Classifiers: Naïve Bayes, k-NN, evaluation

Tomáš Svoboda, Petr Pošík, and Matěj Hoffmann
thanks to Daniel Novák and Filip Železný, Ondřej Drbohlav

Vision for Robots and Autonomous Systems, Center for Machine Perception
Department of Cybernetics
Faculty of Electrical Engineering, Czech Technical University in Prague

May 11, 2023

Bayes optimal strategy

- The Bayes optimal strategy : one minimizing mean risk. $\delta^* = \arg\min_\delta r(\delta)$
- s states, x possible measurements, $P(s,x)$ joint probababilities

$$r(\delta) = \sum_x \sum_s \ell(s, \delta(x)) P(x, s) = \sum_s \sum_x \ell(s, \delta(x)) P(s|x) P(x)$$

$$= \sum_x P(x) \underbrace{\sum_s \ell(s, \delta(x)) P(s|x)}_{\text{Conditional risk}}$$

Risk of a strategy is a weighted sum of conditional risks (conditioned by x)

- The optimal strategy is obtained by minimizing the conditional risk *separately* for each x:

$$\delta^*(x) = \arg\min_d \sum_s \ell(s, d) P(s|x)$$

A special case - Bayesian *classification*

- Attribute vector $\vec{x} = (x_1, x_2, \dots)$: pixels 1, 2, ...
- **State set $\mathcal{S}$ = decision set $\mathcal{D} = \{0, 1, \dots 9\}$.**
- State = actual class, Decision = recognized class
- Loss function :

$$\ell(s,d) = \begin{cases} 0, & d = s \\ 1, & d \neq s \end{cases}$$

Optimal decision strategy:

$$\delta^*(\vec{x}) = \arg\min_d \sum_s \underbrace{\ell(s,d)}_{0 \text{ if } d=s} P(s|\vec{x}) = \arg\min_d \sum_{s \neq d} P(s|\vec{x})$$

Obviously $\sum_s P(s|\vec{x}) = 1$, then:

$$P(d|\vec{x}) + \sum_{s \neq d} P(s|\vec{x}) = 1$$

Inserting into above:

$$\delta^*(\vec{x}) = \arg\min_d [1 - P(d|\vec{x})] = \arg\max_d P(d|\vec{x})$$

A special case - Bayesian *classification*

- Attribute vector $\vec{x} = (x_1, x_2, \dots)$: pixels 1, 2, ...
- **State set $\mathcal{S}$ = decision set $\mathcal{D} = \{0, 1, \dots 9\}$.**
- State = actual class, Decision = recognized class
- Loss function :

$$\ell(s,d) = \begin{cases} 0, & d = s \\ 1, & d \neq s \end{cases}$$

Optimal decision strategy:

$$\delta^*(\vec{x}) = \arg\min_d \sum_s \underbrace{\ell(s,d)}_{0 \text{ if } d=s} P(s|\vec{x}) = \arg\min_d \sum_{s \neq d} P(s|\vec{x})$$

Obviously $\sum_s P(s|\vec{x}) = 1$, then:

$$P(d|\vec{x}) + \sum_{s \neq d} P(s|\vec{x}) = 1$$

Inserting into above:

$$\delta^*(\vec{x}) = \arg\min_d [1 - P(d|\vec{x})] = \arg\max_d P(d|\vec{x})$$

A special case - Bayesian *classification*

- Attribute vector $\vec{x} = (x_1, x_2, \dots)$: pixels 1, 2, ...
- **State set $\mathcal{S}$ = decision set $\mathcal{D} = \{0, 1, \dots 9\}$.**
- State = actual class, Decision = recognized class
- Loss function :

$$\ell(s,d) = \begin{cases} 0, & d = s \\ 1, & d \neq s \end{cases}$$

Optimal decision strategy:

$$\delta^*(\vec{x}) = \arg\min_d \sum_s \underbrace{\ell(s,d)}_{0 \text{ if } d=s} P(s|\vec{x}) = \arg\min_d \sum_{s \neq d} P(s|\vec{x})$$

Obviously $\sum_s P(s|\vec{x}) = 1$, then:

$$P(d|\vec{x}) + \sum_{s \neq d} P(s|\vec{x}) = 1$$

Inserting into above:

$$\delta^*(\vec{x}) = \arg\min_d [1 - P(d|\vec{x})] = \arg\max_d P(d|\vec{x})$$

A special case - Bayesian *classification*

- Attribute vector $\vec{x} = (x_1, x_2, \dots)$: pixels 1, 2, ...
- **State set $\mathcal{S}$ = decision set $\mathcal{D} = \{0, 1, \dots 9\}$.**
- State = actual class, Decision = recognized class
- Loss function :

$$\ell(s,d) = \begin{cases} 0, & d = s \\ 1, & d \neq s \end{cases}$$

Optimal decision strategy:

$$\delta^*(\vec{x}) = \arg\min_d \sum_s \underbrace{\ell(s,d)}_{0 \text{ if } d=s} P(s|\vec{x}) = \arg\min_d \sum_{s \neq d} P(s|\vec{x})$$

