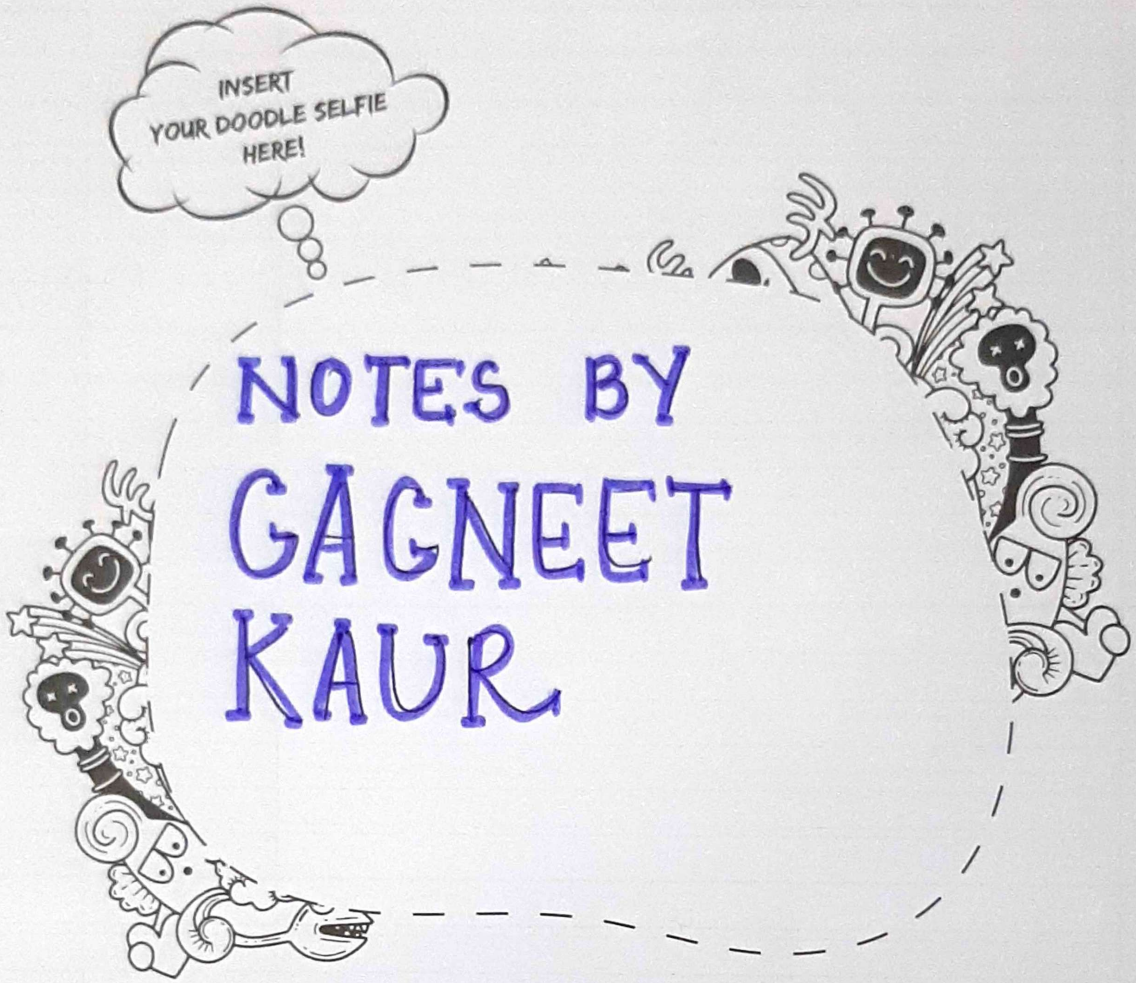


NOTES BY GAGNEET KAUR

NAME Gagneet Kaur

ADDRESS

MACHINE LEARNING TECHNIQUES [MLT]

MOBILE

EMAIL 21F1006856@ds.study.iitm.ac.in

EMERGENCY CONTACT

Date: February 20, 2023

Page
Date

SUPERVISED LEARNING
Linear Regression

WEEK 5

SUPERVISED LEARNING

- Supervised learning, as the name indicates, has the presence of a supervisor as a teacher.
- Basically supervised learning is when we teach or train the machine using data that is WELL-LABELLED.

