Classification II

Lecture 6

Manuel Balsera

IEOR, Columbia University

March 1, 2019

Outline

Introduction

Today we will work on removing the **independence** assumption of Naive-Bayes.
We will discuss three classifiers

QDA Quadratic Discriminant analysis, a very strighforward generalization of Gaussian Naive Bayes

LDA Linear Discriminant Analysis, a modification of QDA that requires less parameters.

Logistic Regression One of the workhorses of classification.

References for Today

- Chapter 4 in ISL [1] covers LDA,QDA and Logistic Regression.
- A detailed derivation of the gradient in multi-class logistic regression.
- The section on feature selection of Chapter 13 from [2] for χ^2 test.

Generative vs Discriminative Models

The goal of a classification algorithm is to learn the conditional probability distribution

$$p(y|x) \tag{1}$$

There are two main approaches to it

Generative Learn P(y,x) and use Bayes theorem to infer

$$p(y|x) = \frac{P(y,x)}{p(x)} = p(x|y)\frac{p(y)}{p(x)} \tag{2}$$

If we just want to pick up the most likely class

$$\hat{y} = \arg\max_y p(x|y)\frac{p(y)}{p(x)} = \arg\max_y p(x|y)p(y) \tag{3}$$

because $p(x)$ is just a normalization constant that affects all classes equally.

Discriminative Postulate a functional for $p(y|x)$ directly and learn that.

Generative vs Discriminative Models II

- **K-Nearest Neighbors** is a discriminative model

$$p(y = k|x) = \frac{\hat{N}_k(x)}{K} \tag{4}$$

- **Naive Bayes** is a generative model

$$\log p(y = k|x) \approx \sum_d \log p(x_d|y = k) + \log p(y = k) \tag{5}$$

- Generative models make **stronger assumptions** that strictly necessary (the distribution of $p(x)$)
- If the assumptions are justified, they learn both $p(y|x)$ and $p(x)$.
- If the assumptions are not justified, they will have higher bias.
- If the number of parameters required to estimate $p(x)$ is large, they may also have higher variance.

Gaussian Naive Bayes Example

Recap from last lesson

The following two examples have $N = 40$ data points with $D = 2$.

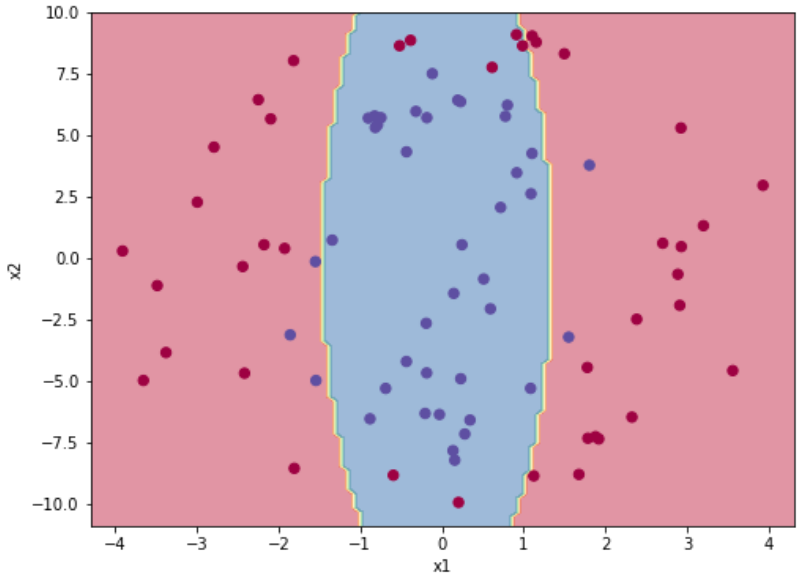

Figure 1: x_1 and x_2 are independent. Accuracy $\approx 78\%$

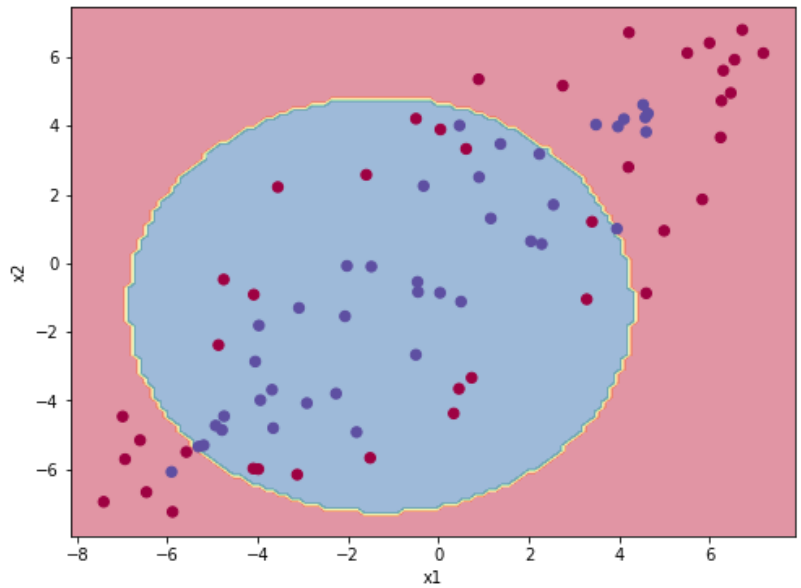

Figure 2: x_1 and x_2 are dependent. Accuracy $\approx 68\%$

As shown in the lecture on **Naive Bayes** NB Gaussian assumption for x_d results in a quadratic decision boundary **aligned** with axis.

Generalizations to Gaussian Naive Bayes

The natural extension **as a generative model** is known as Quadratic Discriminant Analysis (QDA) and assumes

QDA

$$P(x|y=k) = \mathcal{N}(x;\mu_k,\Lambda_k) = \frac{1}{\sqrt{(2\pi)^D |\Lambda_k|}} e^{-\frac{1}{2}(x-\mu_k)^T \Lambda_k^{-1}(x-\mu_k)} \tag{6}$$

a multivariate Gaussian distribution for $x \in \mathbb{R}^D$ with $\mu_k \in \mathbb{R}^D$ and Λ_k a $D \times D$ covariance matrix.
Gaussian Naive Bayes is just a special case where Λ_k is a diagonal matrix

$$\Lambda_{k,d,d',k} = \sigma^2_{k,d}\delta_{d,d'} \tag{7}$$

($\delta_{d,d'}$ is the Kronecker delta, $\delta_{d,d'} = 1$ if $d = d'$ and zero otherwise)
The QDA model assumes that x is distributed as a **mixture of Gaussians**

$$P(x) = \sum_k P(x|y=k)p(y=k) = \sum_k \mathcal{N}(x;\mu_k,\Lambda_k)\pi_k \tag{8}$$

