Classification I

Lecture 4

Manuel Balsera

IEOR, Columbia University

February 9, 2018

Outline

Road Map

- today we start our journey through ML classification algorithms.
- First we try a **brute force** algorithm with very weak assumptions about data: **K-Neighbors**.
  - We will show it can work very well given **enough data**.
  - doomed by the **curse of dimensionality**.
  - We will follow [1]Chapter 4 of Duda et al for K-Neighbors.
  - Chapter 13 of [2]The information Retrieval Book is a good reference on Naive Bayes.
- We will then try making **drastic** assumptions about data: **Naive Bayes**
  If data representation is adequate, it may work remarkably well.
- We will use two examples as a guide:
  1. Image recognition: MNIST
  2. Text Classification: C50

Bayes Classifier

- Assume y is a categorial random variable taking $k = 1, \cdots, K$ classess.
- If we know the conditional probability $p(y = k|x)$ we can implement the

Bayes Classifier

$$\hat{k} = \arg\max_k p(y = k|x) \tag{1}$$

- The error rate of the Bayes classifier is the **Bayes Error Rate**.
- The Bayes error rate is the lowest possible error rate for any classifier.
- We will in practice use $\hat{p}(y, x)$ and approximation to the true conditional probability
- In this class we study two approximations to $\hat{p}$ and their classifiers

Density Estimation

Let's consider a random variable $S \in \mathcal{S}$ with density $p(s)$.
The probability that a sample s_i falls in region $R \subset \mathcal{S}$ is

$$\theta = \int_R \mathrm{d}s\, p(s) \tag{2}$$

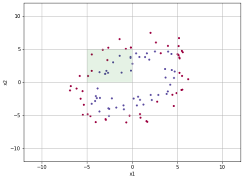
This is a Bernouilli random variable. Given N samples $\{s_i\}$ the max likelihood estimate of θ is

$$\hat{\theta} = \frac{\hat{N}_1}{N} = \frac{k}{N} \tag{3}$$

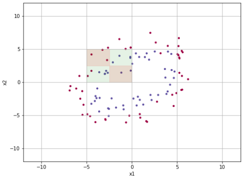
where $\hat{N}_1 = k$ is the number of samples s_i that fell inside the region R. If the region R has a small volume V centered around s_0 we have the

Monte Carlo Density Estimator

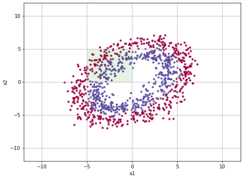
$$\hat{p}(s_0) \approx \frac{k}{N}\frac{1}{V} \tag{4}$$



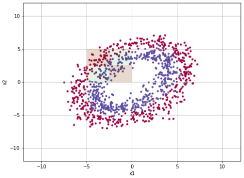
(a) $N = 50$ per class, $\hat{N}_B = 11$, $\hat{N}_R = 5$, $\hat{p}_B = 0.88\%$, $\hat{p}_R = 0.4\%$



(b) Upper Right $\hat{N}_B = 4$, $\hat{N}_R = 0$, $\hat{p}_B = 1.28\%$, $\hat{p}_R = 0.0\%$



(c) $N = 500$ per class, $\hat{N}_B = 89$, $\hat{N}_R = 59$ $\hat{p}_B = 0.68\%$, $\hat{p}_R = 0.48\%$



(d) Upper Right $\hat{N}_B = 28$, $\hat{N}_R = 14$, $\hat{p}_B = 0.90\%$, $\hat{p}_R = 0.44\%$

Table 1: Count estimates for two classes

Convergence of Density Estimator

- We want a small V so that $\int_V \mathrm{d}s\, p(s) \approx p(s_0)V$
- But if V is small the number of "hits" k will be small, our estimate $\hat{\theta}$ will be bad.

