring problem in machine learning:
 - large training sets are necessary for good generalization
 - but large training sets are also computationally expensive
- SGD is an extension of gradient descent that offers a solution
 - Moreover it is a method of generalization beyond the training set

Insight of SGD

- Gradient is an expectation
 - Expectation may be approximated using small set of samples
- In each step of SGD we can sample a *minibatch* of examples $B = \{x^{(1)}, \dots, x^{(m')}\}$
 - drawn uniformly from the training set
 - Minibatch size m' is typically chosen to be small: 1 to a hundred
 - Crucially m' is held fixed even if sample set is in billions
 - We may fit a training set with billions of examples using updates computed on only a hundred examples

$$\nabla \ln p(y | X, \theta, \beta) = \beta \sum_{i=1}^m \left\{ y^{(i)} - \theta^T x^{(i)} \right\} x^{(i)T}$$

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

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

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

Simplest Regularizer is weight decay

- Regularized least squares is

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

- Simple form of regularization term is

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

- Thus total error function becomes

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

- 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

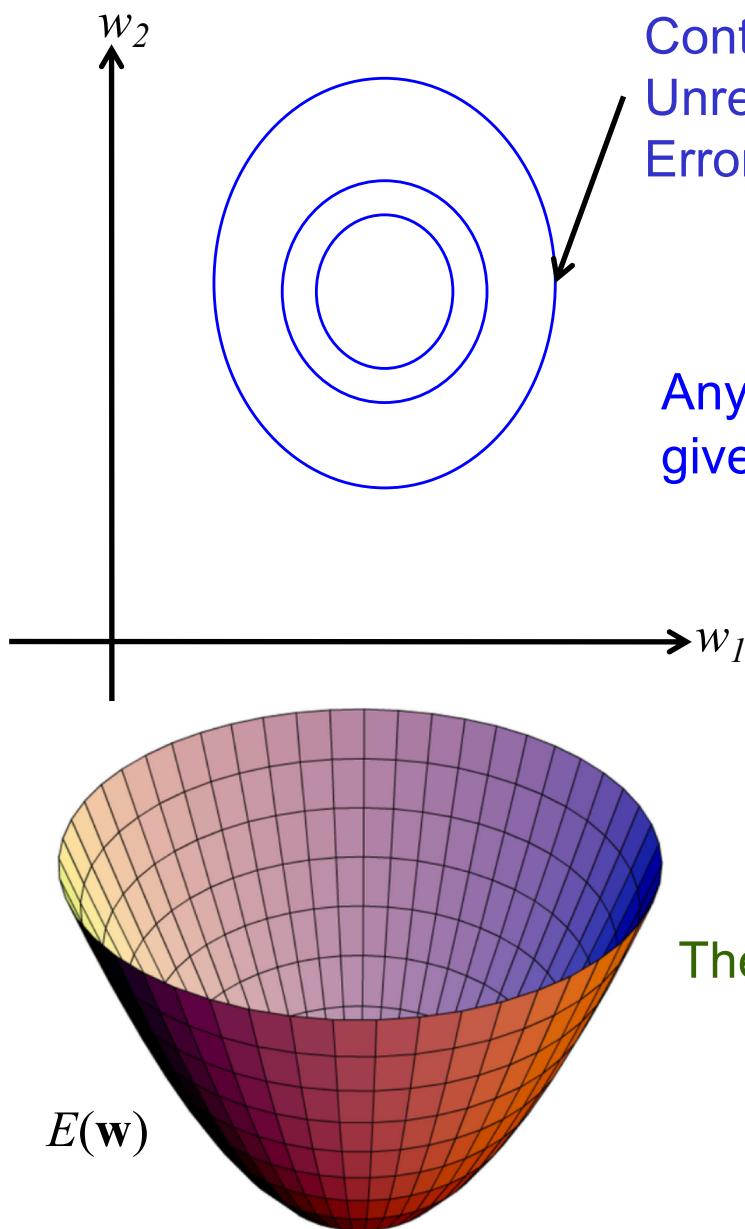
- Error function with *quadratic regularizer* is,

