Machine Learning

by ambedkar@IISc

- Different Types of Learning
- Introdcution to Supervised Learning
- Some foundational aspects of ML
- Linear Regression

Note

On Learning and Different Types

Supervised Learning

Some Foundational aspects of Machine Learning

Linear Regression

Agenda

On Learning and Different Types

Supervised Learning

Some Foundational aspects of Machine Learning

Linear Regression

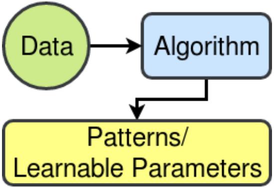
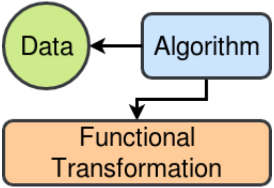
On Learning and Different Types

What is Learning?

It is hard to precisely define the learning problem in its full generality, thus let us consider an example:

	Problem 1	**Problem 2**
Input	Some cat images $\mathbf{C} = \{C_1, C_2, \dots, C_m\}$ and dog images $\mathbf{D} = \{D_1, D_2, \dots, D_n\}$	An array of numbers $\mathbf{a} = [a_1, a_2, \dots, a_n]$
Objective	Identify a new image X as cat/dog	Sort $\mathbf{a}$ in ascending order
Approach	?	Follow a fixed recipe that works in the same way for all arrays $\mathbf{a}$

What is Learning? (contd...)

Cat vs Dog	Sorting
Any approach with hard-coded "rules" is bound to fail	Hard-coded "rules" can sort any array
Algorithm must rely on previously observed data	Arrays sorted earlier will not affect the sorting of a new array
A good algorithm will get better as more data is observed	No such notion
	

Classification of Learning Approaches

- Learn by exploring data
  - Supervised Learning
  - Unsupervised Learning
- Learn from data, in a more challenging circumstances
  - Semi-supervised Learning
  - Domain Adaptation
  - Active Learning
- Learn by interacting with an environment
  - Multi-armed Bandits
  - Reinforcement Learning
- Very recent challenging AI paradigms
  - Zero/One/Few-shot Learning
  - Transfer Learning
  - Multi-agent reinforcement learning

Classification of Learning Approaches

- Supervised Learning - Separating spam from normal emails
- Unsupervised Learning - Identifying groups in a social network
- Reinforcement Learning - Controlled medicine trials
- Zero/One/Few-shot Learning - Learning from few examples
- Transfer Learning - Multi-task learning
- Semi-supervised Learning - Using labeled and unlabeled data
- etc.

Classification of Learning Approaches

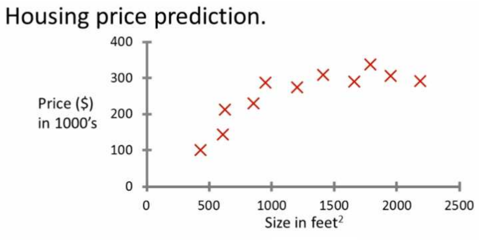
- Supervised Learning - Separating spam from normal emails
- Unsupervised Learning - Identifying groups in a social network

- Reinforcement Learning - Controlled medicine trials

- Zero/One/Few-shot Learning - Learning from few examples
- Transfer Learning - Multi-task learning
- Semi-supervised Learning - Using labeled and unlabeled data
- etc.

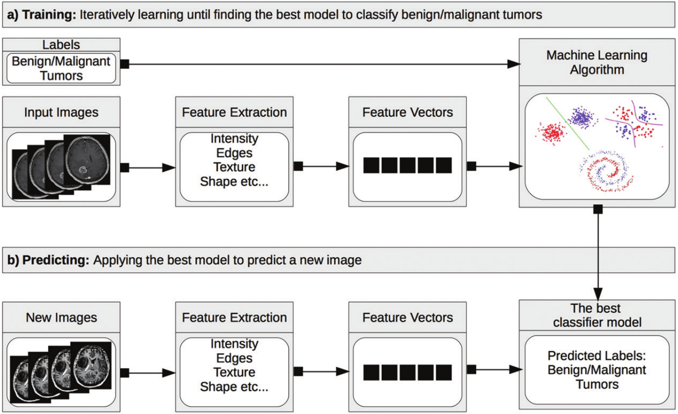
Supervised Learning

Regression: Example



Supervised Learning: Predicting housing prices

Classification: Example



Supervised Learning in Action for Medical Image Diagnosis[1]