INPUT : $\{x_1, x_2, \ldots, x_n\}$ $x_i \in \mathbb{R}^d$ ← Features/Attributes

$\{y_1, y_2, \ldots, y_n\}$ $y_i \in \mathbb{R}$ ← Labels (supervision)

- Supervised learning has following categories :

1. Binary Classification
   → when the output variable is category of 'two' only.
   for example : Spam or non-spam.

2. Multiclass Classification
   → when the output variable is category of many classes.
   for example : digit classification, i.e, $\{0, 1, \ldots, k\}$

3. Regression
   → when the output variable is a 'REAL VALUE'.
   for example : rainfall prediction.

Page No:
Date:

REGRESSION

Input Data / Training Data : $\{x_1, x_2, \ldots, x_n\}$ $x_i \in \mathbb{R}^d$

$\{y_1, y_2, \ldots, y_n\}$ $y_i \in \mathbb{R}$

GOAL : To learn how to predict for a given new feature, which means that we basically want to learn :

$$h : \mathbb{R}^d \longrightarrow \mathbb{R}$$

- How do we measure goodness of a function ?

→ Error function (Error should be non-negative)

$$\text{error}(h) = \sum_{i=1}^{n} \left(h(x_i) - y_i \right)^2$$

- How small can this error be ?

→ as small as ZERO

- Which 'h' achieves zero error ?

→ $h(x_i) = y_i \quad \forall\ i$ → this is memorizing data

ISSUES : (1) By memorizing, we can get zero error on 'TRAINING DATA'

(2) What we care about is — 'TEST PERFORMANCE'

(3) What can we do — Impose 'STRUCTURE' to reduce search space

Page No:
Date:

Simplest Structure : LINEAR STRUCTURE

$$H_{linear} = \{ h_w : \mathbb{R}^d \to \mathbb{R} \mid h_w(x) = w^T x \quad \forall \; w \in \mathbb{R}^d \}$$

MODIFIED GOAL :

$$\min_{h_w \in H_{linear}} \sum_{i=1}^{n} \left(h_w(x_i) - y_i \right)^2$$

OR EQUIVALENTLY

$$\min_{w \in \mathbb{R}^d} \sum_{i=1}^{n} \left(w^T x_i - y_i \right)^2$$

Linear Regression

* If a linear regression model achieves zero training error, we can say that all data points lie on a hyperplane in the $(d+1)$ dimensional space.

* When a model tries to interpolate each data point, the model tends to overfit. Thus, TRAINING ERROR will be low but TEST ERROR may be high.

Page No:
Date:

OPTIMIZING THE ERROR FUNCTION

$$\min_{W \in \mathbb{R}^d} \sum_{i=1}^{n} (W^T x_i - y_i)^2 = \| x^T W - Y \|_2^2$$

Let us try with matrices :

→ Suppose a matrix X where all the data points are stacked in columns. In other words, x^T is a matrix where the data points are stacked in rows

$$X^T = \begin{bmatrix} — x_1 — \\ — x_2 — \\ \vdots \\ — x_n — \end{bmatrix}$$

← This is $n \times d$ matrix as $x_i \in \mathbb{R}^d$.

→ We also have a 'label vector' - Y. (which gives labels for each of our data points)

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

← This is $n \times 1$ matrix

→ Our parameter vector 'W'.

$$W = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix}$$

← This is a $d \times 1$ matrix.

Page No:
Date:

Hence,

$$\min_{w \in \mathbb{R}^d} \sum_{i=1}^{n} (w^T x_i - y_i)^2 = \| x^T w - y \|^2$$

equivalently,

$$\min_{w \in \mathbb{R}^d} (x^T w - y)^T (x^T w - y)$$ ← unconstrained quadratic (in w) optimization problem

It is unconstrained because we allow for any w in $\mathbb{R}^d$; there is no condition on w that we are imposing.

