Gaussians, Logistic Regression, and Naive Bayes

Vijay Jaisankar

Teaching Assistant

May or may not have made these slides at 3 AM

Agenda

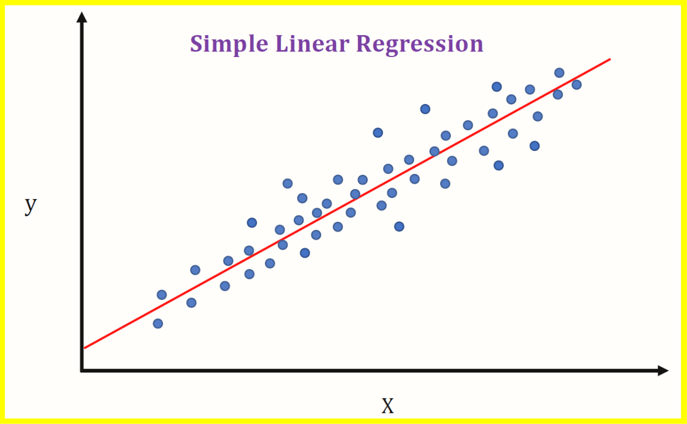
- Revision of last class
- Introduction to classifiers
- The Gaussian function
- Maximum likelihood estimation (MLE)
- Bayes' theorem and the Naive Bayes classifier
- Non-linearity and the Logistic regression algorithm

Recap of last class

The components of an ML system

- Tasks == Problems you wish to apply Machine Learning on; clear declaration and definition of inputs and outputs
- Models == Algorithms run on data that generate insights
- Features == Filtered and Processed Inputs
- Datasets == "Raw" Data

Linear Regression - Essence



Gradient Descent - Essence

- Output
- Costs
- Update Weights

Gradient Descent - Algorithm

Algorithm 1 Gradient Descent

```
W ← random
Costs ← φ
for i = 1 to n_i do
    Ŷ ← M(W, X)
    C ← J(Y, Ŷ)
    W ← W − α∇_W C
    Append C to Costs
end for
```

$W \leftarrow \text{random}$

$\text{Costs} \leftarrow \phi$

for $i = 1$ to n_i **do**

$\quad \hat{Y} \leftarrow M(W, X)$

$\quad C \leftarrow J(Y, \hat{Y})$

$\quad W \leftarrow W - \alpha \nabla_W C$

$\quad$ Append C to Costs

end for

Introduction to classifiers

Classification

- (Mostly) supervised setting
- Features == Inputs
- Labels == Outputs

Noob Classifier

```python
def noob_classifier(features_list, labels_list):
    return labels_list[0]
```

Do you see anything wrong with this?

Things are rarely uniform!

- This can perform crazily well in certain cases!
- But, we can all agree that this might not be a good idea. *Why?*
- Does not scale well - **Under-fitting**
- To create more of an even class distribution, we perform *Over-sampling* or *Under-sampling* or more specific *Hyperparameter tuning*.

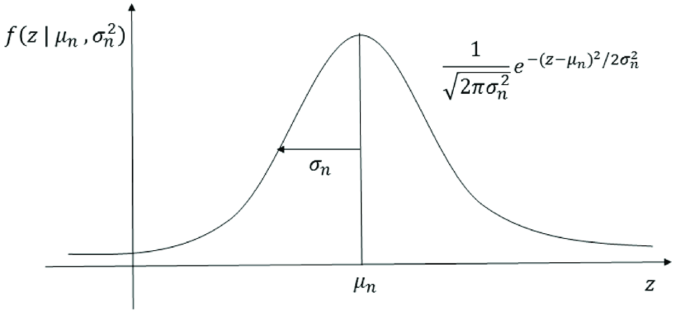
Gaussians

Gaussians are everywhere!

There are three things in IIITB that you will always encounter:

- Assignments
- The appearance of Gaussians in your problems
- Caches and parallelism

Gaussian - Diagram



Gaussian curve - Essence

Let's look at the curve again

- There appears to be a "center point"
- The values seem to spread out
- The "heights" of the values that are far away from the center point are lower
- **Relative Spread** - if a random value is extracted from a sample that follows this curve, there is *high probability* that it belongs to the "middle band". *Just how probable? Look at the equation!*

Fitting a Gaussian - Essence

If we can safely predict that the input data follows a Gaussian curve, what parameters do we need to define the data?