Obviously $\sum_s P(s|\vec{x}) = 1$, then:

$$P(d|\vec{x}) + \sum_{s \neq d} P(s|\vec{x}) = 1$$

Inserting into above:

$$\delta^*(\vec{x}) = \arg\min_d [1 - P(d|\vec{x})] = \arg\max_d P(d|\vec{x})$$

A special case - Bayesian *classification*

- Attribute vector $\vec{x} = (x_1, x_2, \dots)$: pixels 1, 2, ...
- **State set $\mathcal{S}$ = decision set $\mathcal{D} = \{0, 1, \dots 9\}$.**
- State = actual class, Decision = recognized class
- Loss function :

$$\ell(s,d) = \begin{cases} 0, & d = s \\ 1, & d \neq s \end{cases}$$

Optimal decision strategy:

$$\delta^*(\vec{x}) = \arg\min_d \sum_s \underbrace{\ell(s,d)}_{0 \text{ if } d=s} P(s|\vec{x}) = \arg\min_d \sum_{s \neq d} P(s|\vec{x})$$

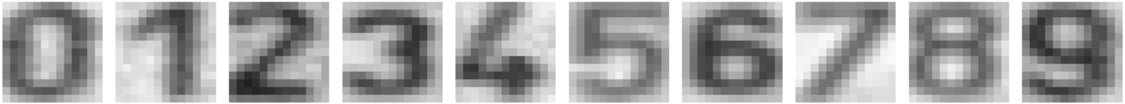
Obviously $\sum_s P(s|\vec{x}) = 1$, then:

$$P(d|\vec{x}) + \sum_{s \neq d} P(s|\vec{x}) = 1$$

Inserting into above:

$$\delta^*(\vec{x}) = \arg\min_d [1 - P(d|\vec{x})] = \arg\max_d P(d|\vec{x})$$

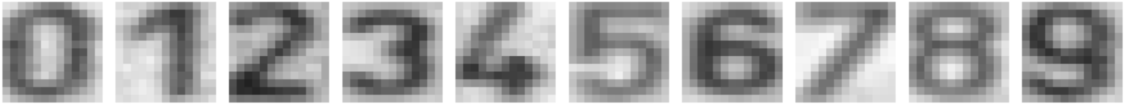
Example: Digit recognition/classification



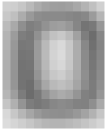
- Input: 8-bit image 13×13, pixel intensities $0 - 255$. (0 means black, 255 means white)
- Output: Digit $0 - 9$. Decision about the class, classification.
- Features: Pixel intensities ...

Decision/classification problem : What cipher is in the (query) image?

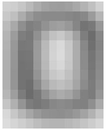
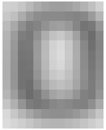
Example: Digit recognition/classification



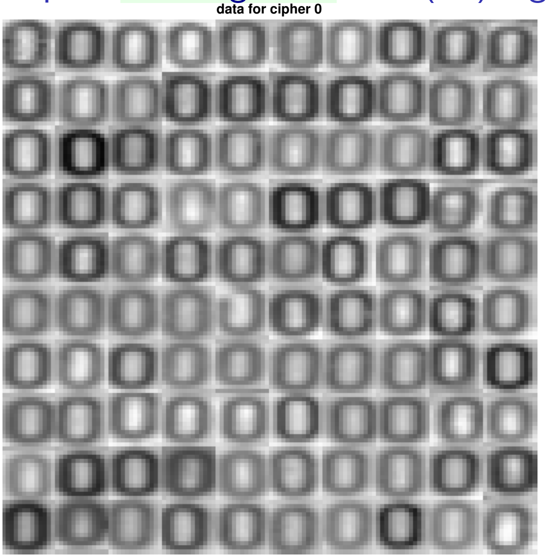
- Input: 8-bit image 13×13, pixel intensities $0 - 255$. (0 means black, 255 means white)
- Output: Digit $0 - 9$. Decision about the class, classification.
- Features: Pixel intensities ...

Decision/classification problem : What cipher is in the (query) image? 

Optimal (Bayes) Classification

$\delta^*($  $) = \arg\max_d P(d \mid$  $)$

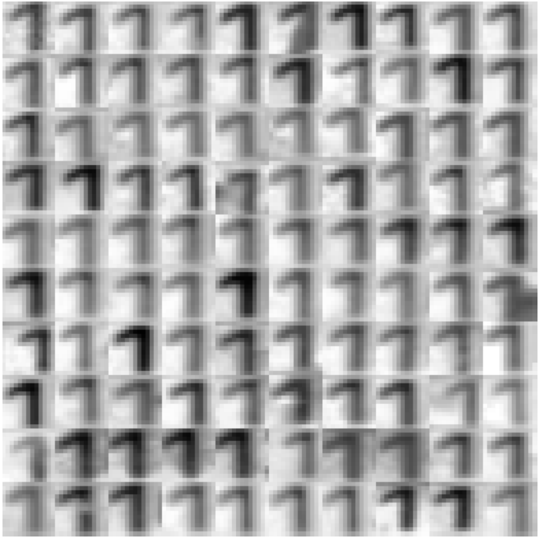
Machine Learnine: Prepare training data , let (an) algorithm learn itself



Training samples: $(\vec{x}_i, s = 0)$

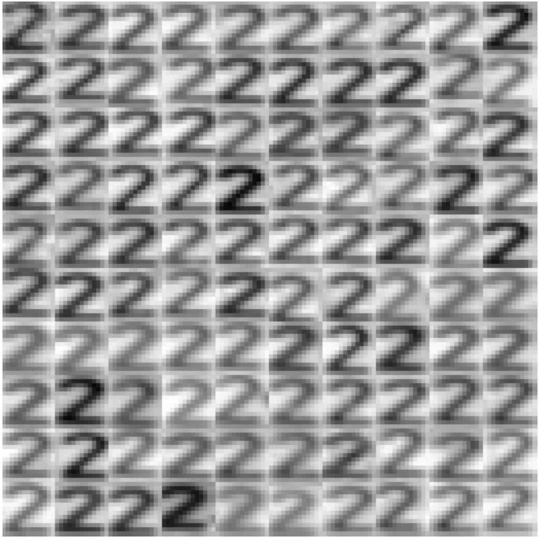
Machine Learnine: Prepare training data , let (an) algorithm learn itself