Mixed Gaussian Example

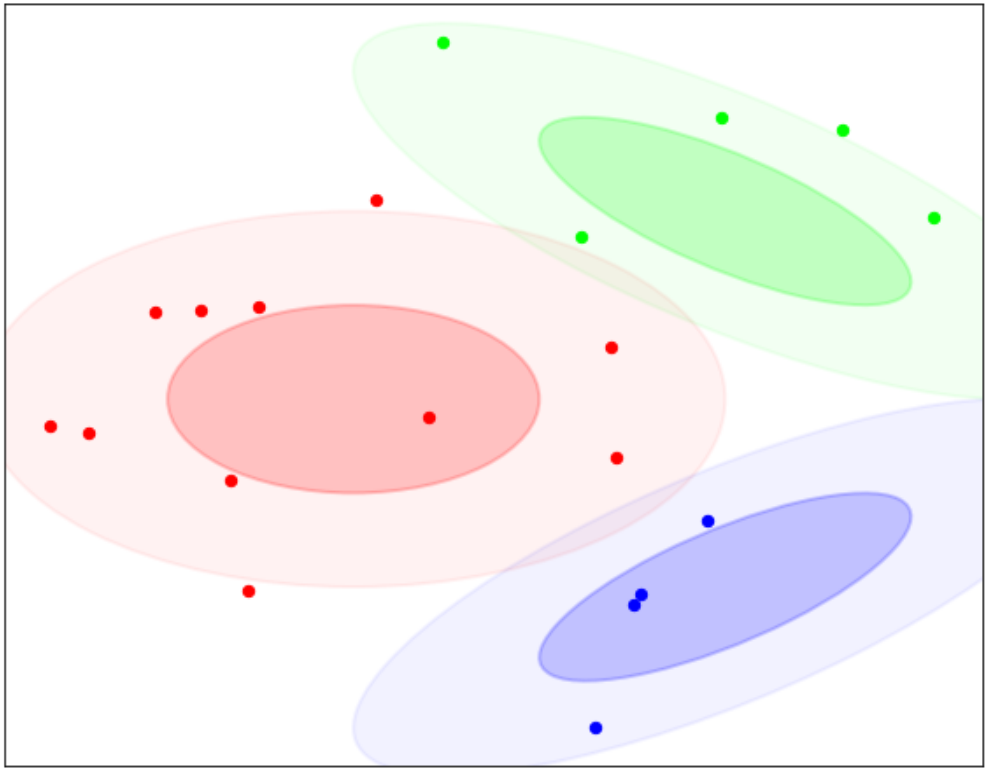

Figure 3: N=20 data samples of mixed Gaussian Distribution with K=3 classess and D=2.

QDA

The maximum likelihood estimates of the parameters, in terms the input data x, the class counts $\hat{N}_k$ and the one-hot encoding of $z_{i,k}$ of the labels is:

$$\pi_k = \frac{\hat{N}_k}{N} = \frac{1}{N}\sum_i z_{i,k}$$

$$\mu_{d,k} = \frac{1}{\hat{N}_k}\sum_i x_{i,d} z_{i,k}$$

$$\Lambda_{d,d',k} = \frac{1}{\hat{N}_k}\sum_i (x_{i,d} - \mu_{d,k})(x_{i,d'} - \mu_{d',k}) z_{i,k} \tag{9}$$

Lets compare the number of parameters estimated and compare to Naive Bayes

Model	π	μ	Λ
Naive-Bayes	$K-1$	$K \times D$	$K \times D$
QDA	$K-1$	$K \times D$	$K \times D \times \frac{D+1}{2}$

If D is large that can be a lots more parameters to estimate!

QDA Gaussian Estimates

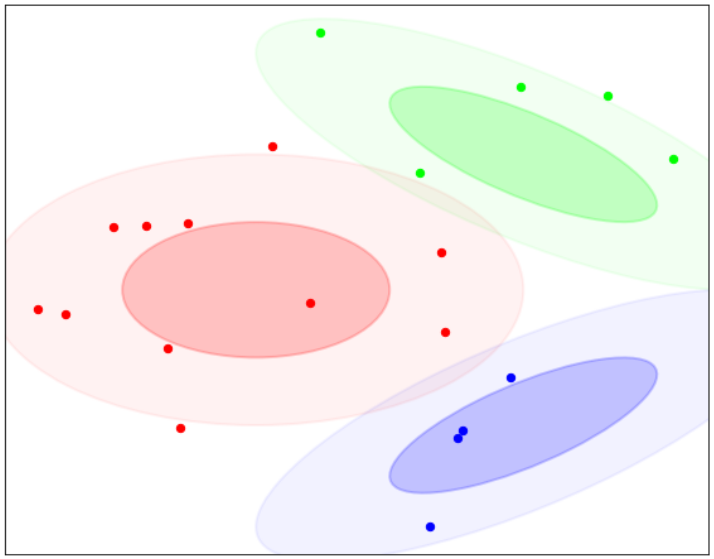

Figure 4: True Gaussian Mixture Distrution

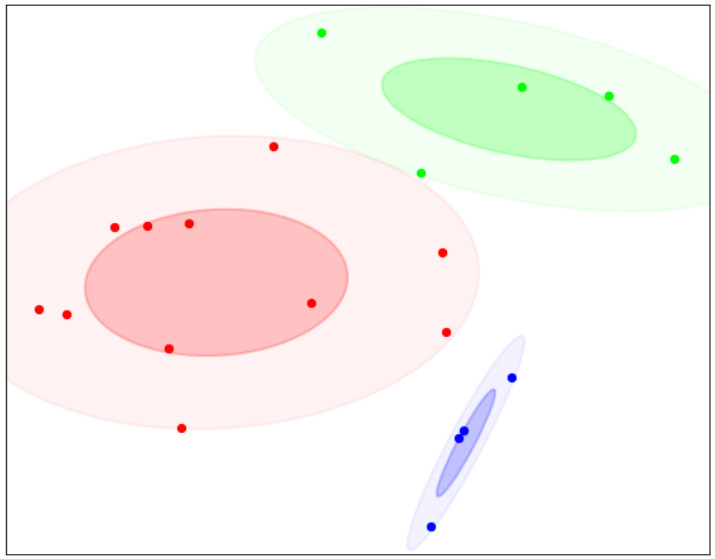

Figure 5: QDA estimate of the Gaussian Mixture Parameters.

Here $D = 2$, $K = 3$ and there are a total of 17 parameters estimated from $N = 20$ samples.