- All the input points?
- The corresponding μ and σ values?

We can represent the data with only the Gaussian Parameters! $\rightarrow$ Saves data

Fitting a Gaussian - The big question

Okay, so we've decided to reduce our data into a Gaussian.
What do we need to represent it? μ and σ
How do we *estimate* these parameters?

Parameter estimation - the big idea

- For any candidate parameter, we can associate a **Likelihood function** - "support" provided by the input data for the given parameter
- In more technical term, the Likelihood function is a **Joint CDF/PDF**.
- So, to find the "right" parameter, it needs to be a **maxima** of the Likelihood function. *Have we done this before?*

Maximum Likelihood Estimation

Why log?

Which one is easier to differentiate?

- $f_1(x) \cdot f_2(x) \cdot f_3(x) \cdot ...$
- $f_1(x) + f_2(x) + f_3(x) + ...$

Also, $log(f_1(x) \cdot f_2(x) \cdot f_3(x) \cdot ...) =$
$log(f_1(x)) + log(f_2(x)) + log(f_3(x)) + ...$

Note: Perform these stunts under the supervision of Convex functions.

Univariate Gaussian MLE - Results

Theorem: Let there be a univariate Gaussian data set $y = \{y_1, \ldots, y_n\}$:

$$y_i \sim \mathcal{N}(\mu, \sigma^2), \quad i = 1, \ldots, n \, .$$

Then, the maximum likelihood estimates for mean μ and variance σ^2 are given by

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} y_i$$
$$\hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} (y_i - \bar{y})^2 \, .$$

MLE - Recap of Steps

- Write a Likelihood function of inputs and parameters (L)
- Under suitable assumptions, take its logarithm (l)
- Under suitable conditions, solve for the **parameter with maximum likelihood** (*Differentiate and equate to 0*) $\rightarrow$ Best θ

Univariate Gaussian: Link

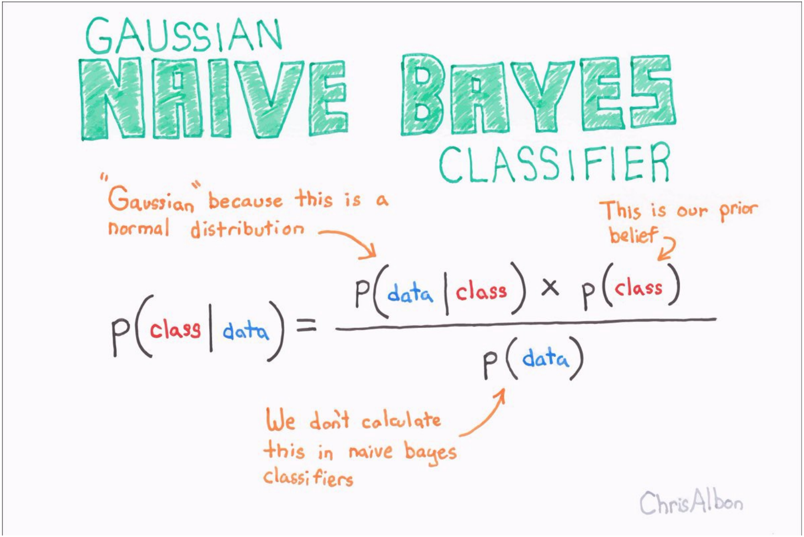
Naive Bayes Classifier

Recap - Problem Setting

- Input vector x
- Set of K classes $C_1, ..., C_K$
- We need to find k where $p(C_k|\mathbf{x})$ is maximised (*Class k has the highest probability of accommodating x*)

- Let's assume that all features are independent of each other.
- Let's assume that each feature follows a Gaussian Distribution.
- The result? From our previous work, we now have a way of calculating $p(\mathbf{x}|C_k)$!