Consider a sequence of experiments with $N \to \infty$, where we make V_N and k_N depend on N. Define

$$\hat{p}_N(s_0) = \frac{k_N}{N}\frac{1}{V_N} \tag{5}$$

As $N \to \infty$ for convergence we need

$$\begin{aligned} V_N &\to 0 \\ k_N &\to \infty \\ \frac{k_N}{N} &\to 0 \end{aligned} \tag{6}$$

Parzen Windows and K-nearest neighbours

We can assure convergence two ways:

Parzen Window Estimation fix $V_N = \frac{1}{\sqrt{N}}$, determine k_N from data.

- we control **bias** explicitly through V_N
- When density is low, k_N is small: **high variance**
- When density is high, k_N is large: **low variance**
- For fixed N can be problematic (k could be zero).

K-neighbors Estimation fix $K_N = \sqrt{N}$, determine V_N from data.

- Control **variance** explicitly. Method is **adaptative**.
- When dentity is high, V_N will be small: **low bias**
- When density is low V_N is high: **high bias**

Top Hat Kernel

Assume $V_N = h_N^D$ is a D-dimensional hyper-cube centered on s_0.
Define the **top hat function**

$$\varphi(u) = \begin{cases} 1 & |u_d| < \frac{1}{2} \quad d = 1, \dots, D \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

$$\varphi(\frac{s_i - s_0}{h_N}) = \begin{cases} 1 & s_i \text{ in } s_0 \text{ hypercube} \\ 0 & s_i \text{ out of hypercube} \end{cases} \tag{8}$$

$$k_N = \sum_i^N \varphi(\frac{s_i - s_0}{h_N}) \tag{9}$$

$$\hat{p}_N(s_0) = \frac{1}{N} \sum_{i=1}^N \frac{1}{V_N} \varphi(\frac{s_i - s_0}{h_N}) \tag{10}$$

Density Kernels

We can generalize the definition of φ to any probability density

Kernel Density

Given a kernel density

$$\varphi(u) \geq 0$$
$$\int \mathrm{du}\, \varphi(u) = 1 \tag{11}$$

we have

$$\hat{p}_h(s_0) = \frac{1}{N} \sum_i^N \frac{1}{h^D} \varphi(\frac{s_i - s_0}{h}) \tag{12}$$

Density estimate is a **convolution** of a kernel density φ over data samples.

Common Kernels

Top Hat	$\varphi(u) = \frac{1}{2}$ if $\lvert u\rvert < 1$.
Gaussian	$\varphi(u) \propto e^{-\frac{u^2}{2}}$.
Epanechnikov	$\varphi(u) \propto 1 - u^2$ for $\lvert u\rvert < 1$.
Exponential	$\varphi(u) \propto e^{-\lvert u\rvert}$.
Linear	$\varphi(u) \propto 1 - \lvert u\rvert$ for $\lvert u < 1$.
Cosine	$\varphi(u) \propto \cos \frac{\pi u}{2}$ for $\lvert u\rvert$¡1.

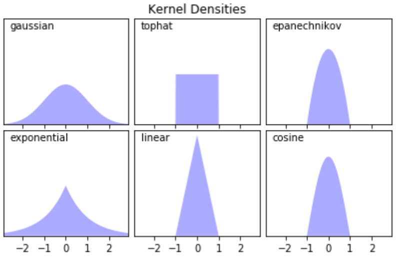


Figure 1: Adapted from: Jake Vanderplas—sklearn

Bandwidth and Bias-Variance Tradeoff

- h controls the bias/variance tradeoff. Is called *bandwidth* or *kernel width*.
- Kernels are similarity measures φ decreases with u.
- Expected value over all samples of $p(s_0)$ is a **convolution**

$$\mathbb{E}(\hat{p}(s_0)) = \int_{\mathcal{S}} \mathrm{d}s \, p(s)\varphi(\frac{s - s_0}{h}) \tag{13}$$

- Larger h implies lower variance (see Section 4.3 of Duda's book)

$$\mathrm{Var}((\hat{p}(s_0)) \leq \frac{\sup(\varphi)\hat{p}(s_0)}{Nh^D} \tag{14}$$

Course of Dimensionality

- Assume $\mathcal{S} = [0,1]^D$ is the unit hyper-cube. $p(s) \approx 1$ uniform.
- With resolution $h \approx 0.1$, $V = 10^{-D}$. We need $N \approx 10^D$ samples to estimate $\hat{k}$ for all s.
- Become impractical to sample $\mathcal{S}$ with any precision.
- Sometimes $P(S)$ is concentrated in a manifold (hyper-surface) of **effective dimension** $D_{\text{eff}} \ll D$ and sampling is still possible.