- π $K - 1 = 2$ parameters.
- μ $K \times D = 6$ parameters.
- Λ $K \times D \times \frac{D+1}{2} = 9$ parameters

QDA Boundary

At the classification boundary between two classess k and k' they have the same loss ($L = -\log P$)

$$L_k - L_{k'} = 0 = \frac{1}{2}(x-\mu_k)^T \Lambda_k^{-1}(x-\mu_k) - (x-\mu_{k'})^T \Lambda_{k'}^{-1}(x-\mu_{k'}) + \frac{1}{2}\log\frac{|\Lambda_k|}{|\Lambda_{k'}|} - log\frac{\pi_k}{\pi_{k'}} \tag{10}$$

Provided $\Lambda_k \neq \Lambda_{k'}$ this is a quadratic equation on x

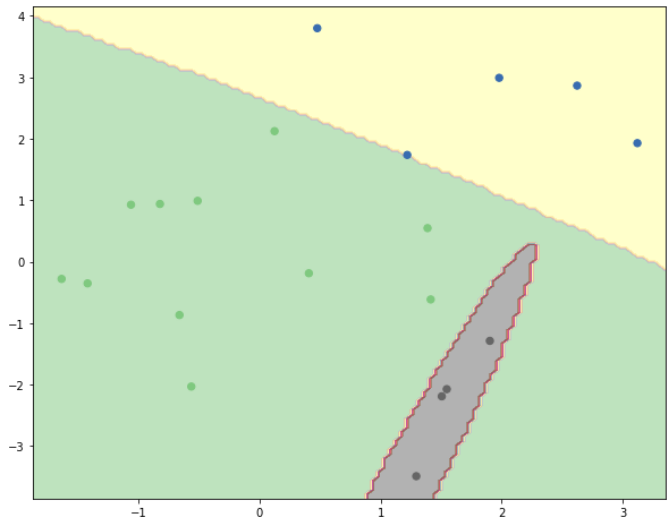

Figure 6: QDA decision Boundary on a D=2, K=3 classification problem. $\approx$ 85% accuracy.

LDA

We can reduce the number of parameters by setting al covariances the same $\Lambda_k = \Lambda$

$$\pi_k = \frac{\hat{N}_k}{N}$$
$$\mu_{d,k} = \frac{1}{\hat{N}_k} \sum_i x_{i,d} z_{i,k}$$
$$\Lambda_{d,d'} = \frac{1}{N} \sum_k \sum_i (x_{i,d} - \mu_{d,k})(x_{i,d'} - \mu_{d',k}) z_{i,k} \tag{11}$$

Lets compute the number of parameters estimated and compare to Naive Bayes

Model	$\boldsymbol{\pi}$	$\boldsymbol{\mu}$	$\boldsymbol{\Lambda}$
Naive-Bayes	K	$K \times D$	$K \times D$
QDA	$K-1$	$K \times D$	$K \times D \times \frac{D+1}{2}$
LDA	K	$K \times D$	$D \times \frac{D+1}{2}$

If D is large that can **still** be a lots more parameters to estimate!

LDA Gaussian Estimates

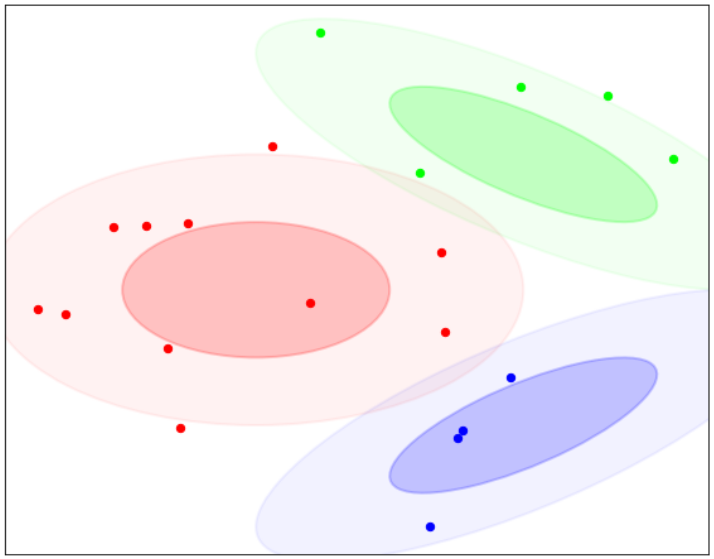

Figure 7: True Gaussian Mixture Distrution

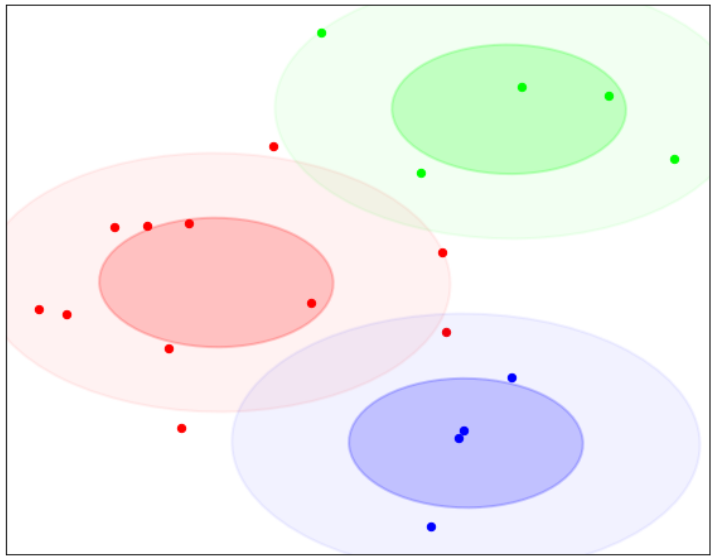

Figure 8: LDA estimate of the Gaussian Mixture Parameters.

There are a total of 11 parameters estimated

- π $K - 1 = 2$ parameters.
- μ $K \times D = 6$ parameters.
- Λ $\times D \times \frac{D+1}{2} = 3$ parameters

LDA Classifier

We will now estimate the functional form of $p(y = k|x)$. Given our assumptions

$$p(y = k|x) = \frac{\frac{\pi_k}{\sqrt{2\pi^D|\Lambda|}} e^{-\frac{1}{2}(x-\mu_k)^T\Lambda^{-1}(x-\mu_k)}}{\sum_{k'} \frac{\pi_{k'}}{\sqrt{2\pi^D|\Lambda|}} e^{-\frac{1}{2}(x-\mu_{k'})^T\Lambda^{-1}(x-\mu_{k'})}} \tag{12}$$