[1] Image is taken from Erickson et al, Machine Learning for Medical Imaging, Radio Graphics, 2017

Who supervises "Learning"?

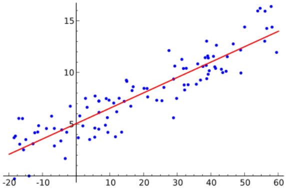
Answer: Ground-truth or labels.

- In supervised learning along with (input data x comes with ground-truth (or response (y)
  - If y takes only two values (at most finitely many values) it is a classification problem
  - If y takes any real number it is a regression
- Aim is to build a system f (or a function) such a way that
  - given x predict y **as accurately as possible**

Supervised Learning

- **Input:** A set of labeled examples $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{m}$ called the *training set*
- Each example $(\mathbf{x}^{(i)}, y^{(i)})$ is a pair of input representation $\mathbf{x}^{(i)} \in \mathcal{X} \subseteq \mathbb{R}^d$ and target label $y^{(i)} \in \mathcal{Y} \subseteq \mathbb{R}$
- The elements of $\mathbf{x}^{(i)}$ are known as *features*
- **Objective:** To learn a functional mapping $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$ that:
  - Closely mimics the examples in training set $(f_\theta(\mathbf{x}^{(i)}) \approx y^{(i)})$, *i.e.*, has low *training error*
  - Generalizes to unseen examples, *i.e.*, has low *test error*
- θ refers to *learnable parameters* of the function f_θ
- **Examples:**
  - Regression: $\mathcal{Y} = \mathbb{R}$
  - Classification: $\mathcal{Y} = \{1, 2, \ldots k\}$ for k class classification problem

- **Objective:** To learn a function mapping input features $\mathbf{x}$ to scalar target y
- Linear regression is the most common form - assumes that f_θ is linear in θ



Example - Linear Regression

- **Examples:**
  - Predicting temperature in a room based on other physical measurements
  - Predicting location of gaze using image of an eye
  - Predicting remaining life expectancy of a person based on current health records
  - Predicting return on investment based on market status

[1] Image source: `https://en.wikipedia.org/wiki/Linear_regression`

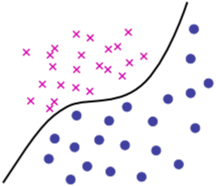
Supervised Learning - Regression (contd...)

Some popular techniques:

- Linear regression
- Polynomial regression
- Bayesian linear regression
- Support vector regression
- Gaussian process regression
- etc.

Supervised Learning - Classification

- **Objective:** To learn a function that maps input features $\mathbf{x}$ to one of the k classes
- The classes may be (and usually are) unordered



Example - Classification

- **Examples:**
  - Classifying images based on objects being depicted
  - Classifying market condition as favorable or unfavorable
  - Classifying pixels based on membership to object/background for segmentation
  - Predicting the next word based on a sequence of observed words

[1] Image source: `https://www.hact.org.uk`

Supervised Learning - Classification (contd...)

Some popular techniques:

- Logistic regression
- Random forests
- Bayesian logistic regression
- Support vector machines
- Gaussian process classification
- Neural networks
- etc.

Supervised Learning Setup

Problem: Given the data $\{(x_n, y_n)\}_{n=1}^{N}$, aim is to find a function.

$$f : \mathcal{X} \to \mathcal{Y}$$

that approximate the relation between X and Y.

- There are small letters, capitol letters, script letters. What are they?