data for cipher 1



Training samples: $(\vec{x}_i, s = 1)$

Machine Learnine: Prepare training data , let (an) algorithm learn itself



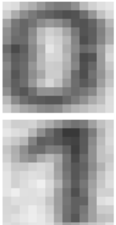
Training samples: $(\vec{x}_i, s = 2)$

Bayes classification in practice; $P(s|\vec{x}) =?$

- Usually, we are not given $P(s|\vec{x})$
- It has to be estimated from already classified examples – training data
- For discrete $\vec{x}$, training examples $(\vec{x}_1, s_1), (\vec{x}_2, s_2), \ldots (\vec{x}_l, s_l)$
  - every $(\vec{x}_i, s)$ is drawn independently from $P(\vec{x}, s)$, i.e. sample i does not depend on $1, \cdots, i-1$
  - so-called i.i.d (independent, identically distributed) multiset
- Without knowing anything about the distribution, a non-parametric estimate:

$$P(s|\vec{x}) = \frac{P(\vec{x}, s)}{P(\vec{x})} \approx \frac{\#\text{ examples where } \vec{x}_i = \vec{x} \textbf{ and } s_i = s}{\#\text{ examples where } \vec{x}_i = \vec{x}}$$

- Hard in practice:
  - To reliably estimate $P(s|\vec{x})$, the number of examples grows exponentially with the number of elements of $\vec{x}$.
    - e.g. with the number of pixels in images
    - curse of dimensionality
    - denominator often 0

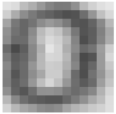


Bayes classification in practice; $P(s|\vec{x}) = ?$

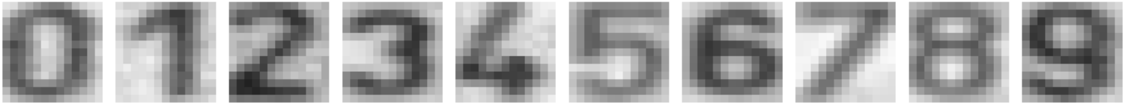
- Usually, we are not given $P(s|\vec{x})$
- It has to be estimated from already classified examples – training data
- For discrete $\vec{x}$, training examples $(\vec{x}_1, s_1), (\vec{x}_2, s_2), \ldots (\vec{x}_l, s_l)$
  - every $(\vec{x}_i, s)$ is drawn independently from $P(\vec{x}, s)$, i.e. sample i does not depend on $1, \cdots, i-1$
  - so-called i.i.d (independent, identically distributed) multiset
- Without knowing anything about the distribution, a non-parametric estimate:

$$P(s|\vec{x}) = \frac{P(\vec{x}, s)}{P(\vec{x})} \approx \frac{\#\text{ examples where } \vec{x}_i = \vec{x} \textbf{ and } s_i = s}{\#\text{ examples where } \vec{x}_i = \vec{x}}$$

- Hard in practice:
  - To reliably estimate $P(s|\vec{x})$, the number of examples grows exponentially with the number of elements of $\vec{x}$.
    - e.g. with the number of pixels in images
    - curse of dimensionality
    - denominator often 0



How many images?



8-bit image 13×13, pixel intensities $0 - 255$. (0 means black, 255 means white)

A: 169^{256}

B: 256^{169}

C: 13^{13}

D: 169×256

E: different quantity

Naïve Bayes classification

- For efficient classification we must thus rely on additional assumptions.
- In the exceptional case of statistical independence between components of $\vec{x}$ for each class s it holds

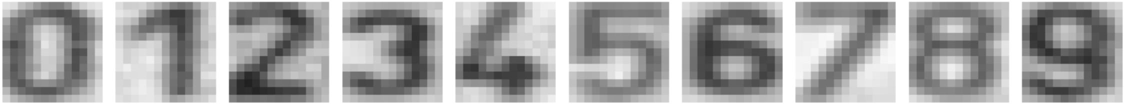
$$P(\vec{x}|s) = P(x[1]|s) \cdot P(x[2]|s) \cdot \ldots$$

- Use simple Bayes law and maximize:

$$P(s|\vec{x}) = \frac{P(\vec{x}|s)P(s)}{P(\vec{x})} = \frac{P(s)}{P(\vec{x})} P(x[1]|s) \cdot P(x[2]|s) \cdot \ldots =$$

- No combinatorial curse in estimating $P(s)$ and $P(x[i]|s)$ separately for each i and s.
- No need to estimate $P(\vec{x})$. (Why?)
- $P(s)$ may be provided apriori.
- naïve = when used despite statistical dependence

Example: Digit recognition/classification

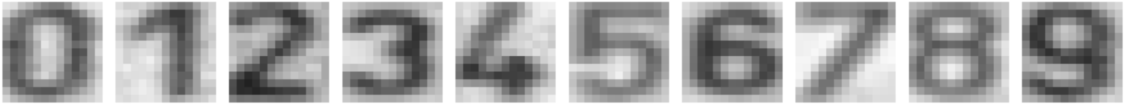


- Input: 8-bit image 13×13, pixel intensities $0 - 255$. (0 means black, 255 means white)
- Output: Digit $0 - 9$. Decision about the class, classification.
- Features: Pixel intensities . . .

Collect data , . . .

- $P(\vec{x})$. What is the dimension of $\vec{x}$? How many possible images?
- Learn $P(\vec{x}|s)$ per each class (digit).
- Classify $s^* = \text{argmax}_s P(s|\vec{x})$.