Expanding the quadratic term in powers of x

$$(x-\mu_k)^T\Lambda^{-1}(x-\mu_k) = x^T\Lambda^{-1}x - 2\mu_k^T\Lambda^{-1}x + \mu_k^T\Lambda^{-1}\mu_k^T \tag{13}$$

we see that the quadratic term in x is the same for all k and can be simplified from the equation. Defining:

$$W_{k,d} = \sum_{d'} \Lambda^{-1}_{d,d'}\mu_{k,d'} \qquad \text{(a } K \times D \text{ matrix)}$$

$$b_k = -\frac{1}{2}\mu_k\Lambda^{-1}\mu_k + \log \pi_k \qquad \text{(a K-dimensional vector)} \tag{14}$$

$$\tag{15}$$

we have

$$p(y = k|x) = \frac{e^{\sum_d W_{k,d}x_d + b_k}}{\sum_{k'} e^{\sum_d W_{k',d}x_d + b_{k'}}} \tag{16}$$

LDA Classifier Boundary

The boundary between classes k and k' is set up when their log likelihood agree

$$L_k - L_{k'} = 0 = (W_{k'} - W_k)x + b_{k'} - b_k \tag{17}$$

which is a linear equation as it was the case on the Naive Bayes classifier when we assumed $\sigma^2_{d,k}$ did not depend on k.

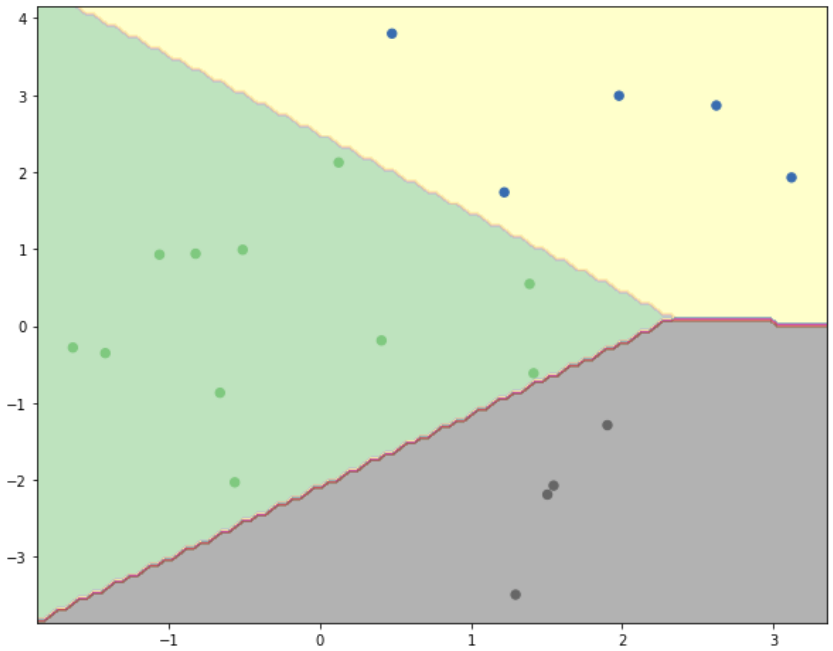

Figure 9: LDA decision Boundary on a D=2, K=3 classification problem. $\approx$ 90% accuracy.

Classifier Comparison

In general, a LDA classifier requires less data to train than a QDA classifier. It may be a good first choice provided the assumption of **linear class boundaries** is approximately correct.

No Free Lunch Theorem

Averaged over all possible data distribution, all classifiers perform the same.
A classifier must make **some assumptions** about the data distribution.

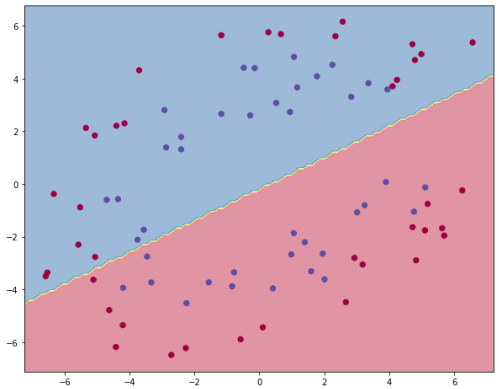

Figure 10: LDA classifier on ellipse training data.
Accuracy $\approx$ 50%

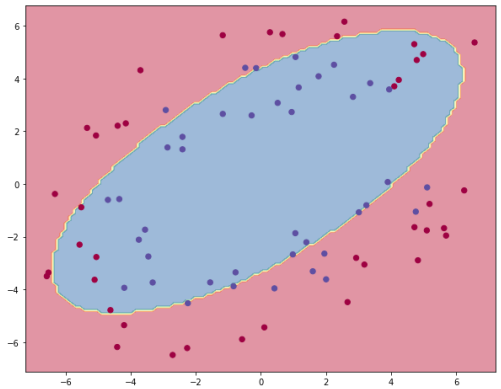

Figure 11: QDA classifier on ellipse training data.
Accuracy $\approx$ 74%

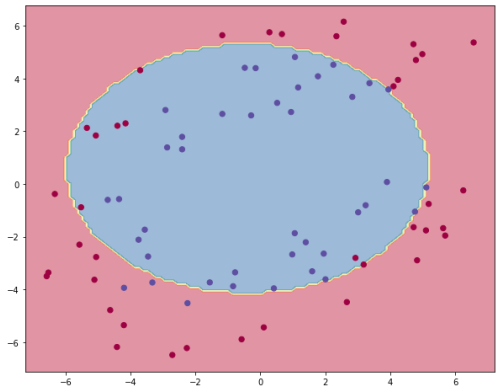

Figure 12: Naive Bayes Classifier on ellipse training data.
Accuracy $\approx$ 85%

LDA as a Generative Classifier

We found that the LDA classifier implies a conditional probability

$$p(y = k|x) = \frac{e^{\sum_d W_{k,d} x_d + b_k}}{\sum_{k'} e^{\sum_d W_{k',d} x_d + b_{k'}}} \tag{18}$$

where, under the **gaussian mixture** assumption

$$W_{k,d} = \sum_{d'} \Lambda^{-1}_{d,d'} \mu_{k,d'} \qquad \text{(a } K \times D \text{ matrix)}$$

$$b_k = -\frac{1}{2} \mu_k \Lambda^{-1} \mu_k + \log \pi_k \qquad \text{(a K-dimensional vector)} \tag{19}$$

(20)

We are estimating

- π K parameters
- μ $K \times D$ parameters
- Λ $D \times \frac{D+1}{2} = O(D^2)$ parameters

But the conditional probability distribution ends up depending only on the combinations

- W $K \times D$ parameters
- b K parameters

If we could estimate directly W and b we will be save $O(D^2)$ parameters!

