50.039 – Theory and Practice of Deep Learning

Alex

Week 01: Ordinary Least Squares (Linear regression), Basis Functions

[The following notes are compiled from various sources such as textbooks, lecture materials, Web resources and are shared for academic purposes only, intended for use by students registered for a specific course. In the interest of brevity, every source is not cited. The compiler of these notes gratefully acknowledges all such sources.]

1 Recap

Recap

- Discriminative Learning problems consist usually of a least four basic components
  1. Input space $\mathcal{X}$ and output space $\mathcal{Y}$
  2. the prediction mapping connecting these $f : \mathcal{X} \to \mathcal{Y}$
  3. a loss function
  4. a method to select a prediction mapping f / or its parameters from a dataset
- the data to be expected to be seen is modeled by a probability distribution
- training and testing data are finites samples from it
- Learning $\approx$ finding a good predictive mapping f of an input space to an output space.
- that probability distribution is unknown, but the expected loss can be approximated using training and testing data sets

Learning goals

- Go through linear regression and ridge regression along the lines of above 1.-4. for the case of explicit solution without bias terms
- be able to explain how the set of constant points for $\{x : f(x) = c\}$ looks like for a linear model
- an insight into the effect of regularization under noise
- extending linear regression by basis functions (polynomial, RBF): you should be able to reproduce their formulas and the reasons for using them

2 Ordinary Least Squares (Linear regression)

Lets go through above methodology.

2.1 1. Input and output space?

Some examples:
A. `http://archive.ics.uci.edu/ml/datasets/Real+estate+valuation+data+set`

$X1$ = the transaction date

$X2$ = the house age (unit: year)

$X3$ = the distance to the nearest MRT station (unit: meter)

$X4$ = the number of convenience stores in the living circle on foot (integer)

$X5$ = the geographic coordinate, latitude. (unit: degree)

$X6$ = the geographic coordinate, longitude. (unit: degree)

- The output is as follows Y = house price per unit of area

B. `http://archive.ics.uci.edu/ml/datasets/Beijing+PM2.5+Data`

$X1$ = Dew Point

$X2$ = Temperature

$X3$ = Pressure

$X4$ = Combined wind direction

$X5$ = Cumulated wind speed

$X6$ = Cumulated hours of snow

$X7$ = Cumulated hours of rain

- The output is as follows Y = PM2.5 concentration ($\mu g/m^3$)

C. `http://archive.ics.uci.edu/ml/datasets/Communities+and+Crime` (`http://archive.ics.uci.edu/ml/datasets/Concrete+Slump+Test` `https://en.wikipedia.org/wiki/Concrete_slump_test`) (`http://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset`)

Input and output space in regression problems?

$\mathcal{X} =?, \mathcal{Y} =?$

2.2 2. the prediction mapping?

Choose linear function without/with a bias

$$f_w(x) = x \cdot w \qquad = \sum_{d=1}^{D} x_d w_d \qquad , \ w \in \mathbb{R}^{D\times 1}$$

$$g_{w,b}(x) = x \cdot w + b \qquad = \sum_{d=1}^{D} x_d w_d + b \qquad , \ w \in \mathbb{R}^{D\times 1}, b \in \mathbb{R}^1$$

2.3 What does a linear mapping represent?

Goal: understand what the mapping does.

A. What is the set of points x: $g_{w,b}(x) = 0$?

B. What is the set of points x the prediction is a constant c, that is $g_{w,b}(x) = c$?

2.4 What is the set of points x: $g_{w,b}(x) = 0$?

$$\begin{aligned} & g_{w,b}(x) && = 0 \\ \Leftrightarrow & x \cdot w = -b \end{aligned}$$