- X and Y denotes the random variables and $\mathcal{X}$ and $\mathcal{Y}$ denotes the sets from where X and Y take values.

Random Variables? Why are we talking about probability here?

Supervised Learning Setup: Notation

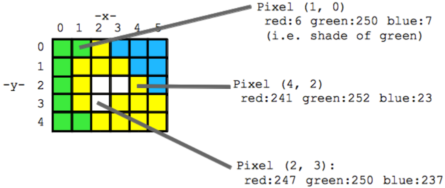
- The number of data samples that are available to us is N
- That is the samples are $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$
  - For example, $x_1, x_2, \ldots, x_N$ denote medical images and,
  - $y_1, y_2, \ldots, y_N$ represent ground-truth diagnosis say -1 or $+1$.
- Note that the data can be noisy
  - Scanner itself may introduce this noise
  - Doctors can make some mistake in their diagnosis

Supervised Learning Setup: Dimension

- Dimension is the size of the input data i.e x_n we denote this by D
- We write $x_n = (x_{n1}, \ldots, x_{nD}) \in \mathbb{R}^D$
  - If a grey scale image size is say 16×16 then $D = 16 \times 16$
  - If it is RGB then $D = 16 \times 16 \times 3$ and each x_{nd} takes value between 0 and 255.
- The dimension of $x_1, x_2, \ldots, x_N$ is typically very high
- Why?

Supervised Learning Setup: Dimension (Contd...)

- Number of pixels in an image 800 pixel wide, 600 pixels high: $800 \times 600 = 480000$. Which is 0.48 megapixels
- Typically digital images are $4 - 20$ megapixels



Pixels in RGB images[2]

- Now what is the dimension of 800×600 image?

[2] Taken from web

Supervised Learning Setup: Dimension (Contd...)

- Note that in some applications dimension of each sample can be varying, for example:
  - sentences in text
  - protein sequence data
- What about the response y?
  - Dimension of y is much much less than x
  - y can be structured and it leads to structure prediction learning
- A major issue in machine learning: **High dimensionality of data**

Some Foundational aspects of Machine Learning

On Statistical Approach to Machine Learning

Assumption behind the statistical approach to Machine Learning:

> Data is assumed to be sampled from a underlying probability distribution

On Statistical Approach to Machine Learning (contd...)

- Suppose we are given N samples $x_1, \ldots, x_N$
- Our assumption is that there is a hypothetical underlying distribution P from which these samples are drawn
  - The problem is that we do not know this distribution
  - Some machine learning algorithms try to estimate this distribution, some try to solve problems without estimating this distribution
- Recall, class conditional densities $P(x|y_1)$ and $P(x|y_2)$
  - In the Bayes classifier uses these distributions
  - We are given only data, from which we need to estimate these distributions (How?)

How complicated this underlying distribution can be?

Loss Function

We need some guiding mechanism that will tell us how good our predictions are given an input.

- $\ell(y, f(x))$ denotes the loss when x is mapped to $f(x)$, while the actual value is y.

Note

- ℓ and f are specific to the problems and a method.
- For example, $\ell(.)$ can be a squared loss and $f(x)$ is linear function i.e $f = w^\intercal x$.

Learning as an optimization

Objective

Given a loss function ℓ, aim is to find f such that,

$$L(f) = \mathsf{E}_{(x,y)\sim P}[\ell(Y, f(X))]$$

is minimum

- Here X and Y are random variables.
- L is the true loss or expected loss or Risk.
- As we mentioned before we assume that the data is generated from a joint distribution $P(X, Y)$.

Diversion: Probability Basics

- Random variable is nothing but a function that maps outcome to a number
  - Consider a coin tossing experiment: Outcomes are `H` and `T`
  - Random variable X can map `H` to 1 and can map `T` to 0
- Now let us assign probabilities
  - Suppose $P(X = 1) = \frac{1}{4}$ and $P(X = 0) = \frac{3}{4}$
  - That is probability mass function of X is $(\frac{1}{4}, \frac{3}{4})$
- Let us calculate expectation of a random variable

$$\mathsf{E}_P X = \sum_{i=1}^{2} x_i p_i = 1\left(\frac{1}{4}\right) + 0\left(\frac{3}{4}\right)$$

Empirical Risk

Problem: We cannot estimate the true loss as we do not know P.