Logistic Regression as a Discriminative Classifier

Given a categorical variable y taking K classes, the logistic regression classifier **assumes**

Logistic Regression

$$p(y = k|x) = \frac{e^{\sum_d W_{k,d} x_d + b_k}}{\sum_{k'} e^{\sum_d W_{k',d} x_d + b_{k'}}} \tag{21}$$

- Logistic Regression is not really regression (y is categorical) but that is the traditional nomenclature.
- We make no assumptions about the distribution of x.
- Logistic Regression is a **discriminative** classifier.
- We must find W and b by **maximum likelihood**.
- The generative LDA model implies a Logistic Regression functional form, but estimates the parameters differently.
- The generative multinomial naive Bayes classifier also has the logistic regression functional form.

Soft Max Function

The vector-valued soft max function appears naturally in multivariate logistic regression

Soft Max Function

$$\text{softmax}(\eta)_k = \frac{e^{\eta_k}}{\sum_{j=1}^{K} e^{\eta_j}} \tag{22}$$

- for all $k = 1, \ldots, K$, $0 \leq \text{softmax}(\eta)_k \leq 1$
- $\sum_k \text{softmax}(\eta)_k = 1$
- because of the exponentials, $\text{softmax}(\eta)_k \approx 1$ for the k with largest η_k, and close to zero for the others, hence the name softmax
- softmax is a smooth differentiable function, with logarithmic derivative

$$\frac{\partial}{\eta_k} \log \text{softmax}(\eta)_{k'} = \delta_{k,k'} - \text{softmax}(\eta)_k \tag{23}$$

- One of the parameters η_k is redundant, as $\text{softmax}(\eta + \Delta) = \text{softmax}(\eta)$ for any scalar Δ

Categorical Random Variable

Recap From lecture 2

- A categorical variable S can take one of K values with probability

$$P(S = k; \theta) = \theta_k \tag{24}$$

- The categorical distribution is characterized a by K dimensional vector of parameters $(\theta_1, \ldots, \theta_K)$ subject to the constrains

$$\sum_{k=1}^{K} \theta_k = 1; \quad 0 \leq \theta_k \leq 1 \tag{25}$$

- A natural representation of S is as the K-dimensional vector

one-hot-encoding

$$Z(S = k) = (\underbrace{0, 0, \ldots, 0}_{k-1}, 1, \underbrace{0, \ldots, 0}_{K-k}) \tag{26}$$

- with that representation we can write

$$P(S = k) = \prod_{k=1}^{K} \theta_k^{z_k} \tag{27}$$

Softmax and Categorical Distribution

In terms of the `softmax` function, we can the categorical distribution parameters as

$$\theta_k(\eta) = \text{softmax}_k(\eta) = \frac{e^{\eta_k}}{\sum_{k'} e^{\eta_{k'}}} \tag{28}$$

note that $\theta_k(\eta)$ automatically satisfies

$$0 \leq \theta_k(\eta) \leq 1$$
$$1 = \sum_{k=1}^{K} \theta_k(\eta) \tag{29}$$

- softmax maps $\mathbb{R}^K$ into the **simplex** of θ parameters required to define a categorical distribution.
- We can not do linear regression on θ because of the constrains on θ.
- but $\eta \in \mathbb{R}^K$ has no constrains.

Logistic Regression II

We can re-state the definition of the logistic regression model with the equivalent definition

Logistic Regression

- $Y|_X$ is a categorial variable with K classess
- The parameters of the categorical distribution $Y|_X$ are

$$\theta_k(x) = \underset{k}{\text{softmax}}(\eta(x)) \tag{30}$$

- the K-dimensional vector η (called **logits**) depend linearly on x

$$\eta(x) = W^T x + b \tag{31}$$

- Because $\text{softmax}(\eta_k + \Delta) = \text{softmax}(\eta)$ one of the dimensions on η is **redundant**.
- There are $(K-1) \times D + (K-1) = (K-1) \times (D+1)$ free parameters.

Logistic Boundary

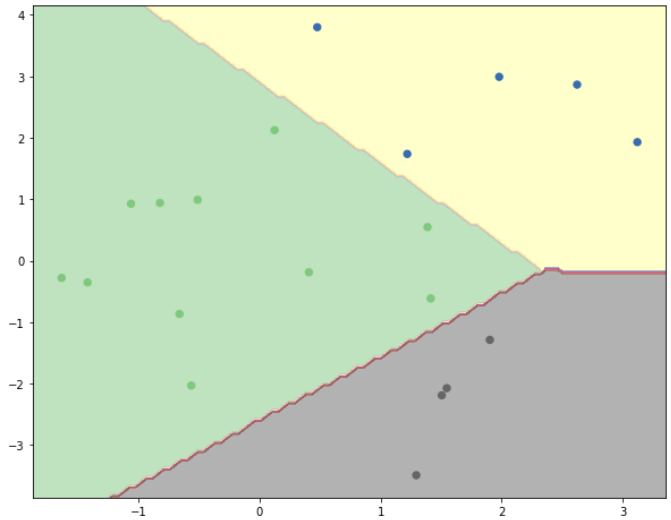

Figure 13: Logistic decision Boundary on a D=2, K=3 classification problem. $\approx$ 95% accuracy.

There are a total of 6 parameters been estimated

b $K-1=2$ parameters

W $(K-1)\times D=4$ parameters

Binary Classification With Logistic Regression

We discuss first the special case of binary classification ($K = 2$), because it appears often in practice and the algebra is simpler.
When there are only two classes, we can set

$$p(y = 1|x) = \frac{e^{W_1^T x + b_1}}{e^{W_0^T x + b_0} + e^{W_0^T x + b_0}}$$
$$p(y = 0|x) = \frac{e^{W_0^T x + b_0}}{e^{W_0^T x + b_0} + e^{W_0^T x + b_0}} \tag{32}$$

factoring out a term $e^{W_0^T x + b_0}$ and defining $w = W_1 - W_0$ and $b = b_1 - b_0$ we obtain

$$p(y = 1|x) = \frac{e^{w^T x + b}}{1 + e^{w^T x + b}} = \text{logistic}(w^T x + b)$$
$$p(y = 0|x) = 1 - \text{logistic}(w^T x + b) \tag{33}$$

- The logistic functions if just a special case of the soft max function for $K = 2$.
- the inverse of the logistic function is the **logit** function

Logit and Logistic Functions

A bit of Nomenclature

Given a probability $0 < p < 1$ the **odds** are defined as

$$O = \frac{p}{1-p} \tag{34}$$

They are the ratio you would be willing to pay on a fair bet for an event with provability p. The logit function computes the log of the odds given a probability is

log odds (logit)

$$\eta = \text{logit}\, p = \log \frac{p}{1-p} \tag{35}$$

The logit function maps $0 \leq p \leq 1$ into $-\infty < \eta < \infty$.
The inverse of the logit function is called the

Logistic Function

$$p = \text{logit}^{-1}\eta = \frac{e^{\eta}}{1+e^{\eta}} \tag{36}$$

It maps $\eta \in \mathbb{R}$ into $p \in [0, 1]$.

