Linear Models for Regression

Stylianos Sygletos

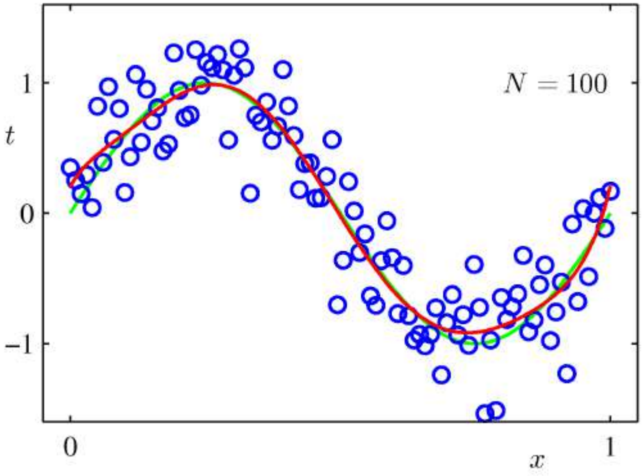
Aston Institute of Photonic Technologies - Aston University

June 10, 2021

The regression task

- It is a supervised learning task, with a goal to:
  - to predict the value of one continuous target *target* variables t
  - given a D-dimensional vector $\boldsymbol{x}$ of *input* variables.
  - and based on a dataset of known inputs and outputs (training set), i.e. $\{\boldsymbol{x}_n, t_n\}, n = 1, \ldots, N$
- From a probabilistic prespective, we aim to model the predictive distribution $p(t|\boldsymbol{x})$.
  - it expresses our uncertainty about the value of t for each value of $\boldsymbol{x}$
  - we can make predictions of t, for any new value of $\boldsymbol{x}$
  - we are not trying to model the distribution of $\boldsymbol{x}$
- We do not expect the predictor to be a linear function of $\boldsymbol{x}$.

An Example



Polynomial Curve Fitting with a Scalar

- With a *single* input variable

$$y(x, \boldsymbol{x}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

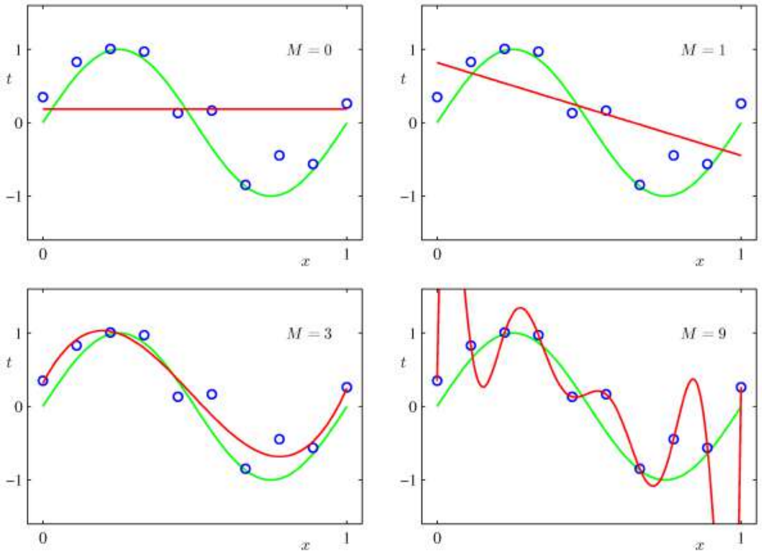
  where M the polynomial order, x^j the j^{th} order of x, and $\boldsymbol{w} = (w_0, \ldots, w_M)^T$

- *What is our task?:* To learn $\boldsymbol{w}$ from training data $\{(x_i, t_i)\}_{i=1}^{N}$. We calculate the misfit between predictions $y(x_n, \boldsymbol{w})$ for each data point x_n and corresponding target values t_n:

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

  it is zero when $y(x, \boldsymbol{w})$ passes through training data points.