Some Relief: But we have some samples that are drawn from P.

Empirical Risk

Instead of minimizing the true loss find f that minimizes empirical risk

$$L_{emp}(f) = \frac{1}{N}\sum_{n=1}^{N} \ell(y_n, f(x_n))$$

$$i.e. \qquad f^* = \arg\min_f \frac{1}{N}\sum_{n=1}^{N} \ell(y_n, f(x_n))$$

Empirical Risk

$$L_{emp}(f) = \frac{1}{N}\sum_{n=1}^{N} \ell(y_n, f(x_n))$$

$$i.e. \qquad f^* = \arg\min_f \frac{1}{N}\sum_{n=1}^{N} \ell(y_n, f(x_n))$$

- Here $\ell(y_n, f(x_n))$ is the per sample loss
- $L_{emp}(f)$ is the overall loss given the data $\{(x_n, y_n)\}_{n=1}^N$
- N is the number of samples and we need "reasonably many" samples so that Empirical Risk is close to the True Risk
- Why do we need Empirical Risk to be closer to the True Risk?

Generalizing Capacity

How well the learned function work on the unseen data?

- We want f not only work on the training data $\{(x_n, y_n)\}_{n=1}^N$ but also it should work on the unseen data.
- For this the general principle:

$$f \text{ should be } \textit{simple}$$

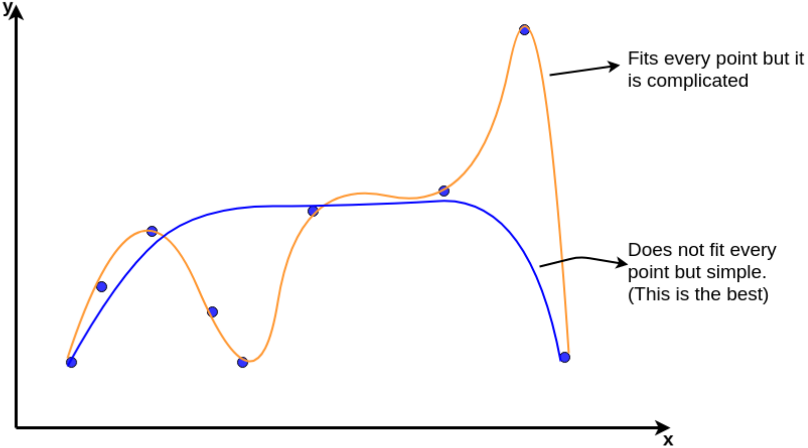
- Regularizer

$$f^* = \arg\min_f \frac{1}{N}\sum_{n=1}^{N} l(y_n, f(x_n)) + \lambda R(f)$$

  - λ controls how much regularization one needs.
  - R measures complexity of f.
  - This is regularized risk minimization.

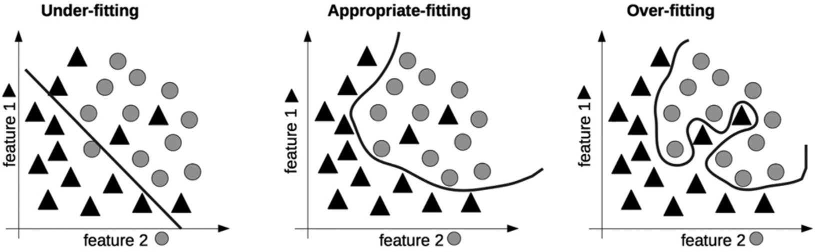
Generalizing Capacity(cont...)

- What we want to achieve.
  - Small empirical error on training data, and at the same time,
  - f needs to be simple.
- There is a trade off between these two goals
  - λ is a hyperparameter that tries to achieve this.