→ Now, How can we find the 'w' that has the smallest value among this ?

Solution : Take the derivative wrt w and set to zero !!

$$f(w) = (x^T w - y)^T (x^T w - y)$$

$$\therefore \nabla f(w) = 2(xx^T)w - 2xy$$

Now, we want to set the gradient to ZERO.

$$\Rightarrow 2(xx^T)w - 2xy = 0$$

$$\Rightarrow (xx^T)w = xy$$

→ w^* satisfies this equation. Hence,

$$(xx^T)w^* = xy$$

Page No:
Date:

Let us write w^* in a closed form :

$$w^* = \underbrace{(xx^T)}^{\dagger} xy$$

$\dagger \rightarrow$ pseudo inverse

$d \times d$ matrix $\rightarrow$ scaled covariance matrix

* pseudo-inverse : it is like a many to one mapping, ie, there are so many points which go to the same point. Now, if you have to do an inverse - if it is one to one mapping, the inverse would give you a correct value. But if there is a whole subspace in a finite space, which maps to the same vector on the other side via transformation, the PSEUDO INVERSE would map it to the one that has the least length.

OBSERVATIONS :

(1) Like PCA, w^* depends on a 'COVARIANCE' like matrix. Unlike PCA, here we have LABELS.

(2) The labels should also dictate, how the answer w^* looks like.

(3) Linear regression has a closed form solution (w^*)