Results with Polynomial Basis



Regression with multiple inputs

- Generalization :
  - predict value of continuous target variable t given value of D input variables $\boldsymbol{x} = [x_1, ..., x_D]$
  - t can also be a set of variables (multiple regression)
  - Linear functions of adjustable parameters
    - specifically linear combinations of *nonlinear* functions of input variables
- Polynomial curve fitting is good only for :
  - single input scalar variable x
  - it cannot be easily generalized to several variables

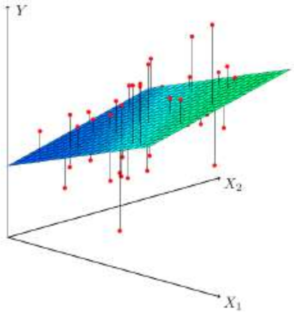
Simplest Linear Model with D inputs

- It involves a linear combination of D-input variables:

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

  where $\boldsymbol{x} = (x_1, \ldots, x_D)^T$

- Known as *linear regression* because it is a linear function of the parameters $w_0, \ldots, w_D$.



- Being a linear function of input variables imposes severe limitations
- We can extend the class of models by considering fixed nonlinear functions of input variables

Basis Functions

- If the original variables comprise the vector $\boldsymbol{x}$, then the features can be expressed in terms of basis functions $\phi_j(\boldsymbol{x})$:

$$y(\boldsymbol{x}, \boldsymbol{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\boldsymbol{x}) = \sum_{j=0}^{M-1} w_j \phi_j(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x})$$

  where $\boldsymbol{w} = (w_0, \ldots, w_{M-1})^T$ and $\boldsymbol{\phi} = (\phi_0, \ldots, \phi_{M-1})$ with $\phi_0(\boldsymbol{x}) = 1$ defining a *dummy basis function* for the bias w_0

- By using nonlinear basis function we allow $y(\boldsymbol{x}, \boldsymbol{w})$ to be a nonlinear function of the input vector $\boldsymbol{x}$

Choice of Basis Function

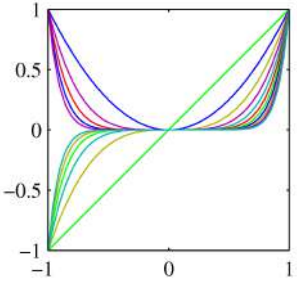
1. Polynomial
2. Gaussian
3. Sigmoidal
4. Fourier
5. Wavelets

Polynomial Basis for one variable

- Linear Basis Function Model:

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

- Polynomial Basis (for single cariable x)
  $\phi_j(x) = x^j$



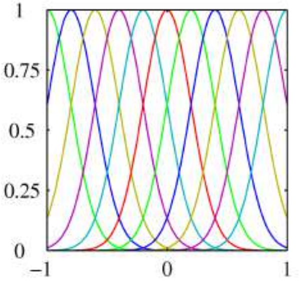
Disadvantages of polynomial basis :

- Global: changes in one region of input space affects others
- Difficult to formulate : number of polynomials increases exponentialy with M
- Can divide input space into regions :
  - use different polynomial in each region
  - use local basis functions (e.g. *spline functions*)

Gaussian Radial Basis Functions

- Gaussian:

$$\phi_j(x) = \exp\left[\frac{(x-\mu_j)^2}{2\sigma^2}\right]$$



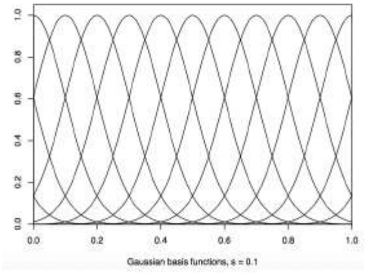
  - does not have probabilistic interpretation
  - normalization term not needed