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

- Its exact minimizer can be found in closed form
 - By setting gradient wrt w to zero and solving for w
- This is a simple extension of the least squared solution

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

Geometric Interpretation of Regularizer



Contours of
Unregularized
Error function

Any value of w on contour
gives same error

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

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

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

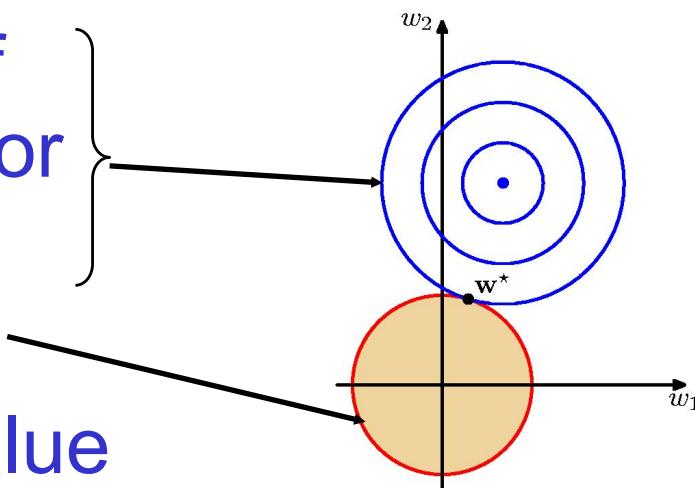
$$\sum_{j=1}^M |w_j|^2 \leq \eta$$

We don't want the weights to become too large
The two approaches related by Lagrange multipliers

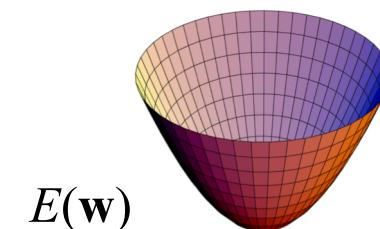
$$E(w) = \frac{1}{2} \sum_{n=1}^N \left\{ t_n - w^T \phi(x_n) \right\}^2 + \frac{\lambda}{2} w^T w$$

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

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

- 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

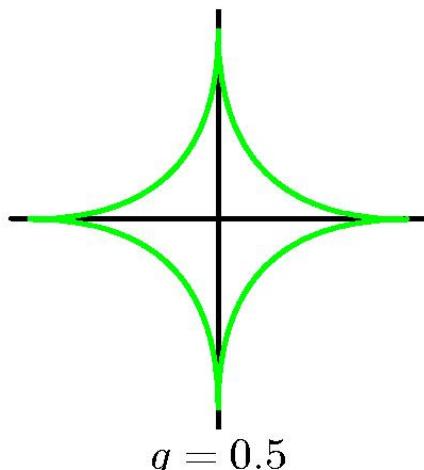
$$\frac{1}{2} \sum_{n=1}^N \left\{ t_n - w^T \phi(x_n) \right\}^2 + \frac{\lambda}{2} \sum_{j=1}^M |w_j|^q$$

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

Space of w_1, w_2

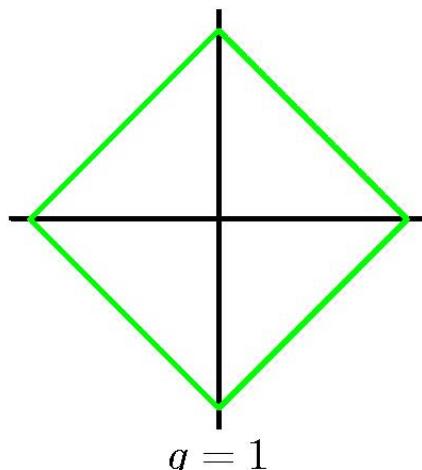
Any choice along the contour has the same value of w