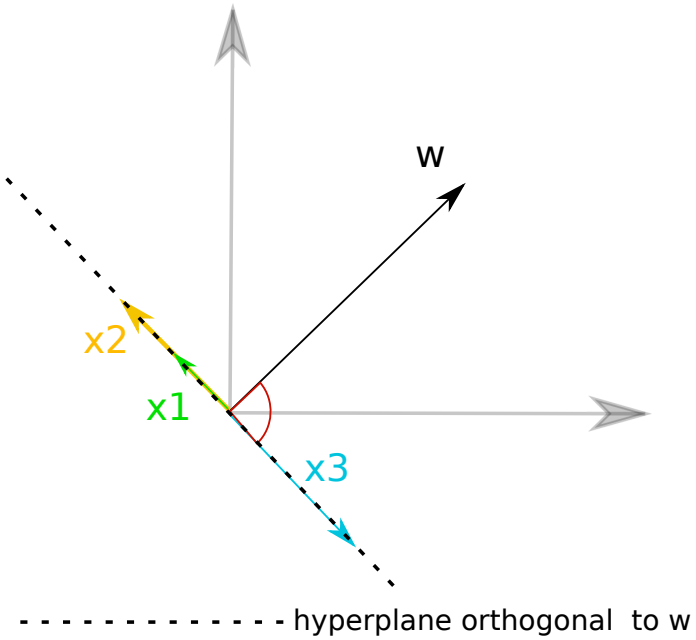
In order to understand how the bias b influences the zero set, lets consider three cases:

- $b = 0$
- $b > 0$
- $b < 0$

The case $b = 0$ We know that for $b = 0$: $g_{w,0}(x) = w \cdot x = 0$ holds for the zero vector $x = 0$.

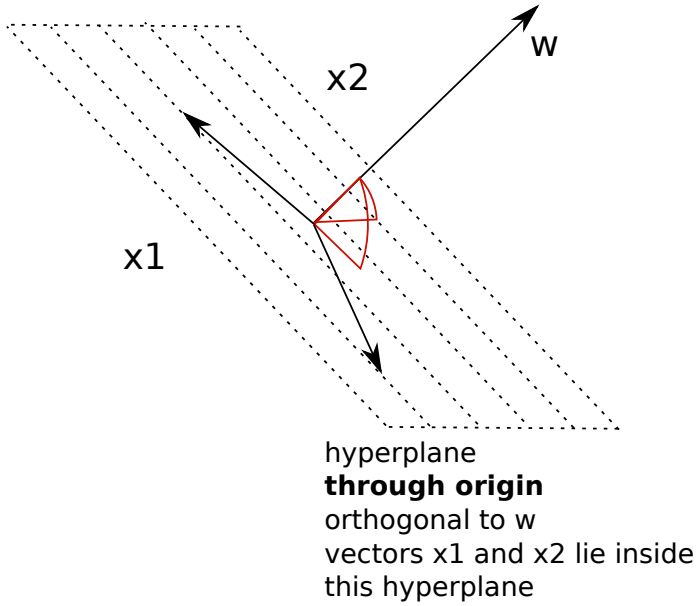
$$x \cdot w = 0$$

The set of points x such that the inner product $x \cdot w = 0$ is in 2 dims a one-dimensional line, which goes through the origin $(x_1, x_2) = (0, 0)$, and which is orthogonal to w.



x_1,x_2,x_3 are all orthogonal to the vector w

The analogy also holds for 3 or more dimensions. So for 3 dims it is a two-dimensional plane, which goes through the origin $(x_1, x_2, x_3) = (0, 0, 0)$.