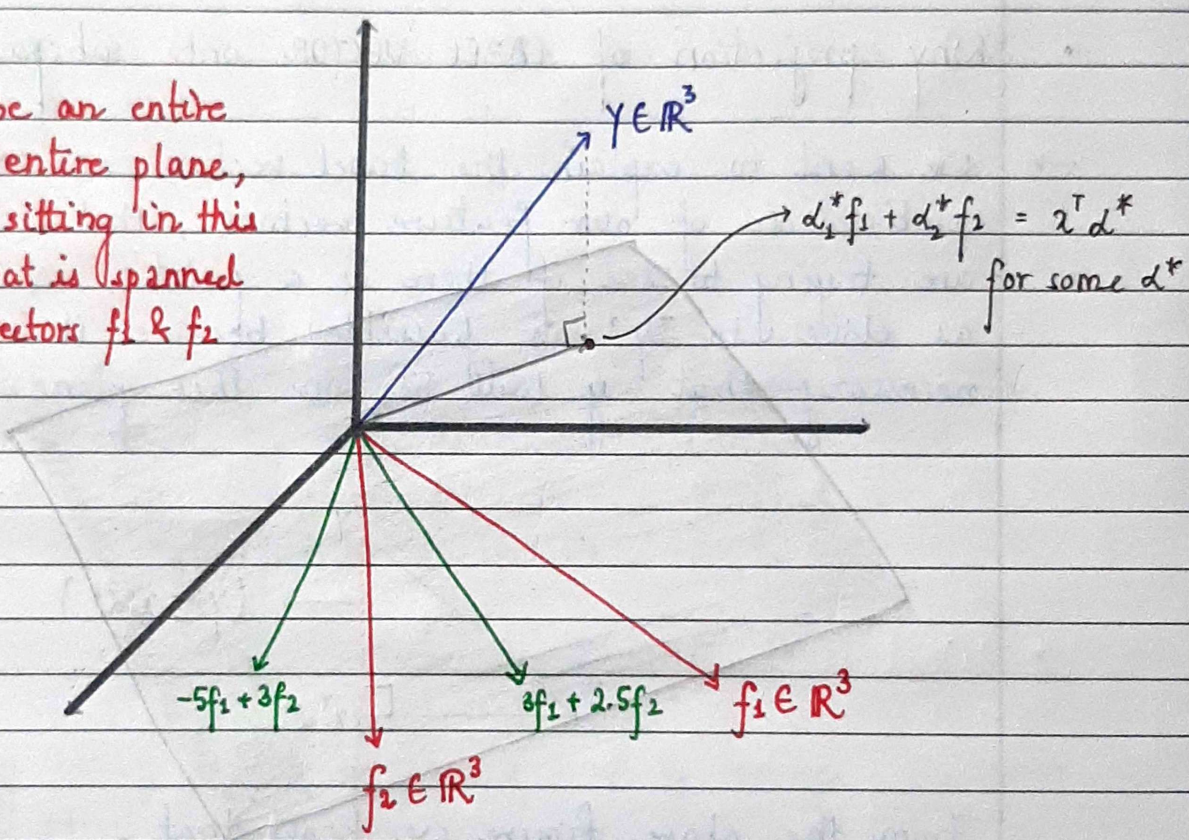
GEOMETRIC INTERPRETATION OF LINEAR REGRESSION

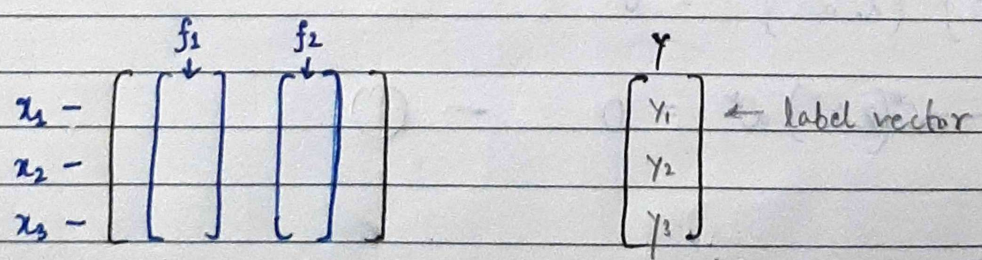
$$w^* = (xx^T)^{\dagger} xy$$

→ How can we interpret this geometrically ?

Let us suppose $d = 2$ (no. of features)
$n = 3$ (no. of data points)

There might be an entire subspace or entire plane, a 2-D plane sitting in this 3-D plane, that is spanned by the two vectors f_1 & f_2





Page No:
Date:

Projection of vector (LABEL VECTOR) y onto the plane spanned by f_1 & f_2 can be written as :

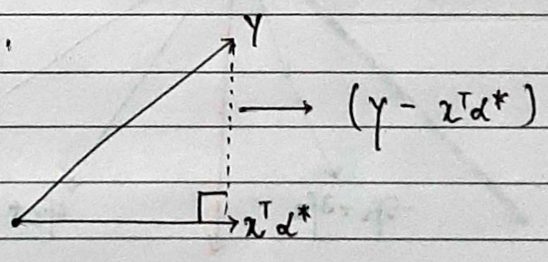
$$\alpha_1^* f_1 + \alpha_2^* f_2 = x^T \alpha^* \quad (\text{for some } \alpha^*)$$

$$\begin{bmatrix} | & | \\ f_1 & f_2 \\ | & | \end{bmatrix} \begin{bmatrix} \alpha_1^* \\ \alpha_2^* \end{bmatrix} = x^T \alpha^*$$

x^T $\quad$ α^*

- Why projection of LABEL VECTOR onto subspace?

→ We want to explain the label vector as some combination of our feature vectors, that is why we are trying to see if there is a point, which is as close to 'y' as possible because it is not necessary that y will be on this plane.



From the above figure we know that,

$$(y - x^T\alpha^*)^T (x^T\alpha^*) = 0$$

$$\Rightarrow \quad y^T x^T \alpha^* - \alpha^* (x x^T) \alpha^* = 0 \quad - (1)$$

We have : $y^T x^T \alpha^* - \alpha^* (xx^T) \alpha^* = 0$

Recall that,

$$w^* = (xx^T)^\dagger xy$$

Substituting w^* as α^* in (1), we get (Use xx^T is symmetric)

$$y^T x^T w^* - w^* (xx^T) w^* = 0$$

$$\Rightarrow y^T x^T \left[(xx^T)^\dagger xy\right] - \left[(xx^T)^\dagger xy\right](xx^T)\left[(xx^T)^\dagger xy\right] = 0$$

This is basically tell us that the point that is closest to y that is spanned by our features is $\rightarrow x^T w^*$.