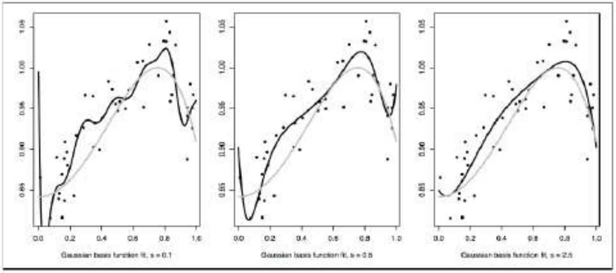
- Choice of parameters :
  - μ_j govern the locations of the basis functions. Can be an arbitrary set of points within the range of data
  - σ governs the spatial scale. Could be chosen from the data set, e.g. average variance

Example with Gaussian Radial Basis Functions

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

basis function for $s = 0.1$ with the μ_j on a grid with spacing s





w_j s for middle model:

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

Sigmoidal Basis Functions

- Basis function of the form:

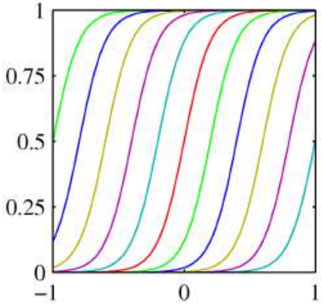
$$\phi_j(x) = \sigma\left(\frac{x - \mu_j}{s}\right)$$

where

$$\sigma(\alpha) = \frac{1}{1 + \exp(-\alpha)}$$

- Equivalently we can use $\tanh$:

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



Logistic Sigmoid for different μ_j

Other Basis Functions

- Fourier
  - Expansion in sinusoidal functions
  - Infinite spatial extent
- Wavelets
  – Localized functions in space and time
  – They have many applications in signal processing

Relationship between Maximum Likelihood and Least Squares

Goal: To show that minimizing the sum-of-squared error function is equivalent to miximizing the likelihood function under the Gaussian noise model

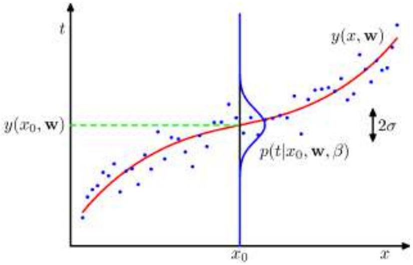
- Assume scalar target variable t given by:

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

  - $y(\boldsymbol{x}, \boldsymbol{w})$ deterministic function
  - ϵ zero mean Gaussian noise

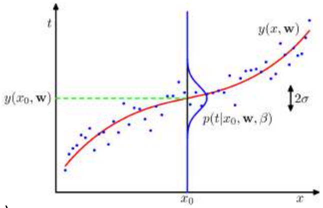
- Distribution of t is univariate normal:

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



Likehood Function

- Dataset:
  – input values $\boldsymbol{X} = \{x_1, \ldots, x_N\}$
  – target values $\mathbf{t} = \{t_1, \ldots, t_N\}$



- *Likelihood*:

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

It is the probability of observing the target data assuming they are independent

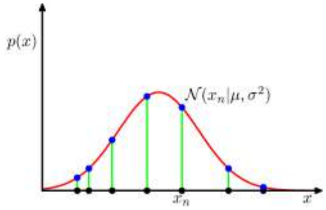
- *Log-Likelihood*: It is computationally more convenient

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

Likehood Function

For the univariate Gaussian we have:

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



where the sum-of-squares error function is defined by:

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

Therefore, maximizing the likelihood is equivalent to minimizing the error function $E_D(\boldsymbol{w})$!

