Bayesian Pattern Recognition

April 4, 2019

Patter Recognition

- Pattern Recognition, or Classification is the problem of assigning data to a discrete set of patterns/classes.
- Applications include medical diagnosis, extracting objects from images, radar, flagging improper content (e.g. videos, forum posts, etc), spam filters, etc.
- Pattern recognition is a huge area in machine learning as well as statistics and electrical engineering.
- What is the difference between pattern recognition in detection theory vs. machine learning?
- Detection theory always[1] imposes a probabilistic model on the data, whereas machine learning does not.

[1]for the purposes of this class

Pattern Recognition Example

- We would like to classify pixels into one of four colors.
- Our communications system has noise so none of the pixels exactly match the levels we are looking for.
- Let $x[m,n] = \theta_i + w[m,n]$ model each pixel value, where $\theta_i \in \Theta = 1, 2, 3, 4$.
- Here n represents the row of the matrix, and m is the column.
- $\mathbf{w} \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$ is the normal white noise.
- For a prior, assume that all colors are equally likely, $\pi(\theta_i) = \frac{1}{4}$

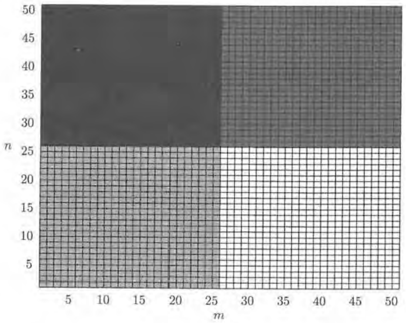


Figure 1: The ground truth image.

Pattern Recognition Example cont.

- Since $w[m,n]$ are normal, $p(x[m,n]|\theta_i) \sim \mathcal{N}(\theta_i, \sigma^2)$.
- Then the posterior is proportional to:

$$\frac{1}{4}\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{-1}{2\sigma^2}(x[m,n]-\theta_i)^2\right)$$

- To make a decision with the MAP rule, all we have to do is evaluate $p(\theta_i|x[m,n])$ for each possible θ_i and then pick the largest, i.e.:

$$i = \underset{\{1,2,3,4\}}{\arg\max} \quad p(\theta_i|x[m,n])$$

for every pixel.

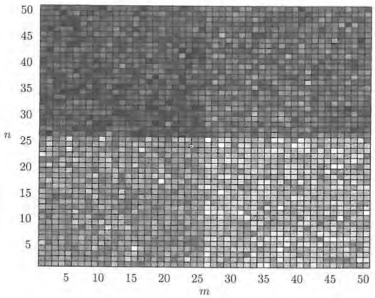


Figure 2: The noise-corrupted image.

Decision Regions for each Pixel

- Maximizing $p(\theta_i|x[n])$ is equivalent to maximizing its logarithm.
- Since the constants are the same in every case, we don't need to worry about those, and in fact the only thing we need to worry about is minimizing $(x[n]-\theta_i)^2$.
- This classifier is just deciding the color based on the closest pixel value.
- Then the decision regions are:

$$(-\infty, 1.5] \implies \theta_1 = 1$$

$$(1.5, 2.5] \implies \theta_2 = 2$$

$$(2.5, 3.5] \implies \theta_3 = 3$$

$$(3.5, \infty) \implies \theta_4 = 4$$

Naive Bayes Classifier

- For millions of pixels, this is a bit inefficient, since we have to do nearly the same thing four times for each pixel.
- A reasonable assumption is that pixels close to each other should be the same color (i.e. pictures are made of "blobs" of different colors).
- Let $\mathbf{X}_{m,n} = \begin{bmatrix} x[n+1, m-1] & x[n+1, m] & x[n+1, m+1] \\ [n, m-1] & x[n, m] & x[n, m+1] \\ [n-1, m-1] & x[n-1, m] & x[n-1, m+1] \end{bmatrix}$ be a 3×3 block from the image.
- To get this into an easier form to deal with let $\mathbf{x}_{m,n}$ be the vector where each column is stacked on top of each other.
- Assuming the pixels are independent

$$p(\mathbf{x}|\theta_i) \sim \mathcal{N}(\theta_i \mathbf{1}_9, \sigma^2 \mathbf{I}_{9\times 9})$$

where $\mathbf{1}_9$ is a vector of ones.

Naive Bayes Classifier Cont.