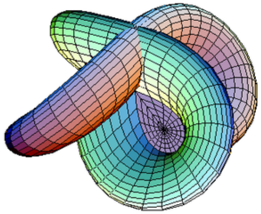


Figure 2: A $D_{\text{eff}} = 2$ manifold embedded in $\mathbb{R}^3$. Source: Wikipedia

K-Neighbors Class Probabilities

Given a categorical variable Y taking values $y = 1, \dots L$, and N samples $s_i = \{y_i, x_i\}$ we can estimate $p(y|x)$ using bayes theorem

$$p(y|x) = \frac{p(x,y)}{p(x)} = \frac{p(y,x)}{\sum_{y'} p(y',x)} \tag{15}$$

As before, we have the Maximum Likehood estimate

$$\hat{p}(y,x) \approx \frac{k_y}{N}\frac{1}{V} \tag{16}$$

where V is a small volume centered aroud x, and k_y is the number of observations i of category y that fell in volume V.
Given $K = \sum_y k_k$, the number of samples that fell on volume V we have

K Neighbors Probabilities

$$\hat{p}(y|x) \approx \frac{k_y}{K} \tag{17}$$

K-Neighbors Classifier

- If we keep the volume V fixed, and use a kernel φ we have **kernel classification**.
- If we fix K and let V be determined by the data $\{x_i\}$ we have a **nearest neighbors** classifier.
- To minimize classification error we select the class with higher probability.

K-Neighbors Classifier

Given a point x, predict the most frequent class y along the K closest neighbors x_i on the training data.

Properties of K-Neighbors Classifier

- Training is trivial: store all samples $\{y_i, x_i\}$
- Prediction is expensive $O(DN)$: Search through all samples. There are speed up techniques.
- The parameter K controls the Bias-Variance trade-off.
- A special case is the **Nearest Neighbor** classifier.
- As $N \to \infty$ Nearest Neighbor classifier error is bounded by *twice* the Bayes error rate.

Example Density Estimation

Let's sample 100 points from a **Mixed Gaussian** distribution

- $S \sim \mathcal{N}(0,1)$ with probability 0.3
- $S \sim \mathcal{N}(5,1)$ with probability 0.7

We set the bandwidth to $h = \frac{1}{2}$

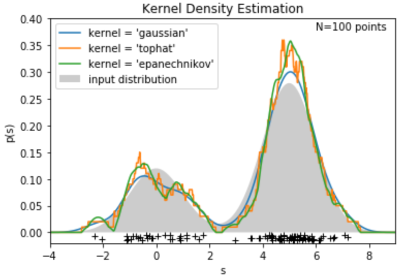
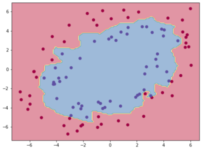
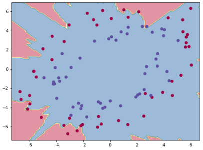
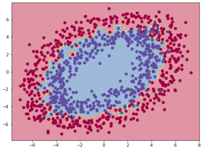


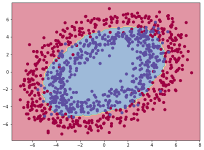
Figure 3: Adapted from: Jake Vanderplas—sklearn

Ellipse Classification Problem


(a) Nearest Neighbors classifier with $N = 100$. Accuracy 88%.


(b) 31st nearest neighbors classifier with $N = 100$. Accuracy 60%.


(c) Nearest Neighbors classifier with $N = 1,000$. Accuracy 90%.