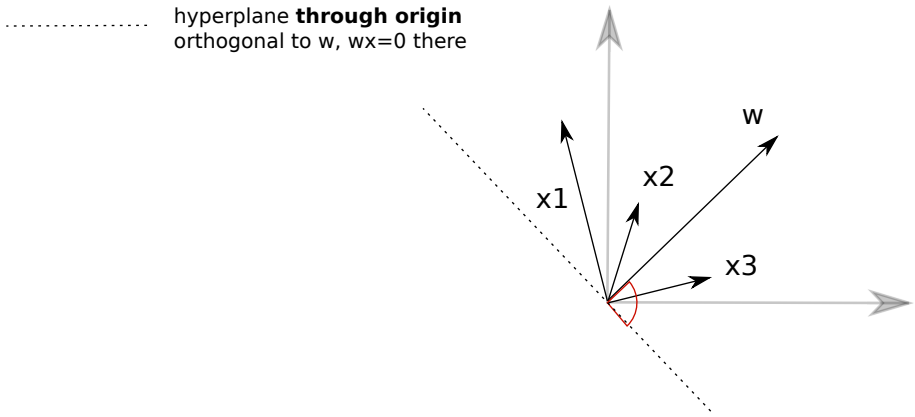


For n dimensions the plane of orthogonal vectors has $n-1$ dimensions (+ goes through the origin), but is still a hyperplane (that is if $x_1 \in P, x_2 \in P \Rightarrow a_1 x_1 + a_2 x_2 \in P$, and P can be represented as a linear combination of $n-1$ basis vectors).

The case $b < 0$

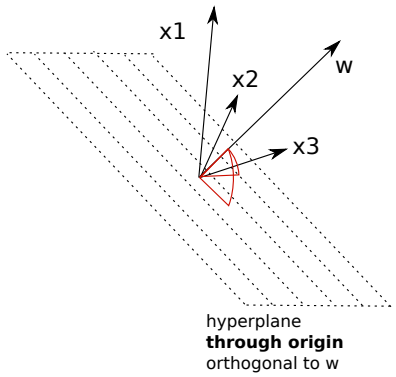
$$b < 0,\ w \cdot x + b = 0 \Rightarrow w \cdot x = -b > 0$$

We know: $w \cdot x > 0$ for all points x that are on that side of the **hyperplane through the origin**, in which w points to.



x_1,x_2,x_3 all have $w \cdot x_i > 0$ because relative to the hyperplane orthogonal to w which goes through the origin, they all point into the direction of w

The same also holds for 3 or more dimensions.



x_1,x_2,x_3 all have $w \cdot x_i > 0$ because relative to the hyperplane orthogonal to w which goes through the origin, they all point into the direction of w

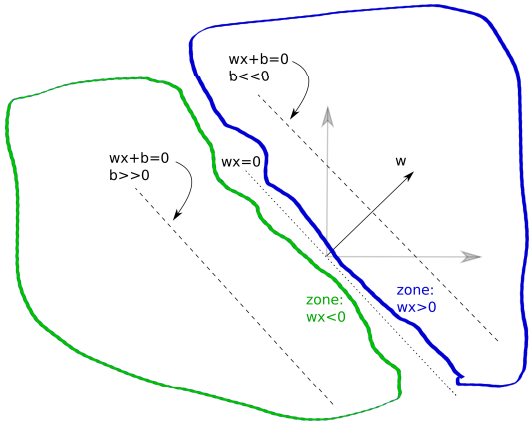
What does that mean? All the above vectors solve $wx + b = 0$ for some bias $b < 0$!

The case $b > 0$
The analogous thought results in the understanding, that set of points x such that

$$\{x : \ wx + b = 0\}$$

is a hyperplane which is parallel to the hyperplane $\{x : x \cdot u = 0\}$ orthogonal to w going through the origin, and which is shifted opposite to the direction of w

As the next graphic shows, the bias b shifts the zero set corresponding to $w \cdot x + b = 0$ anti-parallel to the direction of w.



hyperplane dependency on bias b

- Negative $b < 0$ shift the hyperplane $\{x : wx + b = 0\}$ into the direction of w,
- positive $b > 0$ shift the hyperplane $\{x : wx + b = 0\}$ against the direction of w.
- Large values of $|b|$ shift it away quite far.

hyperplane explicit

As a summary: the linear mapping $g(x) = w \cdot x + b$ has a zero set which is the plane of points

$$\{x : x = u + -b\frac{w}{\|w\|^2}, u \text{ such that } w \cdot u = 0\}$$

In this representation:

- u such that $w \cdot u = 0$ is the hyperplane of vectors u orthogonal to w.
- the vector $-b\frac{w}{\|w\|^2}$ shifts the hyperplane antiparallel to direction of w.