Example: Digit recognition/classification

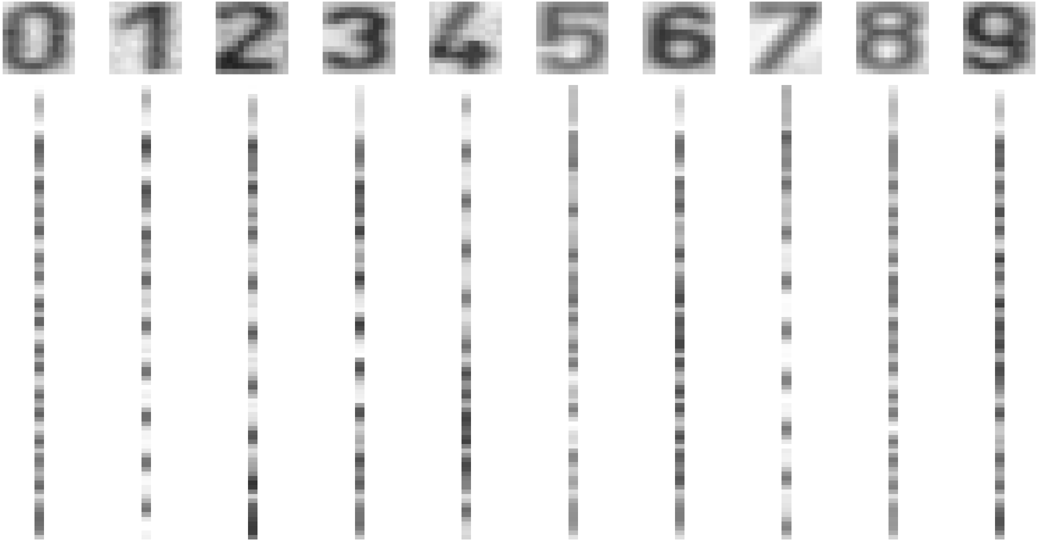


- Input: 8-bit image 13×13, pixel intensities $0 - 255$. (0 means black, 255 means white)
- Output: Digit $0 - 9$. Decision about the class, classification.
- Features: Pixel intensities ...

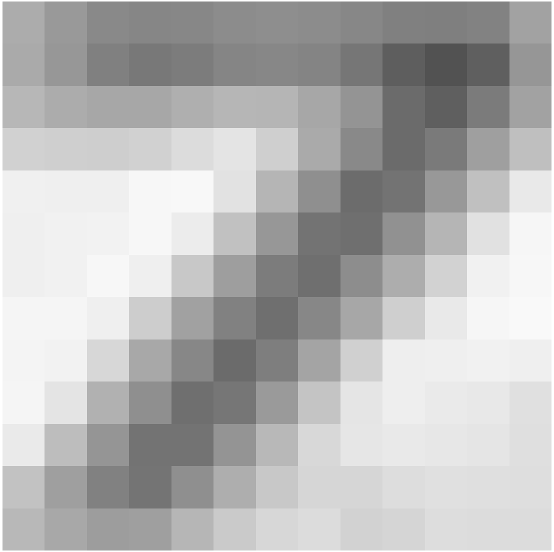
Collect data , ...

- $P(\vec{x})$. What is the dimension of $\vec{x}$? How many possible images?
- Learn $P(\vec{x}|s)$ per each class (digit).
- Classify $s^* = \text{argmax}_s P(s|\vec{x})$.

From images to $\vec{x}$



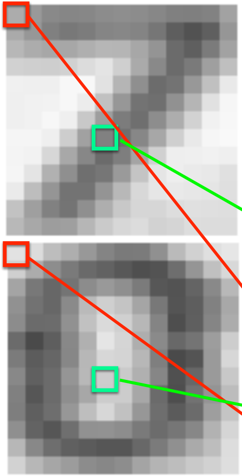
Conditional probabilities, likelihoods



- Apriori digit probabilities $P(s_k)$
- Likelihoods for pixels. $P(x_{r,c} = l_i | s_k)$

Conditional likelihoods

s_k	$P(s_k)$
1	0.1
2	0.1
3	0.1
4	0.1
5	0.1
6	0.1
7	0.1
8	0.1
9	0.1
0	0.1

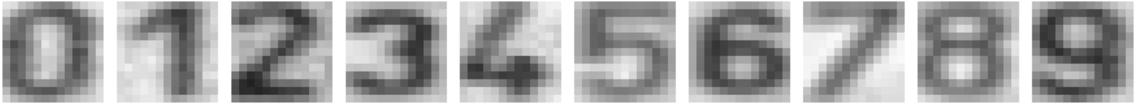


$P(x_{0,0} = 25|s_k)$ $P(x_{7,5} = 25|s_k)$

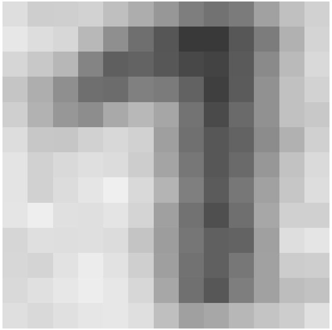
s_k	$P(x_{0,0} = 25\|s_k)$
1	0.0009
2	0.05
3	0.00006
4	0.0005
5	0.07
6	0.0007
7	0.1
8	0.0006
9	0
0	0

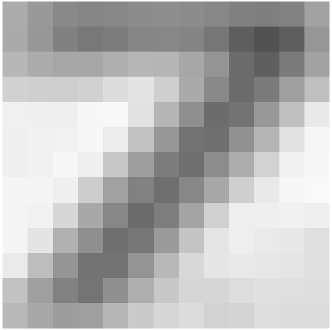
s_k	$P(x_{7,5} = 25\|s_k)$
1	0.0009
2	0.05
3	0.00006
4	0.0005
5	0.004
6	0.07
7	0.1
8	0.006
9	0.08
0	0.005

Unseen events



Images 13×13, intensities $0 - 255$, 100 exemplars per each class.