Binary Logistic Regression

Given a pair of random variables Y, X, where $Y \in \{0, 1\}$ is Bernoulli distributed conditional on $X \in \mathcal{R}^D$, a D-dimensional vector of features

$$P(y|x) = p(\eta)^y \left(1 - p(\eta)\right)^{1-y} \tag{37}$$

In Logistic regression we assume that **log odds** are **linear** on the sample features x.

$$\eta(x) = w^T x + b \tag{38}$$

with

$$\begin{aligned} p(y = 1|x) = p &= \frac{e^\eta}{1 + e^\eta} = \frac{e^{w^T x + b}}{1 + e^{w^T x + b}} \\ p(y = 0|x) = 1 - p &= \frac{1}{1 + e^\eta} = \frac{1}{1 + e^{w^T x + b}} \end{aligned} \tag{39}$$

and, therefore

$$p(y|x) = \frac{e^{y\eta}}{1 + e^\eta} = \frac{e^{y(w^T x + b)}}{1 + e^{w^T x + b}} \tag{40}$$

The parameters w_d $d = 1, \ldots, D$ are called the factor **loadings** and b the **bias**.

Binary Logistic Regression Classifier

- For classification we set $y_{\text{pred}} = 1$ if probability is larger than a threshold $P(y|x) > p_0$.
- If the objective function is classification error, $p_0 = \frac{1}{2}$ is the optimal choice.
- In contrast to LDA, we postulate directly a functional form for $P(y|x)$, this makes the model more robust to model assumptions and may require less data to train.
- Because $p(y|x)$ is non-linear, there is **no analytic solution** to the maximum likelihood problem. We must use numerical optimization to **solve** for w and b.

Max Likelihood

To estimate parameters w and b given a set of samples $\{y_i, x_i\}$ we minimize the max likelihood loss

$$\hat{E}(w, b; \{y_i, x_i\}) = -\frac{1}{N}\sum_i \log p(y_i|x_i; w, b)$$

$$= \frac{-1}{N}\sum_i^N \left\{ y_i(w^T x_i + b) - \log\left(1 + e^{w^T x_i + b}\right)\right\} \quad (41)$$

The minimum error will be found when the first order conditions are satisfied

$$\frac{\partial E}{\partial \hat{w}_d} = 0 = \frac{1}{N}\sum_i (\theta(x_i) - y_i)x_{i,d}$$

$$\frac{\partial E}{\partial \hat{b}} = 0 = \frac{1}{N}\sum_i (\theta(x_i) - y_i) \quad (42)$$

where $\theta(x)$ is logistic model probability for features x

$$\theta(x) = \text{logistic}(w^T x + b) = \frac{e^{w^T x + b}}{1 + e^{w^T x + b}} \quad (43)$$

Note that samples x_i that are well predicted $y_i \approx \theta(x_i)$ do not contribute to the gradient, but miss-predicted samples push the gradient in the direction of x.

Finding a solution to the Logistic MLE Problem

- Once we have $\hat{E}(w, b; \{y_i, x_i\})$ we can use a numerical minimization routine.
- There are lots of algorithmic choices for minimization with different trade-offs.
- In python `scipy.optimize.minimize` is a generic driver to minimization.
- For problems with N and D large, a variation of Steepest Descent is usually a good choice and can be easily hand coded.

Steepest Descent

$$\begin{aligned} w_{d;t+1} &= w_{d;t} - \lambda \frac{\partial \hat{E}(w_t, b_t; \{y_i, x_i\})}{\partial w_d} \\ b_{t+1} &= b_t - \lambda \frac{\partial \hat{E}(w_t, b_t; \{y_i, x_i\})}{\partial b} \end{aligned} \tag{44}$$

- λ is the learning rate, usually a small number to be found by cross-validation.
- $w_{k,0}, b_0$ are some initial guess that we can initialize at random.
- Stop when $\hat{E}_t$ stabilizes or when $t = T$ a specified number of time steps.

Multi Class Logistic Regression

Recap

The case with multi-class Y is analogous.
To recap:

$$P(Y|X) = \prod_{k=1}^{K} \theta_k(X)^{Z_k} \tag{45}$$

where Z is the one-hot encoding of Y. The logistic regression model assumes that

$$\theta_k(X) = \text{softmax}(\eta(X))_k \tag{46}$$

with

$$\eta_k(X) = \sum_{d=1}^{D} W_{k,d} X_d + b_k \tag{47}$$

The $K \times D$ **weight matrix** $W_{k,d}$ and the K-dimensional **bias vector** need to be learned from the data using **max likelihood** inference.

Multiclass Logistic Regression II

The multi-class log likelihood loss can be written as

$$\hat{E}(W, b; \{Z, X\}) = -\frac{1}{N} \sum_{i=1,k'=1}^{N,K} Z_{i,k'} \log \text{softmax}(\sum_{d=1}^{D} W_{k',d} X_{i,d} + b_{k'}) \tag{48}$$

Using Equation 23 we find

$$\frac{\partial \hat{E}}{\partial W_{k,d}} = \frac{1}{N} \sum_i \left(\text{softmax}(\eta)_k - Z_{i,k}\right) X_{i,d} \quad \frac{\partial \hat{E}}{\partial b_k} = \frac{1}{N} \sum_i \left(\text{softmax}(\eta)_k - Z_{i,k}\right) \tag{49}$$

- As in the binary case, samples that are well predicted ($Z_{i,k} \approx \text{softmax}(\eta_k(X_i))$ do not contribute to the gradient.
- We can solve using **gradient descent**

We must solve for a $K \times D$ W matrix and a K dimensional b vector:

Multivariate Logistic Regression Steepest Descent

$$\begin{aligned} W_{k,d;t+1} &= W_{k,d;t} - \lambda \frac{\partial \hat{E}(W_t, b_t; \{y_i, x_i\})}{\partial W_{k,d}} \\ b_{k;t+1} &= b_{k;t} - \lambda \frac{\partial \hat{E}(W_t, b_t; \{y_i, x_i\})}{\partial b_l} \end{aligned} \tag{50}$$

Regularization

We can introduce a weight penalty so that the regularized loss is

$$\hat{E}_\rho(W, b; \{y_i, x_i\}) = \hat{E}(W, b; \{y_i, x_i\}) + \frac{\rho}{2} \sum_{k,d} W_{k,d}^2 \tag{51}$$

The bias is, traditionally not regularized, as that would affect the class means. With that definition the gradient descent algorithm becomes

Regularized Multivariate Logistic Regression Steepest Descent

$$\begin{aligned} W_{k,d;t+1} =& (1 - \lambda\rho) W_{k,d;t} - \lambda \frac{\partial \hat{E}(W_t, b_t; \{y_i, x_i\})}{\partial W_{k,d}} \\ b_{k;t+1} =& b_{k;t} - \lambda \frac{\partial \hat{E}(W_t, b_t; \{y_i, x_i\})}{\partial b_l} \end{aligned} \tag{52}$$

The parameter ρ **shrinks** the weights towards zero at each iteration.