The blue curve has better generalization capacity. The orange curve overfits the data

Generalizing Capacity(cont...)



Learning as the Optimization

- **Note**: We have the following optimization problem "find f such that _ _ _ _ "
- Is it any f ?
- No, The choice f cannot be from a arbitrary set.
- First we fix $\mathcal{F}$: the set of all possible functions that describe relation between X and Y given training data $\{(x_n, y_n)\}_{n=1}^{N}$
- Now our objective is

$$f^* = \arg\min_{f \in \mathcal{F}} \sum_{n=1}^{N} \ell(y_n, f(x_n)) + \lambda R(f)$$

- For example, If $\mathcal{F}$ is set of all linear functions then we call it linear regression.

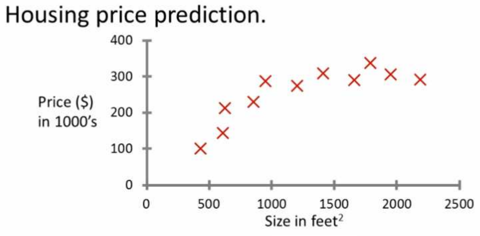
Linear Regression

Linear Regression: One dimensional Case

- Given N data samples of input and response pairs
- Suppose the data given to us is

$$(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$$

- Further, assume that input data dimension is just 1



Problem: *Find a straight line that **best** fits these set of points*

Linear Regression: One dimensional Case (contd...)

Assumption: Input and response relationship is ***linear*** (We hope so)

- $\{(x_n, y_n)\}_{n=1}^{N}$, $x_n \in \mathbb{R}$, $y_n \in \mathbb{R}$, find a straight line that **best** fits these set of points.
- (Rephrase) Given .... choose a straight line that best fits these set of points
  - i.e $\mathcal{F}$ is set of all linear functions.
  - In this case $\mathcal{F}$ denotes set of all straight lines on a plane.

Linear Regression: One dimensional Case (contd...)

From where do we choose or learn our solution from?

- Assume that $\mathcal{F}$ is set of all straight lines
- Further assume that $\mathcal{F}$ is set of all straight lines that are passing through origin.
  - Is this reasonable?
  - Yes! With some preprocessing we can transform the data
- That is define $\mathcal{F}$ as

$$\mathcal{F} = \{f_w(x) = wx : w \in \mathbb{R}\}$$

- $\mathcal{F}$ is paramerized by w

Note: Since f can be identified by w, our aim is to just learn w from the given data

Linear Regression: One dimensional Case (contd...)

'Best' with respect to what?

- We need some mechanism to evaluate our solution.
- For this we need to define a **loss function**
- A loss function takes two inputs: (i) response given by our solution, and (ii) groundtruth
- Loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is defined as

$$\ell(f) = \sum_{n=1}^{N} (y_n - f_w(x_n))^2$$

which is a least squared error.

Linear Regression: One dimensional Case (contd...)

Recall what we are trying to do

$$\ell(f_w) = \sum_{n=1}^{N} (y_n - f_w(x_n))^2$$

- Note that $y_n - f_w(x_n)$ is per sample loss
- $\ell(f_w)$ is the total loss
- Now aim is to find $w \in \mathbb{R}$ that minimizes empirical risk $\ell(f_w)$.

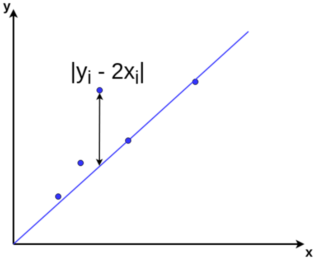
Note: Remember that we supposed to minimize true risk, since we do not know the underlying distribution we minimize empirical risk.

- **Optimization Problem:** Find f in $\mathcal{F}$ that minimizes $\ell(f)$

  $$|||$$

  Find $w \in \mathbb{R}$ that minimizes $\ell(w)$

  Since f is completely determined by w.



