Introduction to Machine Learning
by
Prof. Preethi Jyothi
CS 419 M, IITB

https://somphene.github.io/notes/

Notes by
Som S. Phene

Spring 2020

A visitor remarked that the library at the newly constructed IHES was lacking. Grothendieck replied "We don't read books, we write them."

This is an **incomplete draft**. Please send corrections, comments, pictures of bad drawings, etc. to `somphene1@gmail.com`.

Last updated January 24, 2020.

Contents

1 Lecture Jan 10

Introduction. To be updated (later).

2 Jan 17, Lecture 2

Today's lecture on Linear regression.
Content:

- Linear regression
- Least squares objective: Closed form solution (which is an exception).
- (later)

Moodle will be exhaustive for resouces. Slides, refresher for lin alg and probability are also posted. Support for python. Office hours post 7:30 for TAs. Prof's office hours 4-5pm.
Project partners via Moodle or post.

First assignment will be released after three lectures. Purely programming based. Tutorial session will be held for the same. Quiz at the beginning of Feb.

Question 2.0.1 (How should I sell my laptop)**.** Collect some data for what should be the price if I sold it on OLX,Amazon,.... Data set:

#of cores	Price
1	35000
2	40000
1	28000

Answer: Regression fit to the data set. $y_i = w_0 + w_1 x_i + \epsilon$, where *epsilon* is the inherent noise in the data (coming from stochasticity in the measurement process, measurement process)- that is irresducible noise. Maybe actually good so that we don't overfit the data. probabilistic interpretation in which we treat the noise to be a random variable coming from some stochastic process. Insert diagram (later).

§2.1 Regression

- Curve fitting (**regression**) is the process of constructing a function, that has best fit of a series of data points, possibly subject to constraints (Wikipedia).

Example 2.1.1 (Laptop selling with higher dimensional data)
If the RAM is also considered in the data points, the best fit would become a plane (hyperplane in higher dimensions) in $\mathbb{R}^3$. The function takes the form $f = w_0 + w_1 x_1 + w_2 x_2$.

§2.2 Evaluation Function

$E(f, \mathcal{D})$
$\mathcal{D} = \{(x_i, y_i)\}^N, \quad x_i \in \mathbb{R}^{d+1}, y_i \in \mathbb{R}$

Question 2.2.1. Isn't Zero error overfitting?

Answer: Yes, will be covered in the next lecture after it is defined.

Example 2.2.2 (Absolute value evaluation function)

$$E(f, \mathcal{D}) = \sum_{i}^{N} |y_i - f(x_i)|$$

Not good for optimization purposes because its not a differentiable function so loses a nice mathematical property.

Example 2.2.3 (Method of Least Squares)

$$E(f, \mathcal{D}) = \sum_{i}^{N} (y_i - f(x_i))^2$$

(L_1 loss) Good for optimization purposes because its a differentiable function so has a nice mathematical property. Disadvantage that it magnifies the error because of squaring.

error function: assumed to be least squares from now.

Definition 2.2.4 (Linear regression)**.** f comes from space of linear functions $f_w(x) = w^T x$, which is the inner product that can be written explicitly as:

$$f(x; w) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_d x_d$$

Learning theory: empirical risk related to this quantity.

§2.3 Simplest case of regression over 1-D data

Given a 1-D training instances, $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$

Goal: Find $w_0^*, w_1^* = \arg\min_{w_0, w_1} \sum_{i=1}^N (y_i - w_1 x_i - w_0)^2$, let $L(w)$ denote the sum to be minimized.

$$\frac{\partial L}{\partial w_0} = -2 \sum_i (y_i - w_0 - w_1 x_i) = 0$$

$$\implies -\sum_i (y_i + w_0 - w_1 x_i) = 0$$

(later)

§2.4 Same thing over 2D data

§2.5 Linear regression for d-dimensional data

Find $w_0^*, w_1^* = \underset{w}{argmin} \sum_{i=1}^{N}(y_i - w^T x_i)^2, \quad y_i \in \mathcal{R}, x_i \in \mathcal{R}^{d+1}$, let $L(w)$ denote the sum to be minimized.

$$\nabla_w L(w) = \left[\frac{\partial L(w)}{\partial w_0}\frac{\partial L(w)}{\partial w_1}\frac{\partial L(w)}{\partial w_2}...\frac{\partial L(w)}{\partial w_d}\right] T = -2\sum_i (y_i - w^T x_i)x_i = 0$$

$$\implies -2\sum_i y_i x_i + \sum_i (w^T x_i)x_i) = 0$$

Rewrite equation using the following vectors/matrices:

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1d} \\ 1 & x_{21} & x_{22} & \dots & x_{2d} \\ 1 & & \dots & . & \\ 1 & x_{n1} & x_{n2} & \dots & x_{nd} \end{bmatrix}$$

is a $N \times (d+1)$ It is possible that the matrix is not invertible (if the rank is not full). **Moon-Penrose Pseudo Inverse**. Cumbersome to compute the closed form solution: instead go through Gradient descent.

§2.6 Generic regression model with basis functions

For polynomials, enough to get coefficients of the base polynomials

$$f_w(x_i) = w_0 + w_1 x_i + w_2 x_i^2 + \ldots + w_d x_i^d$$

For generic basis functions:

$$f_w(x) = w_0\phi_(x) + w_1\phi_1(x_) + w_2\phi_2(x) + \ldots + w_d\phi_d(x)$$

$\phi_j(x)$ is a basis function Only finite dimensional basis functions are used here, but for infinite dimensional, we will use kernels (to be introduced in further examples).

§2.7 Geometric interpretation

Orthogonal projection onto the column space of X.
Next class: gradient descent. Probabilistic approach. Logistic regreesion, decision trees(nonlinear) in the first assignment. Then come back to linear models. Perceptrons SVN and all. Gradient descent will show up again when deep neural networks, Probabilistic when variational.

3 Lecture Jan 22

§3.1 Linear Regression

Recall from last class: Linear regression. Least squares regression. Obtained the closed form solution. The matrix need not be full ranked for inversion- "Moore-Penrose PseudoInverse". However, the inversion takes a lot of time.

§3.2 Gradient Descent

> **Question 3.2.1.** Is there a better way to approximate the inversion procedure and get the inverse quickly, which also scales well with the size of the matrix.

Definition 3.2.2. Gradient Descent: Iterative optimization algorithm that generates (modulo constants) convergence to a local optimum.
Algorithm: GD
$W \leftarrow W^0$
While not converged;
$W^{t+1} \leftarrow W^t - \eta \nabla_w L(w^t)$
return w
Where $L(w)$ is Weight update rule. η is the learning rate (hyperparameter). Refer to figure 3.2
If learning rate is set too large, then we may miss out on the optimal value. Gradient descent guarantees convergence to local minimum.

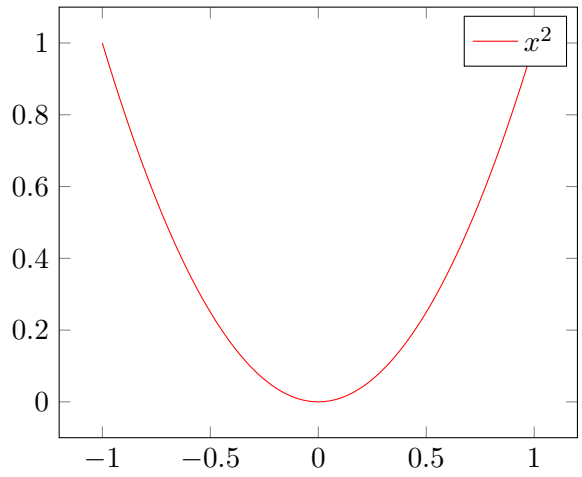

Figure 3.1: Example showing Gradient Descent

> **Question 3.2.3.** In fact the loss function may have non-isolated local minima, then how do we reach the minimum that we want?

Restrict to convex functions! Then we have a unique optimal point.

Definition 3.2.4. $f : x \to \mathbb{R}$ is convex if $\forall x_1, x_2 in X, \alpha \in [0, 1]$

$$f(\alpha x_1 + (1-\alpha)x_2) \leq \alpha f(x_1) + (1-\alpha) f(x_2)$$

For instance, figure 3.2 is a convex function. Least squares is also a convex funxtion.