Binary Classifiers vs Multi class Classifiers

An Aside

- Some classifiers (Binary Logistic, traditional Support Vector Machine) are intrinsically binary.
- There are **heuristic** procedures to convert a binary classifier into a K multiclass classifier

One vs Rest for each class $k = 1, \dots K$ train a binary classifier on the binary classification problem $\tilde{Y}_i = (Y_i == k)$. Classify as

$$\hat{k} = \arg\max_k p(\tilde{Y}_k = 1|x) \tag{53}$$

Because the probabilities of each classifier are trained independently, $\sum_k p(\tilde{Y}_k = 1|x) \neq 1$

One vs One for each pair of classes k, k' train a classifier on the samples that belong to both classes. Train a total of $K(K-1)/2$ classifiers and select the classifier with most **votes**.

Read documentation **carefully** in `sklearn` to understand what kind of multi-classfier you are using.

MNIST

As we saw in the lecture on Naive Bayes, the distribution of intensities on pixels is closer to Bernoulli than to Gaussian, so an LDA model is a poor choice.

Method	**Parameters**	**Accuracy**
Nearest Neighbors	Non-Parametric	97%
Bernoulli Naive Bayes	7,840	84%
QDA	3,085,049	54%
LDA	315,569	87%
Logistic	7,065	92%

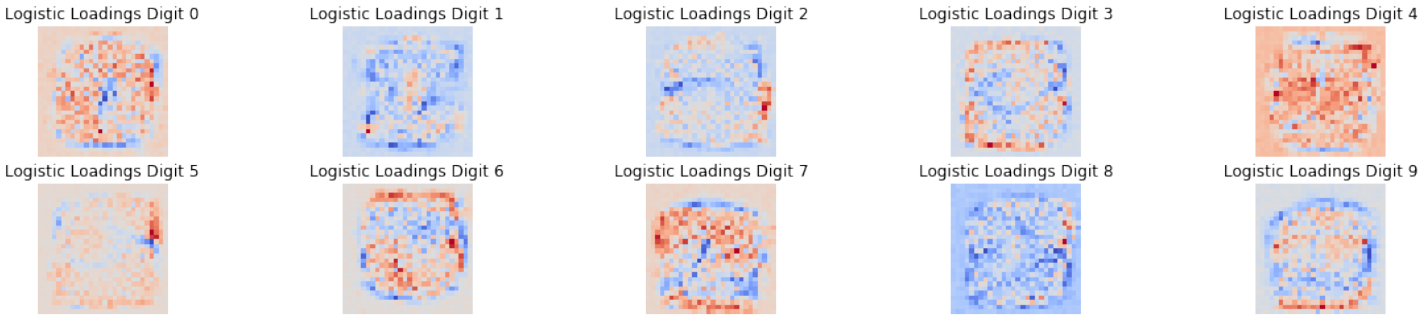

Figure 14: Logistic loadings for MNIST problem.

Fashion MNIST Data Set

- Produced by Zalando Research
- Same structure as MNIST (28×28 gray scale, same file names, etc).
- Harder inference problem:

Method	Accuracy
Nearest Neighbors	85%
Bernoulli Naive Bayes	71%
QDA	56.7%
LDA	81.5%
Logistic	82.6%

- You can see many more classifiers results here.

Figure 15: The Fashion MNIST Data Set.

χ^2 Feature Selection for Text

In text classification problems X is a very large dimensional vector (D is the size of the vocabulary).
We may want to use a **subset** of the vocabulary of size $F < D$.
We can **rank** categorical features X and Y based on the

χ^2 Feature Selection Criteria

$$C^2(X, Y) = \sum_{k=1}^{K} \sum_{d=1}^{D} \frac{(\hat{O}_{k,d} - \hat{E}_{k,d})^2}{\hat{E}_{k,d}} \tag{54}$$

with

$$\hat{E}_{k,d} = W \hat{p}_k^Y \hat{p}_d^X \tag{55}$$

- W is the total number of words in corpus.
- $\hat{p}^Y$ and $\hat{p}^X$ are the marginal distributions of classes (Y), and words (X).
- X and Y are categorical distributed variables.
- As we discussed in Lecture 2 C^2 has a χ^2 distribution if X and Y are independent from document class Y.
- The idea is that we pick the categorical variables X that look **the furthest** from independence.

C50 Text Classification

On the C50 Author classification problem $D \approx 28,000$. We use a χ^2 test to select the F more significant words (using 'Set' features) and train a classifier:

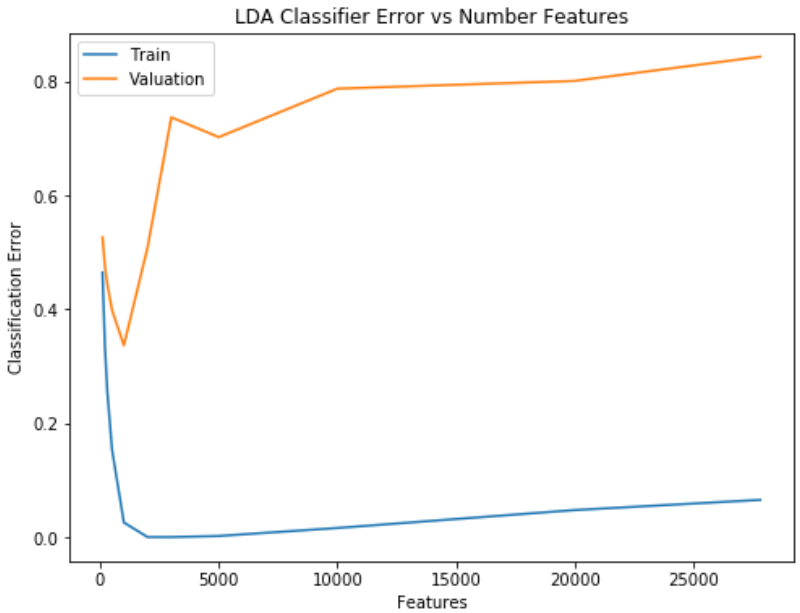

Figure 16: Performance of LDA classifier as a function of the number of features

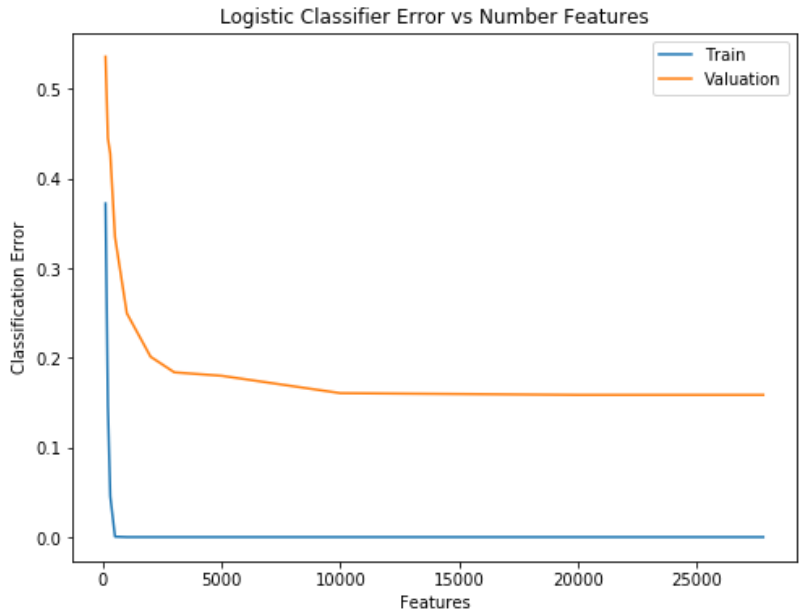

Figure 17: Performance of Logistic Regression as a function of the number of features

LDA is again a poor choice, but Logistic regression does fairly well with $\approx 84\%$ accuracy (Naive Bayes had $\approx 76\%$ accuracy).

Amazon Fine Food Reviews

For the Amazon Fine food reviews, a logistic classifier is a slight improvement over the Naive Bayes classifier trained for the review summary.

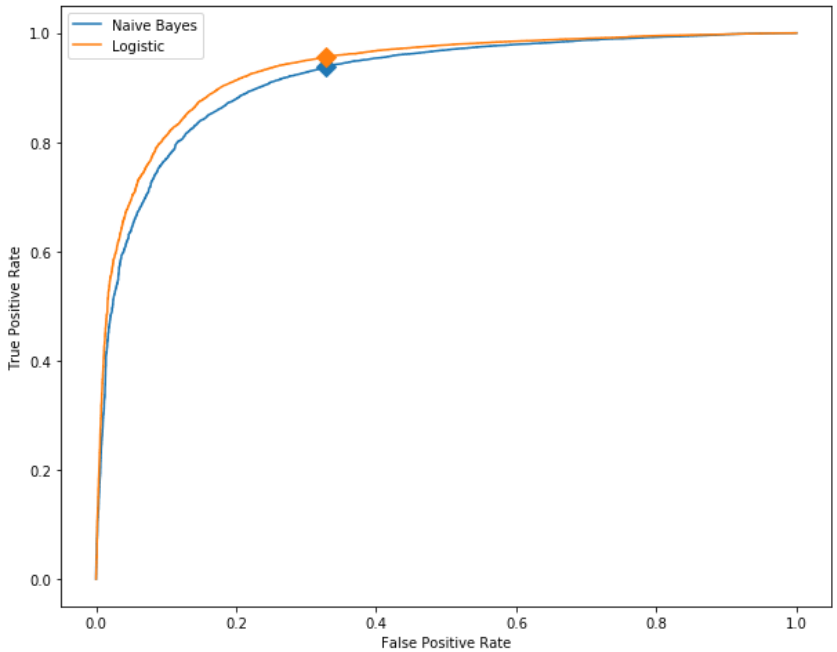

Figure 18: Comparison of the ROC curves for Naive Bayes and Logistic Regression. Logistic Regression dominates.

Comparison of C50 and Amazon Review Problems

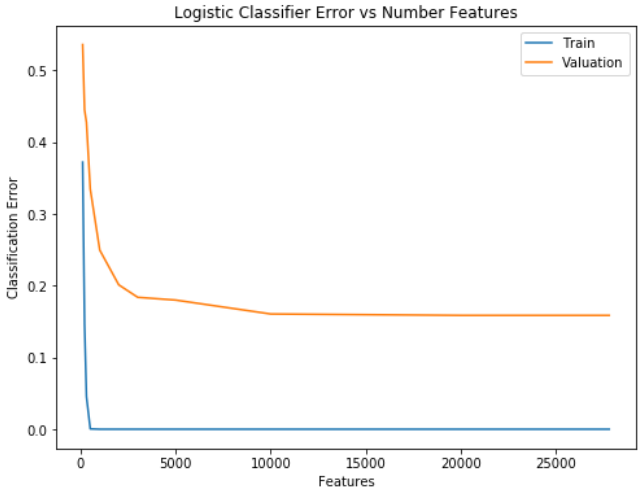

Figure 19: Performance of Logistic Regression as a function of the number of features for the **C50 Corpus**

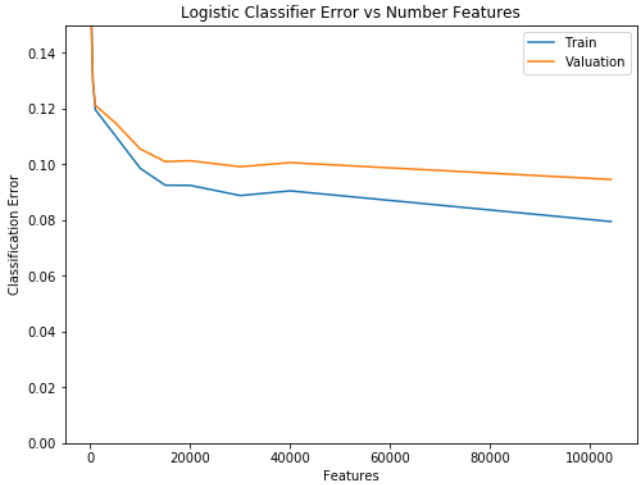

Figure 20: Performance of Logistic Regression as a function of the number of features for the **Amazon Reviews Corpus**

- The C50 Corpus has only 2,500 documents. A logistic Classifier with 1,000 features already over-fits the training data completely.
- The Amazon Reviews corpus has $\approx 500,000$ reviews, even using the complete vocabulary of $\approx 100,000$ words, the performance of training and validation sets are comparable.

Bibliography


Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani.
An Introduction to Statistical Learning with Applications in R.
Springer New York Inc., 2013.


Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze.
Introduction to Information Retrieval.
Cambrige University Press, 2008.
https://nlp.stanford.edu/IR-book/information-retrieval-book.html.