That holds because

$$\begin{aligned} w \cdot x + b &= w \cdot (u + -b\frac{w}{\|w\|^2}) + b \\ &= w \cdot u + w \cdot (-b\frac{w}{\|w\|^2}) + b \\ &= 0 + -b\frac{w \cdot w}{\|w\|^2} + b \\ &= 0 + -b\frac{\|w\|^2}{\|w\|^2} + b = 0 \end{aligned}$$

2.5 What is the set of points x where the prediction is a constant, that is $g_{w,b}(x) = c$?

This can be answered by reducing it to a zero set:

$$\begin{aligned} g_{w,b}(x) = wx + b &= c \\ wx + (b - c) &= 0 \end{aligned}$$

So the set of points x such that $g_{w,b}(x) = c$ is just the zero-set of $g_{w,b-c}(x)$.

2.6 3. Loss function for regression

Loss function for a pair (x, y):

$$L(f(x), y) = (f(x) - y)^2$$

Reasons:

- capture deviations on both sides of the real ground truth value y
- simple derivative

2.7 4. a method to select a prediction mapping $g_{w,b}$ / or its parameters from a dataset

What is the problem that we want to solve? First of all, we assume here no bias for this lecture (to get an explicit solution)
parameters are w

We make the assumption that our training dataset is representative of the distribution of the data that we expect to see in the future. The training dataset consists of (x_i, y_i) that is input samples x_i and their regression labels y_i. We want to find parameters which minimize the loss on this dataset, that is:

$$(w^*) = \text{argmin}_w \sum_{i=1}^{n} L(f(x_i), y_i) = \text{argmin}_w \sum_{i=1}^{n} (x_i \cdot w - y_i)^2$$

for a given dataset $D_n = \{(x_i, y_i)\}$. Then $f_{w^*}(x) = x \cdot w^*$ is the selected mapping.

Rare case: Can be solved explicitly for w. Write in matrix form:

$$X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} x_1^{(1)}, \ldots, x_1^{(D)} \\ x_2^{(1)}, \ldots, x_2^{(D)} \\ \vdots \\ x_n^{(1)}, \ldots, x_n^{(D)} \end{pmatrix} \in \mathbb{R}^{n \times D}$$

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

then:

$$\sum_{i=1}^{n} (x_i \cdot w - y_i)^2 = (X \cdot w - Y)^T \cdot (X \cdot w - Y)$$

Solve the minimization problem by computing the gradient for w and setting it to zero.

$$\begin{aligned} D_w((X \cdot w - Y)^T \cdot (X \cdot w - Y)) = 2X^T \cdot (X \cdot w - Y) &= 0 \\ 2X^T \cdot (X \cdot w - Y) &= 0 \\ (X^T \cdot X) \cdot w &= X^T \cdot Y \\ w &= (X^T \cdot X)^{-1} X^T \cdot Y \end{aligned}$$

if the matrix inverse $(X^T \cdot X)^{-1}$ exists.

solution without bias

Let X be the training data matrix.

$$X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} x_1^{(1)}, \ldots, x_1^{(D)} \\ x_2^{(1)}, \ldots, x_2^{(D)} \\ \vdots \\ x_n^{(1)}, \ldots, x_n^{(D)} \end{pmatrix} \in \mathbb{R}^{n \times D}$$

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \in \mathbb{R}^{n \times 1}$$

The model is $f(x) = w \cdot x$. Then a solution is given as

$$w = (X^T \cdot X)^{-1} X^T \cdot Y$$

if the matrix inverse $(X^T \cdot X)^{-1}$ exists.

How to extend this to a bias?

Extend each feature by an additional dimension set to 1:

$$x = \left(x^{(1)}, \ldots, x^{(D)}\right) \to \hat{x} = \left(x^{(1)}, \ldots, x^{(D)}, 1\right)$$

Then the parameter w also gets an additional dimension

$$\hat{w} = \left(w^{(1)}, \ldots, w^{(D)}, w^{(D+1)}\right)$$

Demo: `t1()` shows a solution. You have `t1()` also in your code for playing with it!