Linear Regression in one dimension.

Linear Regression: One dimensional Case (cont...)

Solution: A solution to this problem is given by

$$\frac{d\ell}{dw} = 0$$

This can be calculated as follows. First we will calculate the derivative of ℓ w.r.t w.

$$\ell(w) = \sum_{n=1}^{N} (y_n - wx_n)^2$$

$$\frac{d\ell}{dw} = \sum_{n=1}^{N} 2(y_n - wx_n)(-x_n)$$

$$= \sum_{n=1}^{N} (wx_n^2 - x_n y_n)$$

$$\implies \sum_{n=1}^{N} (wx_n^2 - x_n y_n) = 0$$

Solution: A solution to this problem is given by

$$\frac{d\ell}{dw} = 0$$

Now by equating the derivative to 0 we get

$$\implies \sum_{n=1}^{N} (wx_n^2 - x_n y_n) = 0$$

$$\implies w \sum_{n=1}^{N} x_n^2 = \sum_{n=1}^{N} x_n y_n$$

$$\implies w = \frac{\sum_{n=1}^{N} x_n y_n}{\sum_{n=1}^{N} x_n^2}$$

Linear Regression (General formulation)

- Given a training data $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^{N}$, where
  - $x_n \in \mathbb{R}^D$ is input
  - $y_n \in \mathbb{R}$ is response
- **Model:** Linear

$$y = f_w(x) = b + \sum_{j=1}^{m} w_j \phi_j(x), \qquad \text{where}$$

w_j : Model parameters

ϕ_j : basis function(changes the representation of x)

or

$$y = b + w^{\intercal}\phi(x), \qquad \text{where}$$

$$w^{\intercal} = [w_1, \ldots, w_m] \qquad \phi^{\intercal} = [\phi_1, \ldots, \phi_m]$$

Linear Regression (cont ...)

- **Model:** $y = f_w(x) = b + \sum_{j=1}^{m} w_j \phi_j(x)$
  - If we set $d = m$ and $\phi(x) = x_i, \quad i = 1, 2, \ldots, D.$
- **Model:** $\qquad y = b + w^{\intercal} x$
- Now by using the training data

$$X = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix}_{N \times D} \qquad Y = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}_{N \times 1}$$

We get

$$Y = XW + b$$

- We have

$$Y = XW + b$$

$$\begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} x_{11} \; x_{12} \; \dots \; x_{1d} \\ \vdots \\ x_{N1} \; x_{N2} \; \dots \; x_{Nd} \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix}_{d \times 1} + \begin{bmatrix} b \\ \vdots \\ b \end{bmatrix}$$

$$\implies \underbrace{\begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}}_{\substack{N \times 1 \\ \text{Matrix}}} = \underbrace{\begin{bmatrix} 1 \; x_{11} \; x_{12} \; \dots \; x_{1d} \\ 1 \; x_{21} \; x_{22} \; \dots \; x_{2d} \\ \vdots \\ 1 \; x_{N1} \; x_{N2} \; \dots \; x_{Nd} \end{bmatrix}}_{\substack{N \times (d+1) \\ \text{Matrix}}} \underbrace{\begin{bmatrix} b \\ w_1 \\ \vdots \\ w_d \end{bmatrix}}_{\substack{(d+1) \times 1 \\ \text{Matrix}}}$$

$$\implies Y = XW$$

Linear Regression (cont...)

- We have the following problem:

$$\text{Given } Y = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \text{ and } X = \begin{bmatrix} 1 \; x_{11} \; x_{12} \; \dots \; x_{1d} \\ \vdots \\ 1 \; x_{N1} \; x_{N2} \; \dots \; x_{Nd} \end{bmatrix}$$

$$\text{Find } W = \begin{bmatrix} b \\ w_1 \\ \vdots \\ w_d \end{bmatrix} \text{ which satisfies}$$

$$Y = XW$$

- Solving the linear system: The above system may not have a solution i.e parameter that satisfies

$$y_n = w^\intercal x_n, \quad n = 1, 2 \dots N$$

may not exists.