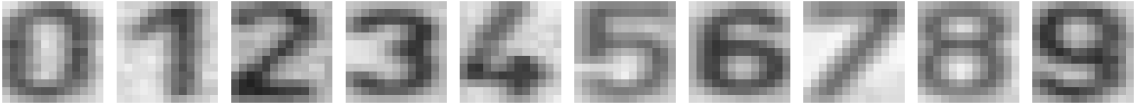




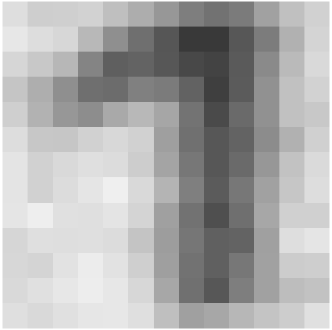
$$
\begin{aligned}
\vdots &= \vdots \\
P(x_{0,0} = 100 \mid s = 7) &= 0.05 \\
P(x_{0,0} = 101 \mid s = 7) &= 0 \\
P(x_{0,0} = 102 \mid s = 7) &= 0.06 \\
\vdots &= \vdots
\end{aligned}
$$

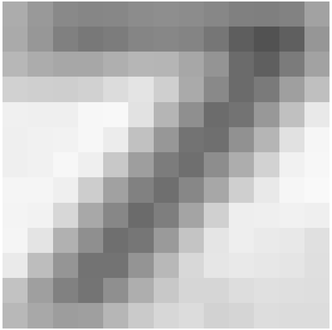
A new (not in training) query image with $x_{0,0} = 101$. How would you classify?

Unseen events



Images 13×13, intensities $0 - 255$, 100 exemplars per each class.





$$
\begin{aligned}
\vdots \quad &= \quad \vdots \\
P(x_{0,0} = 100 \mid s = 7) \quad &= \quad 0.05 \\
P(x_{0,0} = 101 \mid s = 7) \quad &= \quad 0 \\
P(x_{0,0} = 102 \mid s = 7) \quad &= \quad 0.06 \\
\vdots \quad &= \quad \vdots
\end{aligned}
$$

A new (not in training) query image with $x_{0,0} = 101$. How would you classify?

Unseen event, how to decide?

A new (not in training) query image with $x_{0,0} = 101$. How would you classify?

$$P(x_{0,0} = 101 \mid s_j) = 0, \text{ for all classes}$$

Laplace smoothing ("additive smoothing")

Think about a particular pixel with intensity x

$$P(x) = \frac{\text{count}(x)}{\text{total samples}}$$

Problem: $\text{count}(x) = 0$

Pretend you see the (any) sample one more time.

$$P_{\text{LAP}}(x) = \frac{c(x) + 1}{\sum_x [c(x) + 1]}$$

$$P_{\text{LAP}}(x) = \frac{c(x) + 1}{N + |X|}$$

where N is the number of (total) observations; $|X|$ is the number of possible values X can take (cardinality).

Laplace smoothing ("additive smoothing")

Think about a particular pixel with intensity x

$$P(x) = \frac{\text{count}(x)}{\text{total samples}}$$

Problem: $\text{count}(x) = 0$
Pretend you see the (any) sample one more time.

$$P_{\text{LAP}}(x) = \frac{c(x) + 1}{\sum_x [c(x) + 1]}$$

$$P_{\text{LAP}}(x) = \frac{c(x) + 1}{N + |X|}$$

where N is the number of (total) observations; $|X|$ is the number of possible values X can take (cardinality).

Laplace smoothing ("additive smoothing")

Think about a particular pixel with intensity x

$$P(x) = \frac{\text{count}(x)}{\text{total samples}}$$

Problem: $\text{count}(x) = 0$
Pretend you see the (any) sample one more time.

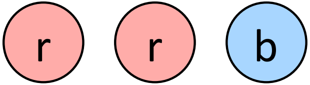
$$P_{\text{LAP}}(x) = \frac{c(x) + 1}{\sum_x [c(x) + 1]}$$

$$P_{\text{LAP}}(x) = \frac{c(x) + 1}{N + |X|}$$

where N is the number of (total) observations; $|X|$ is the number of possible values X can take (cardinality).

$$P_{\text{LAP}}(x) = \frac{c(x)+1}{\sum_x [c(x)+1]} = \frac{c(x)+1}{N+|X|} = ?$$

Observation:



What is $P_{\text{LAP}}(X = \text{red})$ and $P_{\text{LAP}}(X = \text{blue})$?

A: $P_{\text{LAP}}(X = \text{red}) = 7/10$, $P_{\text{LAP}}(X = \text{blue}) = 3/10$

B: $P_{\text{LAP}}(X = \text{red}) = 2/3$, $P_{\text{LAP}}(X = \text{blue}) = 1/3$

C: $P_{\text{LAP}}(X = \text{red}) = 3/5$, $P_{\text{LAP}}(X = \text{blue}) = 2/5$

D: None of the above.