- Now following the same map procedure as before, we have:

$$i = \underset{i \in \{1,2,3,4\}}{\arg\max} \quad p(\theta_i|\mathbf{x}_{m,n}) =$$

$$\underset{i \in \{1,2,3,4\}}{\arg\max} \quad \frac{1}{4}\frac{1}{\sqrt{2\pi\sigma^2}^9}\exp(\frac{-1}{2\sigma^2}(\mathbf{x}_{m,n} - \theta_i\mathbf{1}_9)^T(\mathbf{x}_{m,n} - \theta_i\mathbf{1}_9))$$

- To maximize this, we choose i to minimize $(\mathbf{x}_{m,n} - \theta_i\mathbf{1}_9)^T(\mathbf{x}_{m,n} - \theta_i\mathbf{1}_9) = ||\mathbf{x}_{m,n} - \theta_i\mathbf{1}_9||_2^2$
- Now we are minimizing the mean squared distance from each potential color value vs a single pixel, so outliers are less likely to be misclassified.
- As long as the data is independent, the formulation

$$i = \underset{i \in \{1,2,3,4\}}{\arg\max} \quad p(\theta_i|\mathbf{x}_{m,n}) = \underset{i \in \{1,2,3,4\}}{\arg\max} \quad \pi(\theta_i)\prod_{i=1}^{N} p(\mathbf{x}|\theta_i)$$

is called the Naive Bayes Classifier.

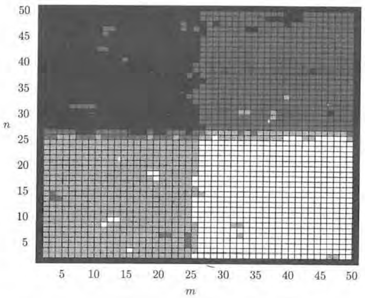
Comments on Naive Bayes Classifier

- Naive Bayes Classifiers are very good when the data are independent from each other.
- In the case where the number of classes is 2, this is a generative-discriminative pair with logistic regression, because:

$$\log \frac{p(\theta_1|\mathbf{x})}{p(\theta_2|\mathbf{x})}$$

  is also the quantity that logistic regression coefficients are modeling.
- As N increases, logistic regression performs better than Naive Bayes - i.e. the likelihood starts to dominate the prior.
- However, Naive Bayes will reach it's asymptotic error faster, so when N is small it can sometimes be better.
- Audience Participation: In our example of classifying 3×3 blocks, where do you think we will get errors?

Image Classification Results



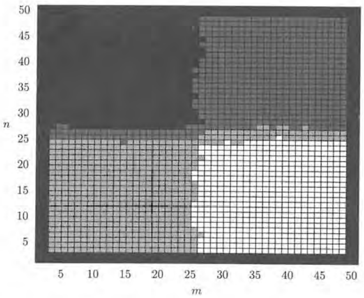


Figure 3: Left: Naive Bayes Classifier applied across a 3×3 window.
Right: Naive Bayes Classifier applied across a 5×5 window.

Comments on Window Size

- For a smaller window, most of the pixels are correct, but there are still some sections in the middle which are incorrectly classified.
- However, the edge is mostly correct.
- In the larger window, almost all the middle sections are correct, but the edges are wrong.
- There is a trade-off between window size, blob detection, and edge detection here.

A Naive Edge Detector

- Edge detection is it's own class of problem in image analysis, but I thought of a really bad edge detector using our probabilistic model.
- Assume we want to detect an edge between two pixels $x[m,n]$ and $x[m',n']$.
- The difference $||x[m,n]-x[m',n']||_2^2$between two pixels on an edge should be high.
- $x[m,n]-x[m',n']$ is normal with variance $2\sigma^2$, and

$$0, \theta_1-\theta_2, \theta_1-\theta_3,$$

$$\theta_1-\theta_4, \theta_2-\theta_3, \theta_2-\theta_4,$$

$$\theta_3-\theta_4, \theta_2-\theta_1, \theta_3-\theta_1,$$

$$\theta_4-\theta_1, \theta_3-\theta_2,$$

$$\theta_4-\theta_2, \theta_4-\theta_3$$

as all the possible means.