(d) 31st nearest neighbors classifier with $N = 1,000$. Accuracy 92.5%.

MNIST Classification

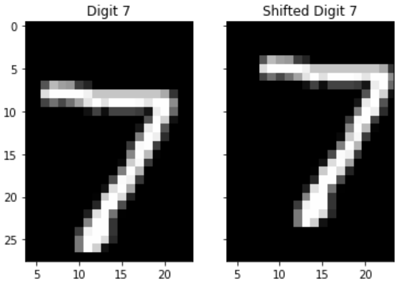
- Images are represented as $R \times C = 28 \times 28$ matrices $F_{r,c}$.
- Feature vector X_d is the linearized image pixels $d = 1, \dots R \times C$

$$X_{i,C*r+w} = F_{i,r,c} \tag{18}$$

- Distance is regular Euclidean L_2 distance for X_d
- Nearest neighbors has **96.9% accuracy** in test sample.
- Prediction is **slow**, can only classify 8 images per second.
- Prediction requires the whole training sample: consumes disk and memory.

MNIST Sample Preparation

- This two images share no pixels (very far in L_2 distance).



- MNIST images have been carefully prepared (centered).
- If we train with digits shifted ± 3 pixels, out of sample performance **drops to 47%**
- Images **must be carefully prepared** for K-neighbors to do well.

C50 Text Classification

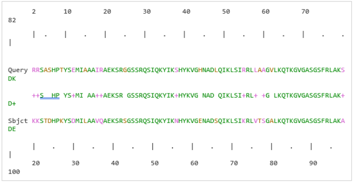
- We classify the **C50** Reuters text documents using our pre-build feature vectors.
- We use **cosine** distance (L_2 distance would be dominated by document length).

Features	$K = 1$	$K = 5$
Set	59.4%	61.4%
Count	53.9%	52.7&
Tf-Idf	54.3%	54.6&

Figure 4: K-Neighbors out of sample classification accuracy on the C50 Documents

- There are 50 balanced classes. Random guessing has 2% accuracy.
- **Set** binary features outperform: Once a word is included in a text there seem to be little extra information if it re-appears.

Application: Protein 3D Structure Prediction



```
     2      10      20      30      40      50      60      70
82
     |  .   |   .   |   .   |   .   |   .   |   .   |   .   |   .   .
|

Query RRSASHPTYSEMIAAAIRAEKSRGGSSRQSIQKYIKSHYKVGHNADLQIKLSIRRLLAAGVLKQTKGVGASGSFRLAKS
DK
      ++S  HP YS+MI AA++AEKSR GSSRQSIQKYIK+HYKVG NAD QIKLSI+RL+ +G LKQTKGVGASGSFRLAK+
D+
Sbjct KKSTDHPKYSDMILAAVQAEKSRSGSSRQSIQKYIKNHYKVGENADSQIKLSIKRLVTSGALKQTKGVGASGSFRLAKA
DE
     |   .   |   .   |   .   |   .   |   .   |   .   |   .   |   .
|
     20      30      40      50      60      70      80      90
100
```

Figure 5: DNA binding protein Hist1 sequence alignment

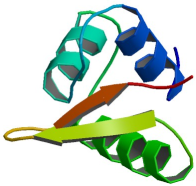


Figure 6: Hist1 3D structure.

- A protein is composed of sequence of amino acids.
- There are 22 choices, represented by letters A, R, N, etc...
- We define a distance $d(A, B)$ between pairs of amino acids

- We wish to predict a protein 3D structure from its sequence.
- After aligning two sequences we can define a distance based on the sum of paired amino acid distances
- We can then use **nearest neighbors** to predict 3D structure.

Naive Bayes Assumption

Curse of Dimensionality

The complexity of describing an **arbitrary function** $f(x)$ for $x \in \mathcal{X} \subset \mathbb{R}^D$ **grows exponentially** in D.