Laplace smoothing - as a hyperparameter k

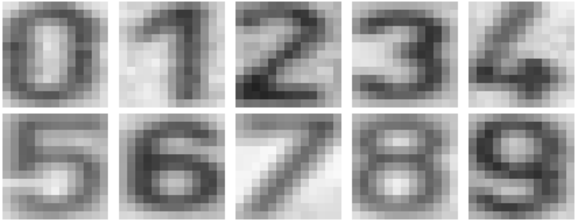
Pretend you see every sample k extra times:

$$P_{\text{LAP}}(x) = \frac{c(x) + k}{\sum_x [c(x) + k]}$$

$$P_{\text{LAP}}(x) = \frac{c(x) + k}{N + k|X|}$$

For conditional, smooth each condition independently

$$P_{\text{LAP}}(x|s) = \frac{c(x, s) + k}{c(s) + k|X|}$$



What is $|X|$ equal to?

A: 10

B: 2

C: 256

D: None of the above

Laplace smoothing - as a hyperparameter k

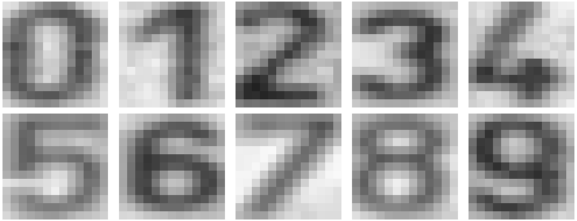
Pretend you see every sample k extra times:

$$P_{\text{LAP}}(x) = \frac{c(x) + k}{\sum_x [c(x) + k]}$$

$$P_{\text{LAP}}(x) = \frac{c(x) + k}{N + k|X|}$$

For conditional, smooth each condition independently

$$P_{\text{LAP}}(x|s) = \frac{c(x,s) + k}{c(s) + k|X|}$$



What is $|X|$ equal to?

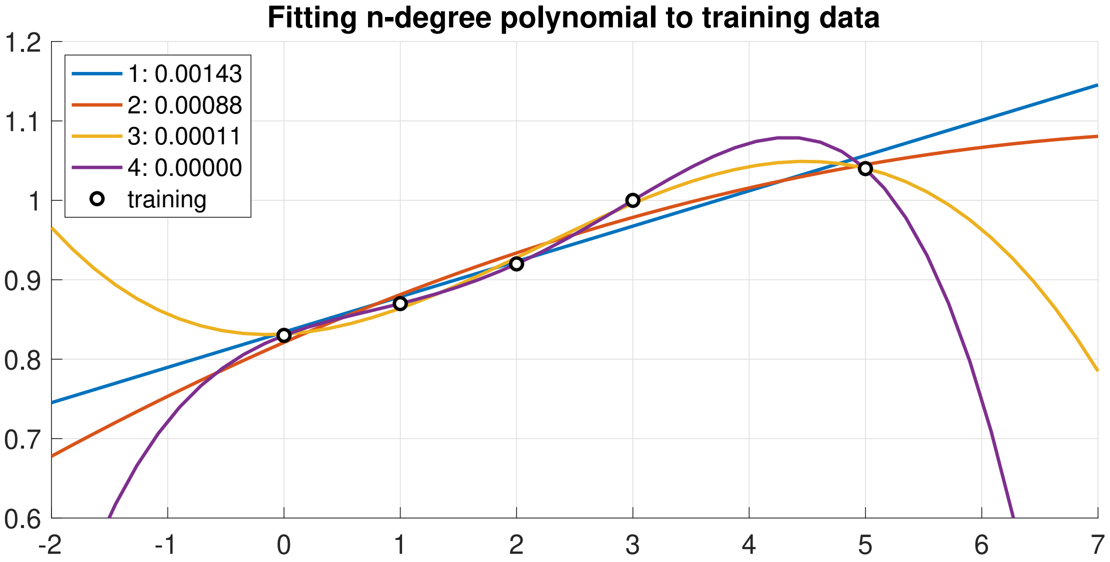
A: 10

B: 2

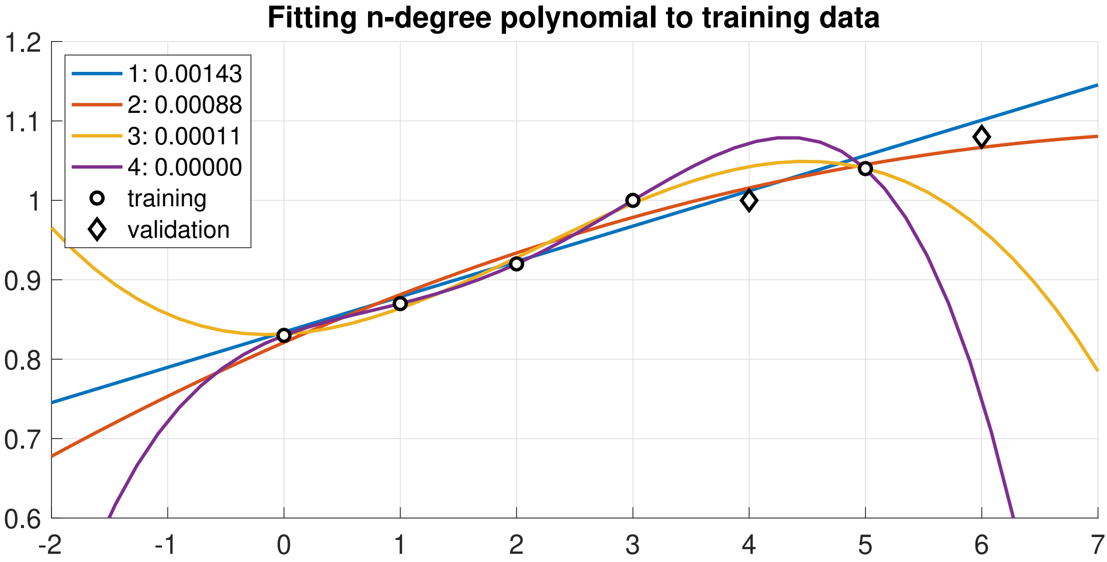
C: 256

D: None of the above

What is the right degree of polynomial (hyperparameter of a regressor)