So Edgey

- Since we are only interested in the presence of an edge, use the fact that:

$$(\frac{x[m,n]-x[m',n']}{\sqrt{2}\sigma})^2 \sim \chi^2(1,\epsilon_i)$$

  where ϵ_i is one of the possible edges and is the mean of the non-central χ^2 distribution.
- Since an edge is not particularly common, give a high prior weight to this, say $\pi(\epsilon_i = 0) = 4/5$.
- Then assume the rest of the edges are equally likely - $\pi(\epsilon = \epsilon_i) = 1/5 * 1/12 = 1/60$
- Then the Naive bayesian classifier is:

$$\arg\max_{i\in\{0,1,2,\ldots,13\}} \quad \pi(\epsilon_i)p((\frac{x[m,n]-x[m',n']}{\sqrt{2}\sigma})^2|\epsilon_i)$$

- Alternatively, we could find $\arg\min_{i\in\{0,1,2,\ldots,13\}} (x[m,n]-x[m',n']-\epsilon_i)^2$

Do we detect the edges?

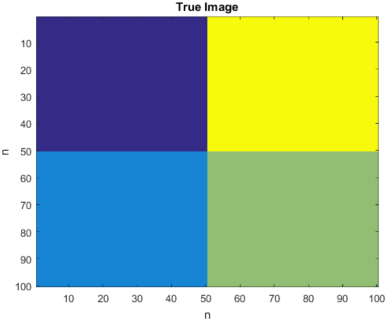


Figure 4: Ground Truth images, colors values are 25, 75,150,225, 100$\times$ 100 pixels

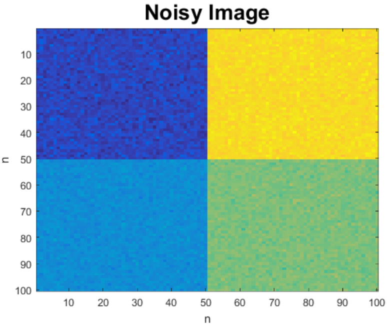


Figure 5: Noise Corrupted Image, $\sigma^2 = 30$

Swedish Flag Simulator

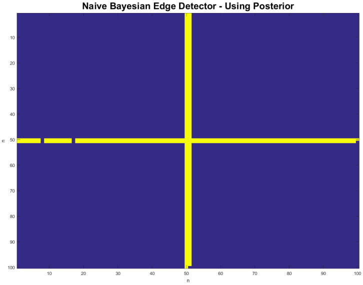


Figure 6: Edge Detector based on Naive Bayes Classifier, prior is $\pi(0) = 1 - 200/100^2$ and $\pi(\epsilon_i) = 1/12 * 200/100^2$ since there are 200 total edges considered.

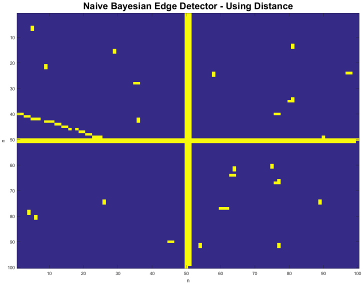


Figure 7: Edge Detector just based on the mean difference.

Why did I call this bad?

- For high variance noise, our test statistic has $2\sigma^2$ even higher variance than before, and the naive approach breaks down.
- Edges usually have some sort of covariance with each other, i.e. locally they follow lines, and overall they follow non-linear curves.
- If the difference in classes is small, the classifier can't tell the difference between noise and real edges. - The distance detector can't tell this at all.
- To solve the noise issue, people use a gaussian filter to smooth the image before edge detection. This was invented by John Canny.
- Inaccurate priors can weight the process too heavily towards one class.
- Like the original classifier, it doesn't scale well - have N^2 long loop which has to check all possible neighbors.

Failure Cases

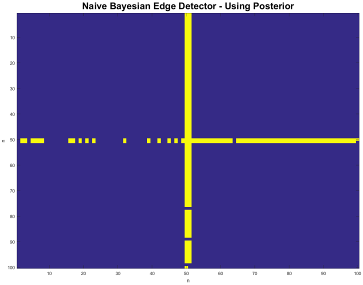


Figure 8: Edge Detector where σ^2 is increased to 75. We start to lose the edge between the two closest classes.

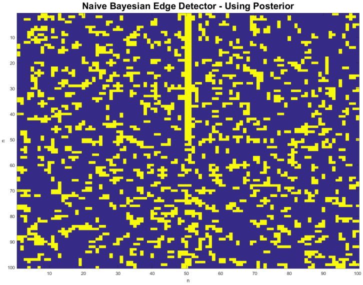


Figure 9: Edge Detector with relatively high variance, where the classes are 25,35,45,55, and the prior is set up incorrectly.