Maximum Likelihood Solution

We take the gradient of the log-likelihood (or of the $E_D(\boldsymbol{w})$):

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

and set it equal to zero:

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

and solving for $\boldsymbol{w}$ we obtain:

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

That is, we can get a closed form solution(!), known as the *normal equations* for the least squares problem

The Density Matrix Φ

- $\mathbf{\Phi}$ is an $N \times M$ matrix:

$$\mathbf{\Phi} = \underbrace{\begin{bmatrix} \phi_0(\boldsymbol{x}_1) & \phi_1(\boldsymbol{x}_1) & \dots & \phi_{M-1}(\boldsymbol{x}_1) \\ \phi_0(\boldsymbol{x}_2) & \phi_1(\boldsymbol{x}_2) & \dots & \phi_{M-1}(\boldsymbol{x}_2) \\ \vdots & & & \\ \phi_0(\boldsymbol{x}_N) & \phi_1(\boldsymbol{x}_N) & \dots & \phi_{M-1}(\boldsymbol{x}_N) \end{bmatrix}}_{M\text{-basis functions}}$$

  whose elements $\phi_j(\boldsymbol{x}_n)$ are M-basis functions, e.g. j^{th} order polynomial, or Gaussians centered on M-data points

- *Moore-Penrose pseudo-inverse* of matrix $\mathbf{\Phi}$

$$\mathbf{\Phi}^{\dagger} \equiv (\mathbf{\Phi}^T \mathbf{\Phi})^{-1} \mathbf{\Phi}^T$$

Maximum Likelihood for precision β

Again, starting from the log-likelihood function:

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

and maximizing with respect to the noise precision parameter β we get:

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

Thus, the inverse of the noise precision is given by the residual variance of the target values around the regression function.

Difficulty of Direct Solution

- Direct solution of the normal equations:

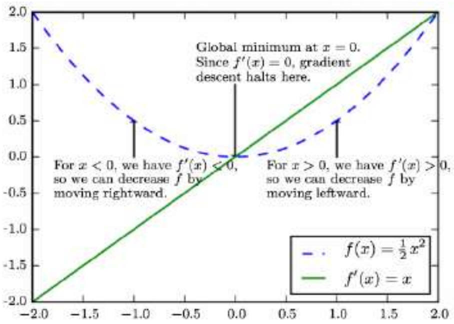
$$\boldsymbol{w}_{ML} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\mathbf{t}$$

- Direct numerical calculation of the closed-form solution is not always feasible
  - when $(\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}$ is (close) to singular (i.e. $\det(\boldsymbol{\Phi}) \simeq 0$)
  - when two basis functions are colinear, the resulting parameters values can have large magnitudes

- This is not uncommon with real data sets

- It can be addressed using:
  - Singular Value Decomposition
  - Addition of regularization term can ensure non-singular matrix
  - Applying alternative numerical methods (e.g. gradient-descent)

Method of Gradient Descent



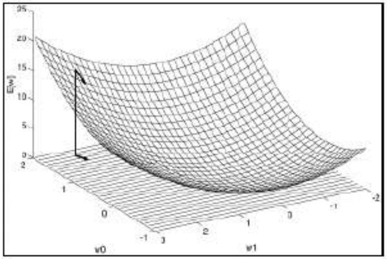
- Criterion $f(x)$ is minimized by moving from current solution in the direction of negative of gradient $f'(x)$
- Steepest descent proposes a new point

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

where $\eta > 0$ is the *learning rate*, scalar set to a small constant

Gradient with multiple inputs

Direction in w_0-w_1 plane producing steepest descent



- For multiple inputs we need partial derivatives.
- $\frac{\partial}{\partial x_i} f(\boldsymbol{x})$ is how f changes as only x_i increases, gradient of f is a vector of partial derivatives $\nabla_x f(\boldsymbol{x})$
- Gradient descent proposes a new point:

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

where $\eta > 0$ is the *learning rate*, scalar set to a small constant

Stochastic Gradient Descent for Regression

- Error function $E_D(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n)\}^2$ sums over data. Denoting $E_D(\boldsymbol{w}) = \sum_n E_n$, we update $\boldsymbol{w}$ using

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

  where τ is the iteration number and η the learning rate parameter item Replacing for the $\nabla E_n = -\{t_n - \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n)\}\boldsymbol{\phi}(\boldsymbol{x}_n)^T$ we get:

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

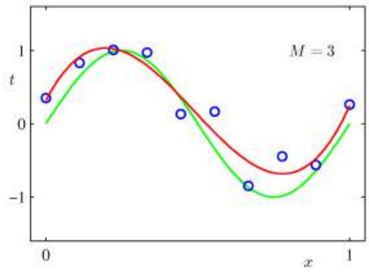
  where

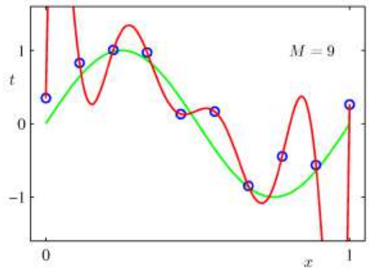
  - $\boldsymbol{w}$ is initialized to some starting vector $\boldsymbol{w}^0$
  - η chosen with care to ensure convergence
- Known as Least Mean Squares Algorithm

Sequential (on-line) Learning

- Maximum likelihood solution is : $\boldsymbol{w}_{ML} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\mathbf{t}$
- It is a batch technique
  - Processing entire training set in one go
  - Computationally expensive for large data sets, due to huge $N \times M$ design matrix $\boldsymbol{\Phi}$
- Solution: use a sequential algorithm where samples are presented one at a time (or a minibatch at a time) – called *stochastic gradient descent - SGD*
- Computational bottleneck: is a recurring problem in ML:
  - large training sets are necessary for good generalization
  - but large training sets are also computationally expensive
- SGD is an extension of gradient descent offering a solution
  - moreover it is a method of generalization beyond the training set

Regularized Least Squares





- As model complexity increases, e.g. degree of polynomial or number of basis function then we are likely to have overfitting

- We can control overfitting by adding a regularization term to the error function

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

where λ is the *regularization coefficient*

Simplest regularization form (*weight-decay*)

- The simplest regularization form is the sum-of-squares of the weight vector elements:

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

then, the total error function becomes:

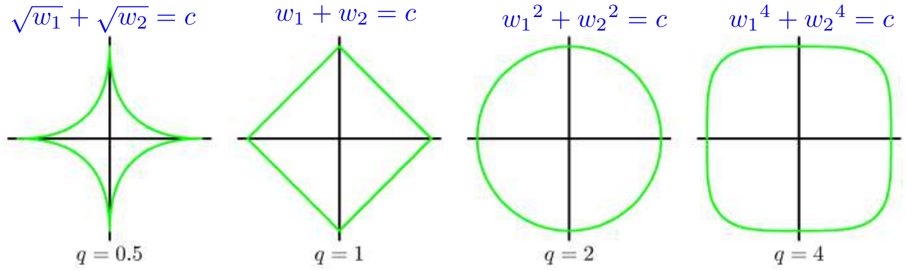
$$\frac{1}{2}\sum_{n=1}^{N}\left\{t_n - \boldsymbol{w}^T\phi(\boldsymbol{x}_n)\right\}^2 + \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w}$$

- The error function remains a *quadratic function* of $\boldsymbol{w}$, so its exact minimizer can be found in analytical form:

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

which is a simple extension of the leas squared solution $\boldsymbol{w} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\mathbf{t}$

A more general regularizer



- In a more general form the regularized error becomes:

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

  - $q = 2$ corresponds to the *quadratic* equalizer
  - $q = 1$ is known as *Lasso*

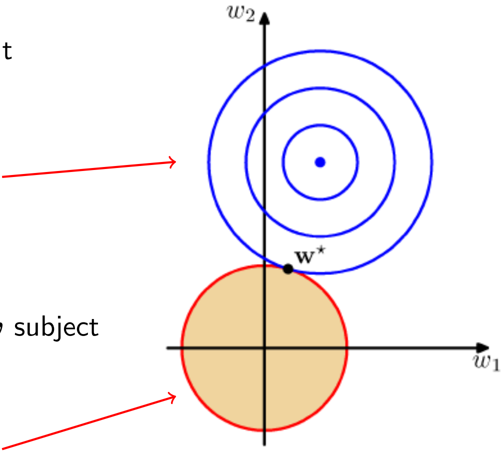
Geometric interpretation of regularizer

- **Unregularized case**
  We are tying to find $\boldsymbol{w}$ that minimizes:

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

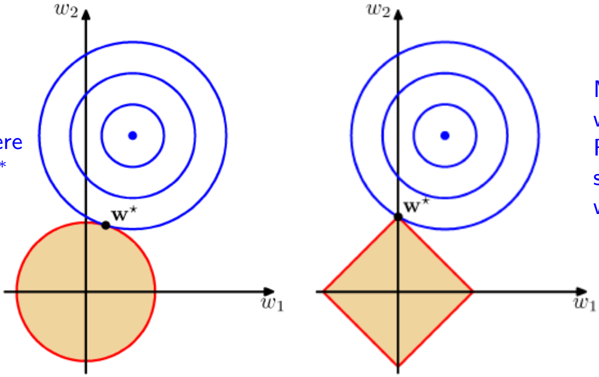
- **Regularized case**
  We choose that value of $\boldsymbol{w}$ subject to the constraint

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



Sparsity with Lasso constraint

Quadratic solution where $w_1{}^*$ and $w_2{}^*$ are nonzero



Minimization with Lasso Regularizer. A sparse solution with $w_1{}^* = 0$

- With $q = 1$ and λ is sufficiently large, some of the coefficients w_j are driven to zero.
- This leads to a sparse model where the corresponding basis functions play no role.

Regularization: Summary

- Regularization allows complex models to be trained on limited size datasets without severe overfitting
- The problem of detemining the optimal model complexity is shifted to determining a suitable λ value

Multiple Outputs: $\boldsymbol{t} = (t_1, \ldots, t_K),\ K > 1$

- It can be treated as multiple independent regression problems, introducing a different set of basis functions for each component of $\boldsymbol{t}$
- A more common approach is to introduce the same set of basis functions to model the target vector $\boldsymbol{t}$ components:

$$\boldsymbol{y}(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{W}^T \boldsymbol{\phi}(\boldsymbol{x})$$

  where $\boldsymbol{y}$ is a K-dimensional column vector; $\boldsymbol{W}$ is an $M \times K$ matrix of parameters; $\boldsymbol{\phi}(\boldsymbol{x})$ is an M-dimensional column vector with elements $\phi_j(\boldsymbol{x})$, and $\phi_0(\boldsymbol{x}) = 1$

Solution for Multiple Outputs

- We assume the target vector $\boldsymbol{t}$ has a conditional distribution of the form:

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

- Log-Likelihood

$$\begin{aligned}\ln p(\boldsymbol{T}|\boldsymbol{X}, \boldsymbol{X}, \beta) &= \sum_{n=1}^{N} \ln \mathcal{N}(\boldsymbol{t}_n|\boldsymbol{W}^T\boldsymbol{\phi}(\boldsymbol{x}_n), \beta^{-1}\boldsymbol{I}) \\ &= \frac{NK}{2}\ln\left(\frac{\beta}{2\pi}\right) - \frac{\beta}{2}\sum_{n=1}^{N}\|\boldsymbol{t}_n - \boldsymbol{W}^T\boldsymbol{\phi}(\boldsymbol{x}_n)\|^2\end{aligned}$$

where

- $\boldsymbol{T}$ is $N \times K$ matrix combining the $\boldsymbol{t}_1, \ldots, \boldsymbol{t}_N$ observation set
- $\boldsymbol{X}$ combines the input vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$

Solution for Multiple Outputs

- The solution that maximized the log-likelihood is:

$$\boldsymbol{W}_{ML} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\boldsymbol{T}$$

- For each target variable t_k, we have:

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

Conclusion:
The solution decouples between the target variables, thus we need only to compute a single pseudo-inverse matrix $\boldsymbol{\Phi}^\dagger$, shared by all of the vectors $\boldsymbol{w}_k$