$$\sqrt{w_1} + \sqrt{w_2} = const$$



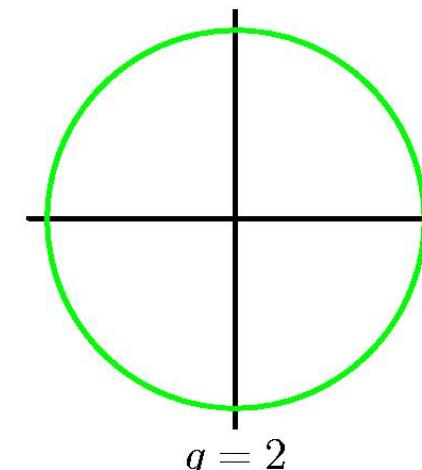
$$q = 0.5$$

$$w_1 + w_2 = const$$



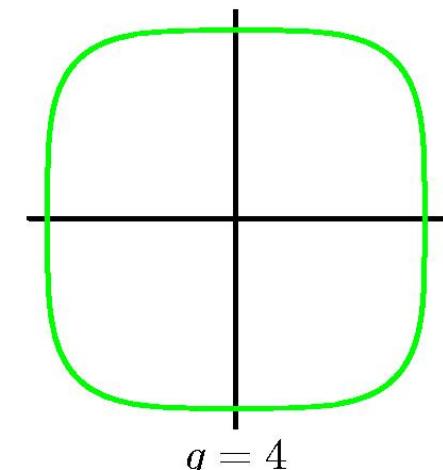
$$q = 1$$

$$w_1^2 + w_2^2 = const$$



$$q = 2$$

$$w_1^4 + w_2^4 = const$$



$$q = 4$$

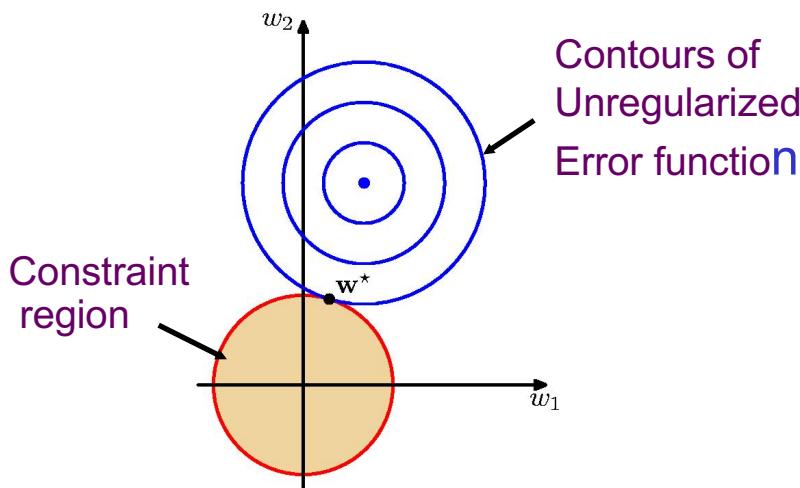
Lasso

Quadratic

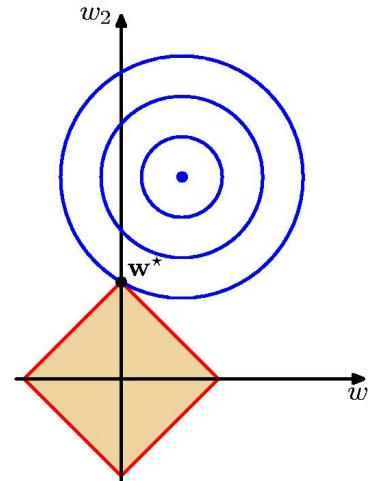
Sparsity with Lasso constraint

- 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:

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

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

- Objective Function *without/with* regularization is

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

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

- Closed-form ML solution is:

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

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

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

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

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

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