- in a classification problem we need to estimate $P(Y = y|x)$ for $y = 1, \dots, K$, a set of K separate D-dimensional functions.
- We need some **simplifying assumption**.
- Simplest approach is to assume different dimensions in x are **conditionally independent** given the class label y.

Naive Bayes Assumption

$$P(x|y) = \prod_{d}^{D} P(x_d|y) \tag{19}$$

Naive Bayes Classifier

We really need $P(y|x)$. Using Bayes theorem

$$P(y|x) = \frac{P(x,y)}{P(x)} = \prod_{d}^{D} P(x_d|y) \frac{P(y)}{P(x)} \tag{20}$$

- Very **strong** assumption. Hardly ever justified.
- Probability estimate almost always **wrong**.
- But we **hope** relative ordering of class probabilities preserved.
- Taking logs and dropping $P(x)$ (does not depend on y):

Naive Bayes Classifier

$$\hat{y}_{\mathrm{NB}} = \arg\max_{y} \sum_{d} \log P(x_d|y) + \log P(y) \tag{21}$$

Properties of Naive Bayes

- **Drastic simplification**: Learn D 1-dimensional functions, rather than one D-dimensional function.
- Ignores all **dependencies** between features x_d.
- We can make different assumptions on $P(x_d|y)$ based on the data. For example:

Bernouilli	if x_d is a binary variable
Multinomial	if x_d is categorical
Gaussian	if x_d is continuous and approximately Gaussian.

- It is really a **framework**: Can mix and match assumptions for different features x_d.
- **Not invariant** to equivalent re-statements of features x_d. If x_A and x_B are independent:

Gaussian	$x_A + x_B$ and $x_A - x_B$ not independent.
Bernoulli	$x_A \,\mathrm{xor}\, x_B$ and x_A not independent.

Gaussian Naive Bayes

Given sample data $\{y_i, x_{i,d}\}$ where

- $i = 1, \ldots N$ runs over the data samples.
- y_i is a categorical variable taking the values $k = 1, \ldots, K$
- the input data is D-dimensional $d = 1, \ldots D$
- $x_{i,d}$ given y is approximately normally distributed

The Gaussian Naive Bayes assumption is

$$p(x_{i,d}|y = k) = \mathcal{N}(x_{i,d}; \hat{\mu}_{d,k}, \sigma^2_{d,k}) \tag{22}$$

where different dimensions d are independent.
Using the one-hot $z_{i,k}$ representation of the labels y_i the max likelihood estimate of $p(x_{i,d}|y_i = k)$ has parameters

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

$$\hat{\sigma}^2_d = \frac{1}{\hat{N}_k} \sum_i (x_{i,d} - \mu_{d,k})^2 z_{i,k} \tag{23}$$

Gaussian Naive Bayes Loss Function

The marginal probability of class k is

$$\hat{\pi}_k = \frac{\hat{N}_k}{N} \tag{25}$$

The loss function for a sample $x = (x_1, \ldots, x_D)$ is

$$L_k(x) = -\log p(y = k|x) = \frac{1}{2} \sum_d \left\{ \left(\frac{x_d - \hat{\mu}_{d,k}}{\hat{\sigma}_{d,k}} \right)^2 + \log \left(2\pi \hat{\sigma}^2_{d,k} \right) \right\} - \log \hat{\pi}_k \tag{26}$$

And finally, the predicted class will be

$$p(y = k) = \hat{k} = \arg\min_k L_k(x) \tag{27}$$

Geometry of Gaussian Naive Bayes Decision Boundary

At the **decision boundary** where the probability of classes k_1 and k_2 is the same must have that

$$L_{k_1}(x) = L_{k_2}(x) \tag{28}$$

or, using the explicit expression of the loss function

$$0 = \frac{1}{2}\sum_d \left\{ \left(\frac{x_d - \hat{\mu}_{d,k_1}}{\hat{\sigma}_{d,k_1}}\right)^2 - \left(\frac{x_d - \hat{\mu}_{d,k_2}}{\hat{\sigma}_{d,k_2}}\right)^2 + \log\frac{\hat{\sigma}^2_{d,k_1}}{\hat{\sigma}^2_{d,k_2}} \right\} - \log\frac{\hat{\pi}_{k_1}}{\hat{\pi}_{k_2}} \tag{29}$$