CONCLUSION :

$x^T w^*$ is the 'PROJECTION' of the labels onto the subspace spanned by the features.

This is the GEOMETRIC INTERPRETATION.

When y belongs to the subspace spanned by features, the error (squared error) will be zero !!

Page No:
Date:

GRADIENT DESCENT

We have,

$$w^* = (xx^T)^{\dagger} xy$$

→ inverse computation is expensive if d is large

PROBLEM : The time complexity of finding linear regression solutions using the first derivative test is $O(d^3)$.
→ Here d is the number of features.
→ It is very computationally expensive

QUESTION : Is there a different way to achieve the same w^* without computing the inverse?

- We know w^* is the solution of an unconstrained optimization problem.
- w^* is just the minimizer of the sum of the squared error.
- One method to get w^* is → GRADIENT DESCENT.

What is Gradient-Descent?

→ It is an iterative way to find minimizers of functions using just first order information.

What is Gradient Descent Algorithm?

1. Start with an initial guess for the solution
2. Let us call that guess $0 - w_0$ (arbitrary points in d dimension)
3. Compute the gradient of the objective function at w_0.

Page No:
Date:

4. Since, a gradient would tell you the direction in which you have to move such that the function increases the most. So, we have to go opposite of it.
5. We want to minimize the function, naturally GRADIENT DESCENT would tell you to move in the direction opposite to what the gradient is pointing.

Update Rule:

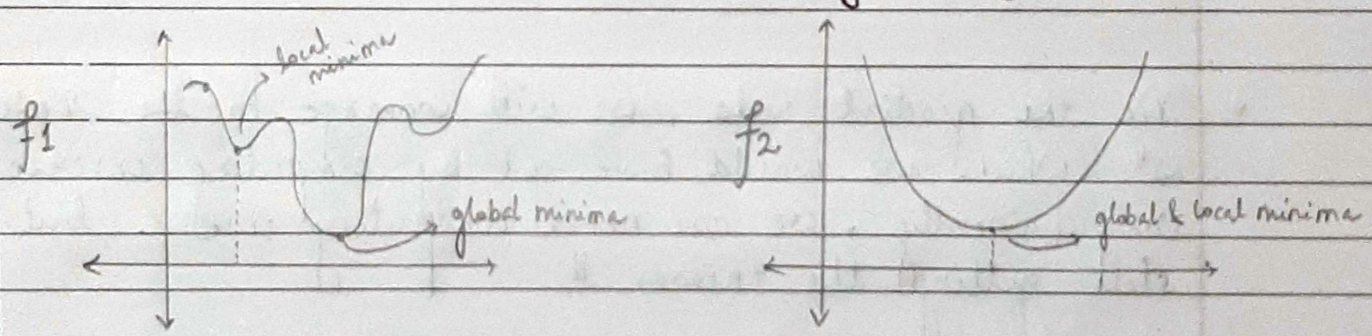
$$w^{t+1} = w^t - \eta^t \nabla f(w^t)$$

η^t → scalar step size

$\nabla f(w^t)$ → gradient at w^t

6. Eventually, after iterations of algorithm, eventually we will reach - LOCAL MINIMA of the function.

- Local Minima is a point where the gradient becomes ZERO.
→ it does not mean that it is necessarily the global minima.



7. Once you reach a local minima, then your algorithm does not proceed any further. It stops at that point because the gradient becomes zero & then the update rule does not proceed any further.

Page No:
Date:

The function which we care about is SUM OF SQUARED ERRORS → it is a quadratic function, ie, it has a unique global minima. → which means the gradient descent algorithm will converge to the optimal point.