3 From Linear to Ridge regression

Overfitting: low training error, high test error. Reason: during learning one picks up too much of the noise in the data, and learns weights that listen to noise signals.
One way to deal with it: avoiding weights w getting too large. How to do that ? Add a penalty on the euclidean length of w

$$\text{argmin}_w \sum_{i=1}^{n} (x_i \cdot w - y_i)^2 + \lambda \|w\|^2$$

Ridge regression

Linear regression with added penalty term on weights

$$\lambda \|w\|^2$$

λ is a hyperparameter in this approach. Solution changes to

$$w = (X^T \cdot X + \lambda I)^{-1} X^T \cdot Y$$

Overfitting: looking at the data too closely. Regularization – avoid to do that too fine-grained look.
in matrix form

$$\sum_{i=1}^{n} (x_i \cdot w - y_i)^2 + \lambda \|w\|^2 = (X \cdot w - Y)^T \cdot (X \cdot w - Y) + \lambda w \cdot w$$

$$D_w((X \cdot w - Y)^T \cdot (X \cdot w - Y) + \lambda w \cdot w) = 2X^T \cdot (X \cdot w - Y) + 2\lambda w = 0$$

$$\begin{aligned} X^T \cdot (X \cdot w - Y) + \lambda w &= 0 \\ (X^T \cdot X + \lambda I) \cdot w &= X^T \cdot Y \\ w &= (X^T \cdot X + \lambda I)^{-1} X^T \cdot Y \end{aligned}$$

4 The effect of Ridge Regression on the solution and eigenvalues in particular

One can see: for any $\lambda > 0$ the matrix $A = (X^T \cdot X + \lambda I)$ has only positive eigenvalues, so it is always invertible.
A matrix A has only positive eigenvalues if for all $v \neq 0$ we have $v^T A v > 0$. This can be shown. Assume $v \neq 0, \lambda > 0$, then

$$\begin{aligned} v^T (X^T \cdot X + \lambda I) v &= v^T X^T \cdot X v + v^T \lambda I v \\ &= (Xv)^T Xv + \lambda v^T v = \underbrace{\|Xv\|^2}_{\geq 0} + \lambda \|v\|^2 \\ &\geq \lambda \|v\|^2 > 0 \end{aligned}$$

Moreover: adding λI to $X^T \cdot X$ puts a lower bound on eigenvalues of $(X^T \cdot X + \lambda I)$, so for the inverse the eigenvalues become upper-bounded. "better-conditioned inverse".

Demo: `t5()` shows train and test error for varying values of $\lambda = \exp(\gamma), \gamma \in \{-50, -40, -30, -20, -15, -10, -1, 0, 1, 10\}$ and $F = 9$ – this value of F overfitted in the last example notable (because $n = 10 = F+1$), but with regularization it becomes manageable. Regularization allows to use a high dimensional feature space, and still find a reasonable solution

5 Why does regularization help?

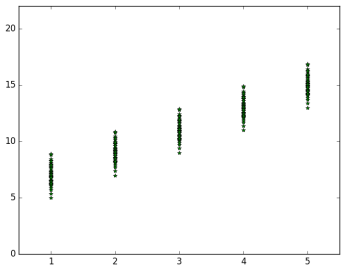
not always, depends on its strength:

A high regularization results in poor performance if the prediction problem has little noise.
A high Regularization results in good performance if the prediction problem has high noise.
The following explanation is a bit intuitive.

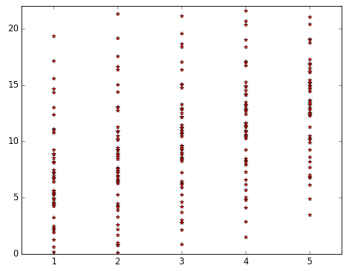
- High noise implies that the training data varies a lot, every time when one would obtain a training data set - due to the noise.

  To see this consider the following models for data: the true relationship $y = 2x + 5$ to be learned corrupted by gaussian noise ϵ

  $y = 2x + \epsilon, x = 1, 2, 3, 4, 5$



$\epsilon \sim N(0, 1)$



$\epsilon \sim N(0, 5)$

- w is learnt from the training data.