- This is a D-dimensional quadratic surface. If $D = 2$ this is a conic section: ellipse, hyperbola or parabola.
- Because there are no cross terms $x_d x_{d'}$, the conic axis must be align with the coordinate axis

Gaussian Naive Bayes Example

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

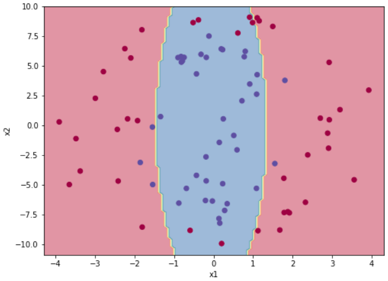


Figure 7: x_1 and x_2 are independent. Accuracy $\approx 78\%$

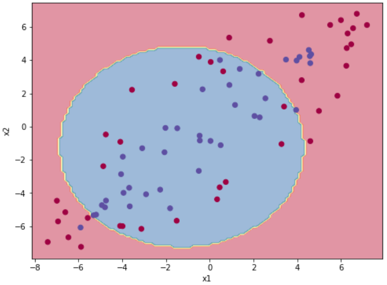


Figure 8: x_1 and x_2 are dependent. Accuracy $\approx 68\%$

Single Variance Approximation

Gaussian Naive Bayes approximation estimates $2D \times K + K - 1$ parameters
If data is scarce we can make the approximation that variance does not depend on class

$$\hat{\sigma}_d^2 = \frac{1}{N} \sum_k \sum_i \left(x_{i,d} - \hat{\mu}_{d,k}\right)^2 z_{i,k} \tag{30}$$

The equation satisfied by the boundary between two classes is then

$$0 = \frac{1}{2} \sum_d \left\{ \left(\frac{x_d - \hat{\mu}_{d,k_1}}{\hat{\sigma}_d} \right)^2 - \left(\frac{x_d - \hat{\mu}_{d,k_2}}{\hat{\sigma}_d} \right)^2 \right\} - \log \frac{\hat{\pi}_{k_1}}{\hat{\pi}_{k_2}} \tag{31}$$

which simplifies to

$$0 = \sum_d \frac{\left(\hat{\mu}_{d,k_1} - \hat{\mu}_{d,k_2}\right)}{\hat{\sigma}_d^2} \left\{ \frac{\left(\hat{\mu}_{d,k_1} + \hat{\mu}_{d,k_2}\right)}{2} - x_d \right\} - \log \frac{\hat{\pi}_{k_1}}{\hat{\pi}_{k_2}} \tag{32}$$

a **linear equation** for the boundary.

Naive Bayes and Text

Text representations are very high dimensional $\rightarrow$ Naive Bayes.

- A document is a sequence of words $D = \{w_1, \dots, w_t, \dots, w_T\}$.
- We wish to classify documents in $y = 1, \dots, K$ classes.
- **Naive Bayes Assumption**: given y, words w_t are chosen **independently** from a multinomial distribution.
- w_t is one of $1, \cdots, d, \dots, V$, where V is the size of the **vocabulary**.
- probability

$$P(D|y = k) = \prod_t^T P(w_t|y = k) \tag{33}$$

We need to stimate $P(w_t|y)$ from an N document **corpus** $D_i = \{w_{i,t}\}$.