- Does this solve our problem of not computing inverse? Let's see :

$$f(w) = \| x^T w - y \|^2 = \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

$$\therefore \nabla f(w) = 2(xx^T)w - 2xy$$

GRADIENT DESCENT UPDATE RULE FOR LINEAR REGRESSION :

$$w^{t+1} = w^t - \eta^t \left[2(xx^T)w - 2xy \right]$$

- Via the gradient rule, we will converge to the same w^* which we would have got by computing inverse. Computationally, we are never computing inverse but still getting the answer !!

Even if 'd' is much, much larger, even if it is in millions, can still run this algorithm.

Now, another ISSUE !! What if 'n' is large ??

XX^T is a $d \times d$ matrix which came from $d \times n$ times $n \times d$ matrix ! So there is an 'n' dependency hiding inside this XX^T itself.

→ So, we might want to avoid XX^T computation itself.

→ Not just the inverse of it, but XX^T might be hard to compute, if you have huge no. of data points.

How to adapt gradient descent to this situation?

SOLUTION : STOCHASTIC GRADIENT DESCENT

for $t = 1, \ldots, T$ (batch size)

- At each step, sample a bunch of datapoints (k) uniformly at random from the set of all points
- PRETEND this sample is the entire dataset and take a gradient step wrt it

$$2\left[(\tilde{x}\tilde{x}^T)w - \tilde{x}\tilde{y}\right]$$ → manageable because $\tilde{x} \in \mathbb{R}^{d \times k}$ where k = # sample points.

- END

Note : At every round, you are sampling k points, which means these k points are not the same at every round.

This sample & the gradient that comes out of the sample is a noisy gradient.

After T rounds use,

$$w_{SGD}^T = \frac{1}{T}\sum_{i=1}^{T} w^t$$

→ Guaranteed to converge to OPTIMA with high probability.

Page No:
Date:

KERNEL REGRESSION

How can we do non-linear regression?

→ It is as if you want to map your data points to a higher dimension and then learn a linear model in high dimension. But we don't want to explicitly compute these high dimensional mappings.

WHAT CAN WE DO?

→ if our regression solution somehow can be written in terms of the dot products between data points

$$w^* = (xx^T)^{\dagger} xy \quad \leftarrow \text{Regression Solution.}$$

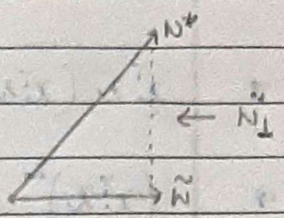
- w^* must lie in the span of the data points
- As we know, w^* is the solution that minimizes the sum of squared error:

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

- If this w^* was actually out of this plane, let us assume $\tilde{w}$ which is the projection of this w^* onto this plane, i.e. the closest point of w^* on this plane.
- Thus the sum of squared error will be:

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

- What can we say about the above eqn?



We can write :

$$w^* = \tilde{w} + \tilde{w}_\perp$$

Now,

$$w^{*T} x_i = \tilde{w} + \tilde{w}^T x_i$$

It is same as :

$$w^{*T} x_i = \tilde{w}^T x_i + \tilde{w}_\perp^T x_i$$

$$\Rightarrow \boxed{w^{*T} x_i = \tilde{w}^T x_i}$$

Which means that the prediction of w^* & $\tilde{w}$ for the all data points are exactly the same, which means that the error both of these will suffer is exactly SAME.

$$\sum_{i=1}^{n} (w^{*T} x_i - y_i)^2 = \sum_{i=1}^{n} (\tilde{w}^T x_i - y_i)^2$$

Conclusion : w^* equals some linear combinations of the data points.

$$w^* = x\alpha^* \quad \text{for some } \alpha^* \text{ in } \mathbb{R}^n$$

Also, $w^* = (xx^T)^\dagger xy$

$$\therefore \quad x\alpha^* = (xx^T)^\dagger xy$$

$$\Rightarrow \quad (xx^T)\, x\alpha^* = (xx^T)(xx^T)^\dagger xy$$

$$\Rightarrow \quad x(x^T x)\alpha^* = xy$$

$\Rightarrow \quad x^T x\,(x^T x)\,\alpha^* = x^T x\, y$

$\Rightarrow \quad (x^T x)^2\,\alpha^* = (x^T x)\, y$

Now,

$x^T x \longrightarrow$ kernel matrix (k)

$\therefore \quad k^2 \alpha^* = k y$

and if k is invertible,

$$\boxed{\alpha^* = k^{-1} y}$$

What Is The Use Of This ?