Gaussian Naive Bayes: Putting it all together



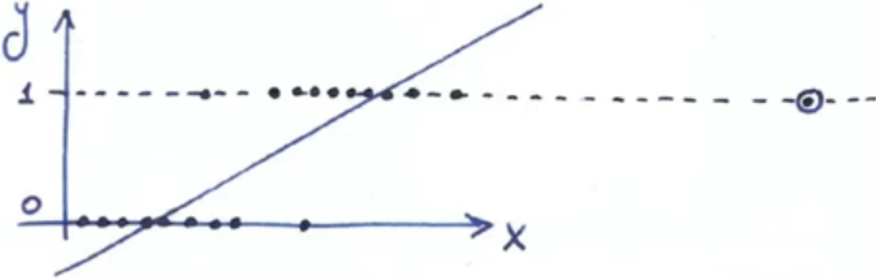
- If we have all $p(C_k|\mathbf{x})$s, how do we find the right k for $\mathbf{x}$? **Ans: argmax**
- Why don't we need to compute $p(data)$? **Ans: It's just a proportionality constant and is positive.**

Generalising this model

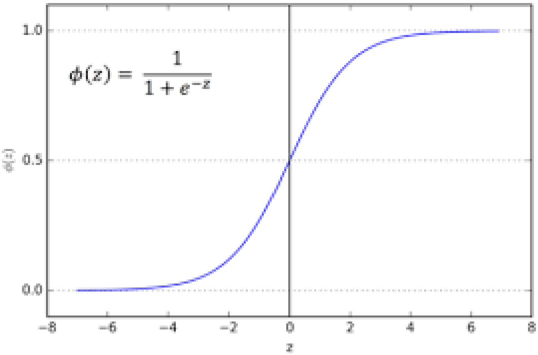
- Can the input features come from other distributions?
- What if the features are not independent?

Logistic Regression

Why not use Linear Regression?



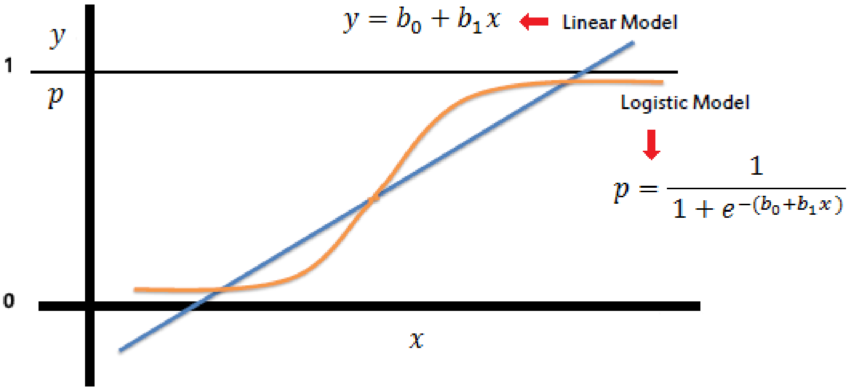
Adding non-linearity: The sigmoid function



Logistic Regression - Essence

- Much like Linear regression, we learn a line.
- We feed this line into the sigmoid function to get a value between 0 and 1.
- We then threshold this value to a particular class (0 or 1).

Logistic Regression - Essence



How does this approach differ from the Naive Bayes model we just saw? Ans: We're calculating $p(y|x)$ directly!

MLE for Binary Classification

Bernoulli random variable: $Y = 1$ with probability p and $Y = 0$ probability $1 - p$.

The likelihood:

$$\prod_{i|y_i=1} h(\mathbf{x}_i) \cdot \prod_{i|y_i=0} (1 - h(\mathbf{x}_i)).$$

The negative log-likelihood:

$$\begin{aligned}\mathcal{L} &= - \sum_{i|y_i=1} \log h(\mathbf{x}_i) - \sum_{i|y_i=0} \log (1 - h(\mathbf{x}_i)) \\ &= - \sum_i \left[y_i \log h(\mathbf{x}_i) + (1 - y_i) \log (1 - h(\mathbf{x}_i)) \right].\end{aligned}$$

Loss Function for Binary Classification

The loss function is

$$\mathcal{L} = -\sum_i \left[y_i \log h(\mathbf{x}_i) + (1 - y_i) \log (1 - h(\mathbf{x}_i)) \right]$$

where

$$h(\mathbf{x}) = \frac{1}{1 + e^{-\beta^\top \mathbf{x}}}.$$

- Sigmoid $==$ Probability of Class "1"
- MLE through Bernoullian analysis
- Gradient Ascent Algorithm by taking the gradient of the loss

Thank you