Estimating Word Probabilities

- Given count $x_{i,d}$ of word d in document i we need to estimate the probability that work d is generated in class k.
- word d is a categorical variable with D possible values.
- The number of categories is large: $D = V$ the size of vocabulary.
- As V is large most of the counts will be small: we must use **Bayesian smoothing**
- The estimate of generating word d in class k is

$$\hat{P}(w = d|y = k) = \frac{\hat{n}_{k,d} + \alpha}{\hat{n}_k + \alpha V} \tag{34}$$

  - The counts of word d in class k are

$$\hat{n}_{k,d} = \sum_i x_{i,d} z_{i,k} \tag{35}$$

  - The total number of words in class k

$$\hat{n}_k = \sum_d \hat{n}_{k,d} \tag{36}$$

Naive Bayes Text Classifier

Given the count x_d of word $d = 1, \ldots, V$ in document $D = \{w_1, \ldots w_T\}$. The probability that document D is generated in class k is

$$P(D|y=k) = \prod_t^T P(w_t|y=k)P(y=k) = \prod_d^D \hat{P}(w=d|y=k)^{x_d}\hat{P}(y=k) \tag{37}$$

Taking logs and defining

$$w_{k,d} = \log \hat{P}(w=d|y=k) = \log \frac{\hat{n}_{k,d}+\alpha}{\hat{n}_k+\alpha V}$$

$$b_k = \log \hat{P}(y=k) = \frac{\hat{N}_k}{N} \tag{38}$$

where $\hat{N}_k$ is the number of documents in class k. The log probability is

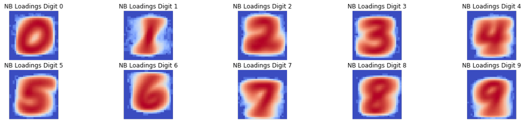
$$\hat{L}_k = \left(\sum_d x_d w_{k,d} + b_k\right) \tag{39}$$

Naive Bayes Text Classifier

$$y_{\mathrm{NB}} = \arg\max_k \hat{L}_k = \arg\max_k \left(\sum_d x_d w_{k,d} + b_k \right) \tag{40}$$

Naive-Bayes: MNIST

- Features are the linearized pixels $X_{i,Cr+c}$.
- $P(x_d|y)$ is probability that a pixel be on in class y.
- Gaussian model does poorly ($\approx 56\%$ accuracy): pixels have a bi-modal distribution, either on or off.
- Bernoulli approximation does much better $\approx 84\%$ (using $\alpha = 1$).
- Loadings $\log P(x_d|y)$ are easy to interpret

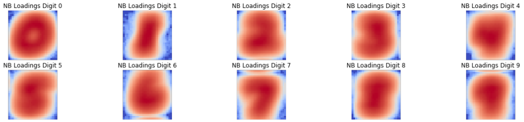


Naive-Bayes: Translation Invariance

- Allowing **off-center digits** degrades performance.

Train	Test	Accuracy
Centered	Centered	84%
Centered	Off-Center	38%
Off-Center	Off-Center	55%

- Loadings $\log P(x_d|y)$ are blurred.



- With enough blurring pixel densities will be uniform: Naive Bayes is useless (K-Neighbors still works given enough data).
- Problem is with our **features**: images are defined by **pixel correlations** not positions.

Naive-Bayes: C50 Documents

- Set Features again do better

Features	Accuracy
Set	65.8%
Count	65.0%
Tf-Idf	63.5%

- Set features again do better, but by only a slight margin.
- Naive-Bayes outperform K-neighbors: Text is very high-dimensional.
- For text classification Naive Bayes is the **baseline** model.

Conclusions

- We have discussed the **simplest** Machine learning algorithms.
- They make very different assumptions about feature distribution.

K-neighbors	: non-parametric. $\mathcal{X}$ is continuous and has a metric.
Naive Bayes	: conditional independence of features.

- They are both useful baselines to compare more sophisticated models to.
- They both can do remarkably well, but data must be prepared carefully.
- We have begun to explore how **features** affect performance.
- K-neighbor can be quite costly at prediction time, not always practical.
- We must find a disciplined way to choose **hyper-parameters**: K, α, text feature (*next lecture*).

Bibliography

Richard O. Duda, Peter E. Hart, and David G. Stork.
Pattern Classification (2nd Ed).
Wiley, 2001.
http://cns-classes.bu.edu/cn550/Readings/duda-etal-00.pdf.

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.