Now, when you do prediction - if you have a new data point x_{test} in d dimension, then the KERNEL would tell you that, I need $w^{*T}\phi$ of x_{test}. So I have to map my x_{test} to high dimension & then take dot product with w^*.

So, $\quad w^* = \underset{min}{arg} \sum_{i=1}^{n} (\phi x_i - y_i)^2$

PREDICTION : for some $x_{test} \in \mathbb{R}^d$,

$$w^* \phi(x_{test})$$

$$= \left[\sum_{i=1}^{n} \alpha_i^* \phi(x_i)\right]^T \phi(x_{test})$$

$$= \sum_{i=1}^{n} \alpha_i^* k(x_i, x_{test})$$

Page No:
Date:

$k(x_i, x_{test})$ → KERNEL FUNCTION (how similar is x_{test} to x_i)

α_i^* → How important is i^{th} point towards w^*

Basically, we have managed to formalize our linear regression problems to solve for non-linear regression as well.

Page No:
Date:

PROBABILISTIC VIEW OF LINEAR REGRESSION

Dataset : $\{(x_1, y_1), \ldots\ldots, (x_n, y_n)\}$

$x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$

Let us assume : $Y/x \sim W^T x + \epsilon$

- Y/x : The label given the data point is generated as
- $W^T x$: W unknown but fixed, $\in \mathbb{R}^d$
- ϵ : Noise, $N(0, \sigma^2)$ ↳ Gaussian

WHAT EXACTLY THE ABOVE EQUATION MEANS ?

- Every y_i was generated according to this process.
- Somebody gave us x_i & then to get the y_i , there is an unknown but fixed W using which $W^T x_i$ was generated & then a noise got added and we are only seeing the noisy version of $W^T x_i$ whereas we know that the statistics of this noise is 0 mean and some known variance σ^2.

→ Can view this as an 'ESTIMATION PROBLEM' we are trying to estimate the W which after adding noise affects our labels.

→ Solution Approach : MAXIMUM LIKELIHOOD

NOISE STATISTICS → Density of the noise impacts
↳ LOSS FUNCTION

Page No:
Date:

Likelihood Function

$$L\left(w ; \begin{matrix} x_1, \ldots, x_n \\ y_1, \ldots, y_n \end{matrix}\right) = \prod_{i=1} e^{\frac{-(w^T x_i - y_i)^2}{2\sigma^2}} \cdot \frac{1}{\sqrt{2\pi}\sigma}$$

$$\log L\left(w, \begin{matrix} x_1 \ldots x_n \\ y_1 \ldots y_n \end{matrix}\right) = \sum_{i=1}^{n} \frac{-(w^T x_i - y_i)^2}{2\sigma^2} \cdot \frac{1}{\sqrt{2\pi}\sigma}$$

equivalently,

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

$$\equiv \min_w \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

This is exactly the linear regression problem with squared error that we already put on, which means we know the solution to this.

$$\hat{w}_{ML} = w^* = (xx^T)^\dagger xy$$

CONCLUSION : Maximum Likelihood Estimator assuming 'ZERO MEAN GAUSSIAN NOISE' is same as 'LINEAR REGRESSION WITH SQUARED ERROR'

- We are saying that if we chose squared error to solve the linear regression problem, it is as if we are implicitly making the assumption that there is a Gaussian noise 0 mean that gets added to our labels that corrupts our labels.

Choosing Noise means implicitly choosing ERROR FUNCTION.