While not converged, we can add a **Stopping criterion**:

1. $\|\nabla_w L(w)\|_2 \leq \epsilon$
2. $\|L(w^{t+1}) - L(w^t)\|_2 \leq \epsilon$

$$\nabla_w L_{MSE}(w) = \begin{bmatrix} \frac{\partial L(w)}{\partial w_0} \\ \frac{\partial L(w)}{\partial w_1} \\ \vdots \\ \frac{\partial L(w)}{\partial w_d} \end{bmatrix} = -2 \sum_i (y_i - w^T x_i) x_i$$

§3.2.i Variant of Gradient descent: Stochastic Gradient Descent

Find the gradient corresponding to the loss function of the i^{th} sample. Making a gittery movement for every sample: might throw off a convergence due to one outlier point. On average, theorem is that the Expected risk for Stochastic GD are equal to the risk of GD. SGD Uses Single Sample.

Variable learning rate

Strategy: Start with initial learning rate η to be higher. As you progress, decay the magnitude of η depending on the progress of the descent. Track progress using behaviour on examples: if the labels saturate, we have good progress.
Some comments on curriculum learning example.

§3.2.ii Mini-batch GD

Choose a batch-size b. For $b = 1 \implies$ SGD, when $b = d$ GD.

for $i \in 1, b+1, 2b+1, \ldots$
$w \leftarrow w - \eta \nabla_w L^{i,b}(w)$
Where $L(w) = \sum_{t=i}^{i+b-1} (y_i - w^T x_t)^2$

§3.3 Probabilistic Interpretation of LR

For a dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$, let the target y_i's come from an underlying probabilistic distribution.
Let us assume a "noisy" linear model

$$y_i = w^T x_i + \epsilon_i$$

Where ϵ_i is stochastic noise modelled as a Random Variable (RV).
Let $\epsilon \sim \mathcal{N}(0, \sigma^2)$
(If you want a primer for Linear Algebra: Gilbert Strang's 18.065 Linear Algebra for ML lectures videos are up on MIT OCW. For instance geometric interpretation of least squares optimal solution as the orhtogonal projection.)
If $y_i = w^T x_i + \epsilon_i$ and $\epsilon \sim \mathcal{N}(0, \sigma^2)$, then $y_i \sim \mathcal{N}(w^T x_i, \sigma^2)$

Question 3.3.1. Determine w from that prob. distribution.

Solution: ***Maximum Likelihood Estimate (MLE)*** If $y = \{y_1, y_2, \ldots, y_n\}$ are the independent ans identically distributed (i.i.d) samples drawn from a prob. dist. parameterized by $P(y|w)$ or $P_w(Y)$,
then $W_{MLE} = \text{argmax}\, P(y|w) = \text{argmax} \prod_i P(y_i|w)$
Where $P(y|w) = \text{argmax} \prod_i P(y_i|w)$ is called the **Likelihood** and $logP(y_i|w)$ is called the **Log-Likelihood**.

$$W_{MLE} = \underset{w}{\text{argmax}} \sum_i \underbrace{logP(y_i|w)}_{Log-likelihood}$$

Taking Log doesnt affect maxima as it is a monotone function. To compute MLE, one typically uses log probabilities. Helps to find the derivative as well (no product rule required). Also helps to reduce computational cost as high values don't saturate.

Question 3.3.2. Say you toss a coin N times. Each coin toss y_i,

In class Quiz:

Question 3.3.3. Let y be generated, conditional on x, by the following process: $\epsilon \sim \mathcal{N}(0, \sigma^2).y = ax + \epsilon$

$$P(y|x, a) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(y - ax)^2}{2\sigma^2}\right\}$$

What is the ML estimate of the parameter a. You are given a dataset
$\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$

Answer is

$$a = \frac{\sum_i x_i y_i}{\sum_i x_i^2}$$

4 Lecture Jan 24

Last class we looked at Gradient descent. Move opposite to the gradient. Guaranteed to converge to local optimum.
Substitute the expensive task of computing the gradient by taking one sample and using it to compute the gradient. However on average, stocahstic gradient descent is the same as GD. Mini-batch: work on a small batch of the training set. Started talking about probabilistic interpretation. Slides will be updated with details over the weekend. Maximize the likelihood (MLE). MLE for n coin tosses case. Building up to linear regression.