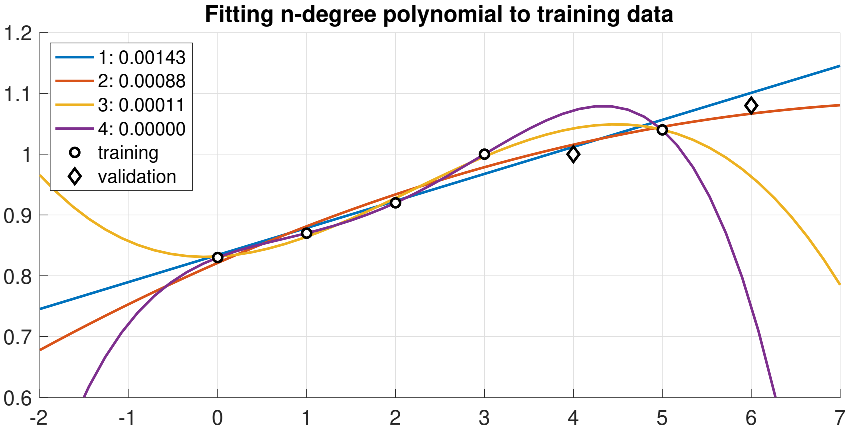


What is the right degree of polynomial (hyperparameter of a regressor)



Generalization and overfitting

- Data: training, validating, testing . Wanted classifier performs well on what data?
- Overfitting: too close to training, poor on testing.



Training and testing

Data labeled instances.

- Training set
- Held-out (validation) set
- Testing set.

Features : Attribute-value pairs.

Learning cycle:

- **Learn** parameters (e.g. probabilities) on training set.
- **Tune** hyperparameters on held-out (validation) set.
- **Evaluate** performance on testing set.

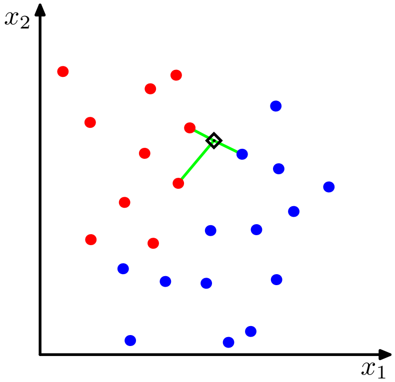
$K-$ Nearest Neighbor and Bayes $j^* = \text{argmax}_j P(s_j|\vec{x})$

Assume data:

- N samples $\vec{x}$ in total.
- N_j samples in s_j class. Hence, $\sum_j N_j = N$.

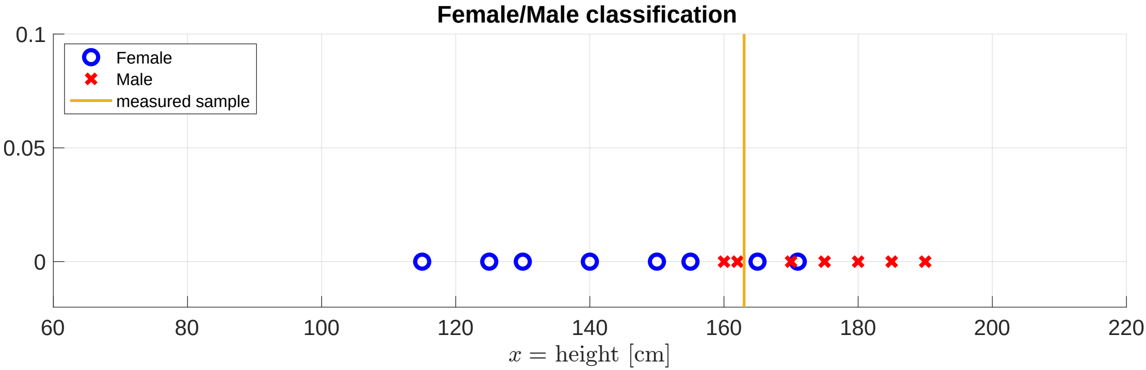
We want classify to $\vec{x}$. We draw a circle (hypher-sphere) centered at $\vec{x}$ containing K points irrespective of class. V is the volume of this sphere. $P(s_j|\vec{x}) = ?$

$$P(s_j|\vec{x}) = \frac{P(\vec{x}|s_j)P(s_j)}{P(\vec{x})}$$



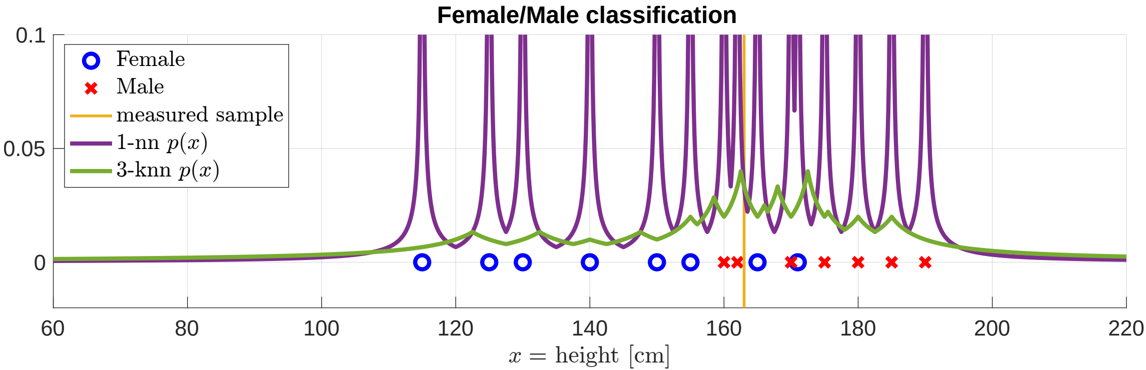
(a)

Female/male classification based on height. N data points available.