Lets plug in $Y = 2X + noisematrix$ into our solution for w:

$$\begin{aligned} w &= (X^T \cdot X + \lambda I)^{-1} X^T \cdot Y \\ &= (X^T \cdot X + \lambda I)^{-1} X^T \cdot (2X + noisematrix) \\ &= \underbrace{2(X^T \cdot X + \lambda I)^{-1} X^T \cdot X}_{\text{learns true function, for } \lambda \approx 0 \text{ it is } \approx 2(X^T \cdot X)^{-1} X^T \cdot X = 2} \\ &+ \underbrace{(X^T \cdot X + \lambda I)^{-1} X^T \cdot noisematrix}_{\text{impact of noise on } w} \end{aligned}$$

So w has two components, one that learns the true function, and one that comes from the noise.

Linear algebra: A large λ makes the matrix $(X^T \cdot X + \lambda I)^{-1}$ have small eigenvalues, therefore $(X^T \cdot X + \lambda I)^{-1} v$ maps all vectors v closer to zero.

As a consequence: A large λ reduces the impact of noise (reduced variance), but also distorts the learning of the true function (increased bias). Bias here is not about the bias term b of the model, but it is about the systematic mismatch between the true model and what is learnt from data.

- This is summarized in the so called bias-variance tradeoff for linear regression.

6 Basis functions

Deep learning $\approx$ learning basis functions from data. Here we discuss the usage of fixed basis functions as a prelude.

Linear functions try to fit a line - can be too restricted for fitting many datasets. Alternative: Map data $x_1, \ldots, x_n$ into some feature space by some mapping ϕ, then do a linear regression on $\phi(x_1), \ldots, \phi(x_n)$
Example: polynomial basis functions

$$\phi(x) = \begin{pmatrix} 1 \\ x^1 \\ x^2 \\ x^3 \\ \ldots \\ x^F \end{pmatrix}$$

(for a vector x do this on each dimension). By that:

$$w \cdot x = w_0 + w_1 x^1 + w_2 x^2 + w_3 x^3 + \ldots$$

Now the output is a polynomial in x of degree F with learnable coefficients w.

What is the relevance of basis functions beyond direct usage in linear regression?

- Radial basis functions are able to represent very general functions
- A layer in a neural network can be interpreted as: being a set of (learned, not fixed) basis functions that get used in the next layer.

Demo: `t2()` shows a solution for varying values of F

Demo: `t3()` shows train and test error for varying values of F

Problem: for n data points a polynomial with $F = n - 1$ achieves zero training error, but test error is too high – overfitting

Example: radial basis functions
Given a set of points $P = (x_1, \ldots, x_S)$, the gaussian radial basis function mapping is defined for any sample x as:

$$\phi(x) = (rbf(x, x_1), rbf(x, x_2), \ldots, rbf(x, x_S)) \in \mathbb{R}^S$$
$$rbf(u, v) = \exp(-\frac{\|u - v\|^2}{\lambda^2})$$

Idea: $\|x - x_i\|^2$ is a distance to the representative $x_i \in P$.
$exp(-\|x - x_i\|^2)$ is a kind of similarity.
The feature vector is a vector of similarities with respect to the samples in the set P.

Note here: x is a vector with the same dimensionality as the x_i. $\|u - v\|^2$ computes a real number from 2 vectors. So $rbf(x, x_i) = \exp(-\frac{\|x - x_i\|^2}{\lambda^2})$ is a real number. And $\phi(x)$ is a vector with S dimensions - which is the number of samples in the S.

Note also: this feature mapping has two parameters: the set of points P and the kernel width λ

How to obtain P ? Examples:

- sample at randomly S elements from the input features of the training data set $\{x_1, \ldots, x_n\}$

- perform k-means clustering on $(x_1, x_2, \ldots, x_n)$ with S centroids, P will be the centroids
- $P = D_n$ (use the whole training data, not always a good idea, locks feature dimensionality to sample size)

$$w \cdot \phi(x) = \sum_r w_r \exp(-\frac{\|x - x_r\|^2}{\gamma})$$

is a weighted sum of similarities between x and the set of points P

Implement it, and see its impact for different choices of kernel width γ in `learn.py`
Call `tRBF([val1,val2,val3])` with different values of the kernel width. For a too small kernel width one will have train error zero.