Least Square Approximation

- Least Square error

$$l(y_n, w^\intercal x_n) = (y_n - w^\intercal x_n)^2$$

  - [*] Note: One can also use

$$l(y_n, w^\intercal x_n) = |y_n - w^\intercal x_n|$$

which is more robust to outliers.

- The total empirical error

$$\begin{aligned} L_{emp}(w) &= \sum_{x=1}^{N} l(y_n, w^\intercal x_n) = \sum_{n=1}^{N} (y_n - w^\intercal x_n)^2 \\ &= (Y - XW)^\intercal (Y - XW) \end{aligned}$$

-

$$W^* = \arg\min_w \sum_{n=1}^{N} (y_n - w^\intercal x_n)^2$$

Least Square Solution

- **Recall** Least square objective : Given data $\{(x_n, y_n)\}_{n=1}^N$, find w such that $L_{emp}(w) = \sum_{n=1}^N (y_n - w^\intercal x_n)^2$ is minimum.
- **Solution**

$$\frac{\partial L_{emp}}{\partial w} = \sum_{n=1}^{N} 2(y_n - w^\intercal x_n)\frac{\partial}{\partial w}(y_n - w^\intercal x_n) = 0$$

$$\implies \sum_{n=1}^{N} x_n(y_n - x_n^\intercal w) = 0 \qquad (\underline{\text{Note:}}\ x_n^\intercal w = w^\intercal x_n)$$

$$\implies \sum_{n=1}^{N} x_n y_n - \sum_{n=1}^{N} x_n x_n^\intercal w = 0$$

$$\implies \sum_{n=1}^{N} x_n x_n^\intercal w = \sum_{n=1}^{N} x_n y_n$$

Least Square Solution (Cont...)

Objective: Given data $\{(x_n, y_n)\}_{n=1}^{N}$, find w such that minimize

$$L_{emp}(w) = \sum_{n=1}^{N} (y_n - w^\intercal x_n)^2$$

Final Solution:

$$\begin{aligned} w &= (\sum_{n=1}^{N} x_n x_n^\intercal)^{-1} \sum_{n=1}^{N} y_n x_n \\ &= (X^\intercal X)^{-1} X^\intercal Y \end{aligned}$$

When output is vector valued:

- The same solution holds if response y is vector valued i.e Y is $n \times K$ matrix (i.e k responses per input)
- In this case W will be $d \times K$ matrix

Some Remarks

- $X^\intercal X$ is a $d \times d$ matrix(d is the dimension of the data) and it can be very expensive to invert $X^\intercal X$
- $W = [b, w_1, \ldots, w_d]$, w_is can become very large trying to fit the training data.
- IMPLICATION: The model becomes very complicated.
- RESULT: The model overfits.
- SOLUTION: Penalize large values of the parameter.
- Regularization.

Ridge Regression (Linear Regression with Regularization)

- **Modified Objective**: Given data $\{(x_n, y_n)\}_{n=1}^{N}$, find w such that

$$L_{emp}(w) = \sum_{n=1}^{N}(y_n - w^\intercal x_n)^2 + \lambda||w||^2$$

  - Here $||w||^2 = w^\intercal w$
  - λ is the hyperparameter, that controls amount of regularization.

- **Solution**:

$$\frac{\partial L(W)}{\partial w} = \sum_{n=1}^{N} 2(y_n - w^\intercal x_n)(-x_n) + 2\lambda w = 0$$

Ridge Regression(cont...

$$\implies \lambda(w) = \sum_{n=1}^{N} x_n(y_n - x_n^\intercal w)$$

$$\implies \lambda(w) = \sum_{n=1}^{N} x_n y_n - \sum_{n=1}^{N} x_n x_n^\intercal w$$

$$\implies \lambda W = X^\intercal Y - X^\intercal X W$$

$$\implies \lambda W + X^\intercal X W = X^\intercal Y$$

$$\implies (\lambda \mathtt{I}_d + X^\intercal X) W = X^\intercal Y$$

$$\implies W = (X^\intercal X + \lambda \mathtt{I}_d)^{-1} X^\intercal Y$$

<u>Note</u>: $X^\intercal X$ is a $d \times d$ matrix

On Regularization

Claim: Small weights, $w = (w_1, \ldots, w_d)$ ensure that the function $y = f(x) = w^\intercal x$ is *smooth*.