§4.1 Probabilistic view of LR

Lets assume a linear model

$$y_i = w^T x_i + \epsilon_i$$

Where ϵ_i is stochastic noise modelled as a Random Variable (RV).
Let $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Recall from last time that:
If $y_i = w^T x_i + \epsilon_i$ and $\epsilon \sim \mathcal{N}(0, \sigma^2)$, then $y_i \sim \mathcal{N}(w^T x_i, \sigma^2)$

Question 4.1.1. Determine w from that prob. distribution. What is the ML estimate of the parameter? You are given a dataset
$\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$

$$Ll(w) = \Pi_i P(y_i \,|\, x_i w)$$

$$\begin{aligned} Ll(w) &= \Pi_i \mathcal{N}(w^T x_i, \sigma^2) \\ &= \Pi_i \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{ -\frac{(y_i - w^T x_i)^2}{2\sigma^2} \right\} \\ \underbrace{LL(w)}_{log-likelihood} &= constant - \sum_i \frac{(y_i - w^T x_i)^2}{2\sigma^2} \end{aligned}$$

$$W_{MLE} = \underset{w}{\text{argmax}} LL(w) = \underbrace{\underset{w}{\text{argmin}} \sum_i \sum_i (y_i - w^T x_i)^2}_{Least-squares-objective}$$

§4.2 Errors

Three sources of error: Bias, Variance and Noise (irreducible).

- Bias/Variance tradeoff
- Bias/Variance decomposition in regression
- Prediction errors

We can't bridge the gap for Prediction error. Have to live with it. The other two errors related to bias/variance are more to do with our models. We have control over them.

§4.3 Prediction errors

Insert diagram (later). This is a low complexity model, so its highly biased. In learning theory, we will formalize the concept of complexity.

§4.3.i Bias/Variance tradeoff

Identical training data, choose a different size of sample points. It is possible that this choice affects the fitiing curve significantly. Over all samples, the mean predictor varies smoothly. How much the fitting graph hovers around is captured by the variance. For a high complexity model, the variance is high.

$$\text{High complexity} \implies \text{high variance} \tag{4.1}$$
$$\implies \text{low bias} \tag{4.2}$$

whereas

$$\text{Low complexity} \implies \text{low variance} \tag{4.3}$$
$$\implies \text{high bias} \tag{4.4}$$

Its clear that there's a **Bias/Variance tradeoff**. It's our choice to make the model to mimimize bias or variance. We can also tune hyperparameters to get to the sweet spot of complexity of the model after which the bias suffers at the expense of variance or vice-versa. In fact Error = bias^2 + variance. Lets see why.

§4.3.ii Bias/Variance decomposition in regression

Suppose we have an (x, y), where $y = f(x) + \epsilon$, where ϵ is Gaussian noise with 0 mean and σ^2 variance.
Let $h(x) = w^T x$, where w is determined by minimizing the least squares error on a training set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$.
For an X (with an underlying y), the expected prediction error $= E[y - h(x)^2]$

$$E[y - h(x)^2] = \underbrace{E[h(x)^2]}_{①} + \underbrace{E[y^2]}_{②} - \underbrace{2E[y]E[h(x)]}_{③}$$
$$①E[h(x)^2] = E[(h(x) - \bar{h}(x))^2] + Barh(x)^2$$
$$②E[y] = E[f(x) + \epsilon] = f(x)$$
$$E[y^2] = E[(y - f(x))^2] + f(x)^2$$
$$E[(y - h(x))^2] = E[(h(x) - \bar{h}(x))^2] + \bar{h}(x)^2 + E[(y - f(x))^2] + f(x)^2 - 2f(x)\bar{h(x)}$$
$$= \underbrace{[h(x) - \bar{h}(x))^2]}_{variance} + \left(\underbrace{\bar{h}(x) - f(x)}_{Bias}\right)^2 + \sigma^2$$

Here $\bar{h}(x)$ is the mean parameter. We have used: **Corollary**: $E[x^2] = E[(x - \bar{x})^2] + \bar{x}^2$

Overfitting and Underfitting

Model capacity limitations: only som many degrees of freedom to estimate the parameters. As we increase the complexity, training error will tends to zero. Test error: evaluate

predictor on unseen samples. If the model is too good a fit for the training samples (high complexity model) then for an unseen sample that is different from the training data, the error is large. include diagram (later).
Overfitting: Setting where the training error is low but test error is high.

How to counter Overfitting

> **Question 4.3.1.** What can we do to counter model complexity?

- Reduce the model complexity
- For a fixed model complexity. Increase the number of training samples. Variance reduces.
- Reduce the number of features. **Feature selection, Dimensionality reduction**

Next class setup for Overfitting and underfitting in Linear Regression.