Justification:

- Let $x_n, x_m \in \mathbb{R}^d$ such that

$$x_{n_j} = x_{m_j}, \quad j = 1, 2, \ldots, d-1 \quad \text{but } |x_{n_d} - x_{m_d}| = \epsilon$$

- Now $|y_n - y_m| = \epsilon w_d$
- If w_d is large then $|y_n - y_m|$ is large.
- This implies in this case $f(x) = w^\intercal x$ does not behave smoothly.

On Regularization (cont...)

- Hence regularization helps: which makes the individual components of w small.
- That is, **Do not** learn a model that gives a simple feature too much importance
- Regularization is very important when N is small and D is very large.

Ridge Regression Solution

- Directly with matrices

$$
\begin{aligned}
L(w) &= \frac{1}{2}(Y - XW)^\intercal (Y - XW) + \frac{\lambda}{2} W^\intercal W \\
\nabla L(w) &= -X^\intercal (Y - XW) + \lambda W = 0 \\
&\implies X^\intercal X W + \lambda W = X^\intercal Y \\
&\implies (X^\intercal X + \lambda \mathtt{I}) W = X^\intercal Y \\
\text{Hence } W^* &= (X^\intercal X + \lambda \mathtt{I})^{-1} X^\intercal Y
\end{aligned}
$$

- One more advantage of Regression:
- If $X^\intercal X$ is not invertible, one can make $(X^\intercal X + \lambda \mathtt{I}_d)$ invertible.

Gradient Descent Solution for Least Squares

- We have the following least square solution

$$W^* = (X^\intercal X)^{-1} X^\intercal Y$$
$$W^*_{reg} = (X^\intercal X + \lambda \mathtt{I}_d)^{-1} X^\intercal Y$$

- Which involves inverting a $d \times d$ matrix.
- In the case of high dimensional data it is prohibitively difficult.
- Hence we turn to gradient Descent Solution.
  - Optimization methods that is based on gradients.
  - May stuck in a local optima.

Gradient Descent Procedure

Procedure:

1. Start with an initial value $w = w^{(0)}$

2. Update w by moving along the gradient of the loss function $L(L_{emp}$ or $L_{reg})$

$$w^{(t)} = w^{(t-1)} - \eta \frac{\partial L}{\partial w}\Big|_{w=w^{(t-1)}}$$

3. Repeat until convergence.

Gradient Descent Procedure (contd...)

We have

$$\frac{\partial L}{\partial w} = \sum_{n=1}^{N} x_n (y_n - x_n^{\intercal} w)$$

Procedure:

1. Start with an initial value $w = w^{(0)}$
2. Update w by moving along the gradient of the loss function $L(L_{emp}$ or $L_{reg})$

$$w^{(t)} = w^{(t-1)} - \eta \sum_{n=1}^{N} x_n (y_n - x_n^{\intercal} w^{(t-1)})$$

3. Repeat until convergence.

On Convexity

- The squared loss function in linear regression is convex.
  - With ℓ_2 regularizer it is strictly convex.

Convex Functions:

For scalar functions :	Convex if the second derivative is nonnegative everywhere
For vector valued :	Convex if Hessian is positive semi definite

On ℓ_1 Regularizer

ℓ_1 regularizer $\quad R(w) = ||w||_1 = \sum_{j=1}^{d} |w_j|$

- Promotes w to have very few non zero components.
- Optimization in this case is not straight forward.