Ignore the y axis. A new measurement comes, $x = 163\,\text{cm}$. Female or male?

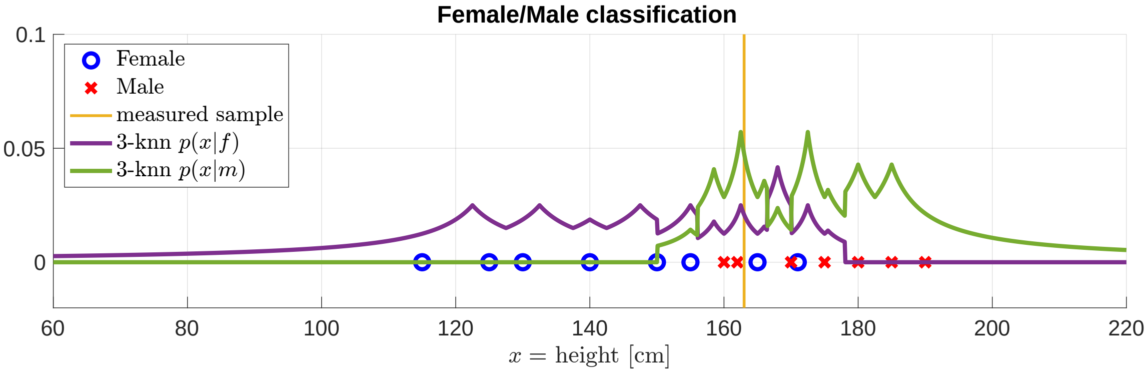
$K-$NN $p(x)$ estimate



$$p(x) = \frac{K}{NV}$$

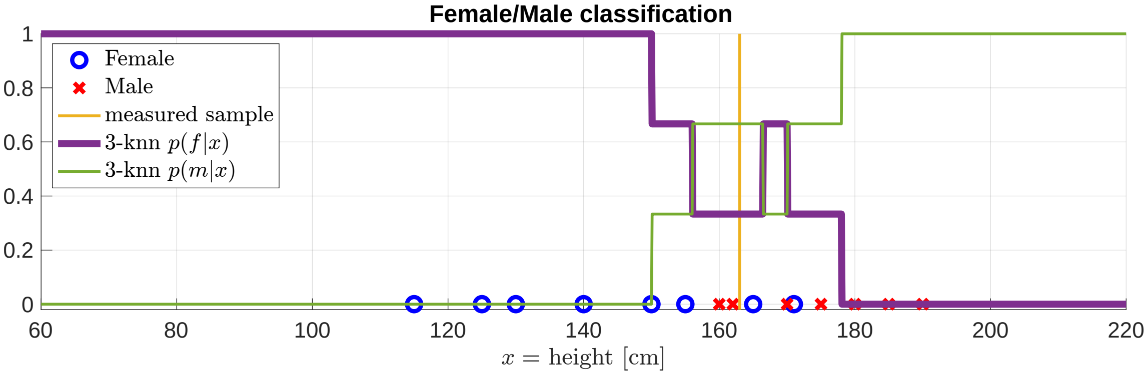
$V = 2r_k(x)$, where $r_k(x)$ is the distance of $k-$th nearest data point to x

$K-$NN $p(x|s_j)$ estimates



$$p(x|s_j) = \frac{K_j}{N_j V}$$

K−NN $p(s_j|x)$ posteriors



$$p(s_j|x) = \frac{p(x|s_j)p(s_j)}{p(x)}$$

Volume in $k - NN$ in higher dimensions

Complement slide, for the sake of completeness. The decision rule $P(s_j|x) = N_j/N$ is the same for all dimensions.

$$P(\vec{x}) = \frac{K}{NV}$$

$$V = V_d R_k^d(\vec{x})$$

$R_k(\vec{x})$ - distance from $\vec{x}$ to its $k-$th nearest neighbour point (radius)

$$V_d = \frac{\pi^{d/2}}{\Gamma(d/2+1)}$$

volume od unit $d-$dimensional sphere,
Γ denotes gamma function.
$V_1 = 2, V_2 = \pi, V_3 = \frac{4}{3}\pi$

Precision and Recall, and . . .

Consider digit detection (is there a digit?) or SPAM/HAM, Male/Female classification

Recall :

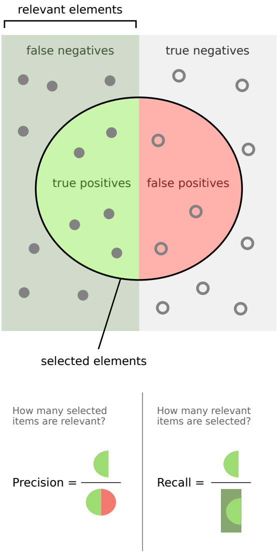
- How many relevant items are selected?
- Are we missing some items?
- Also called: True positive rate (TPR), sensitivity, hit rate . . .

Precision

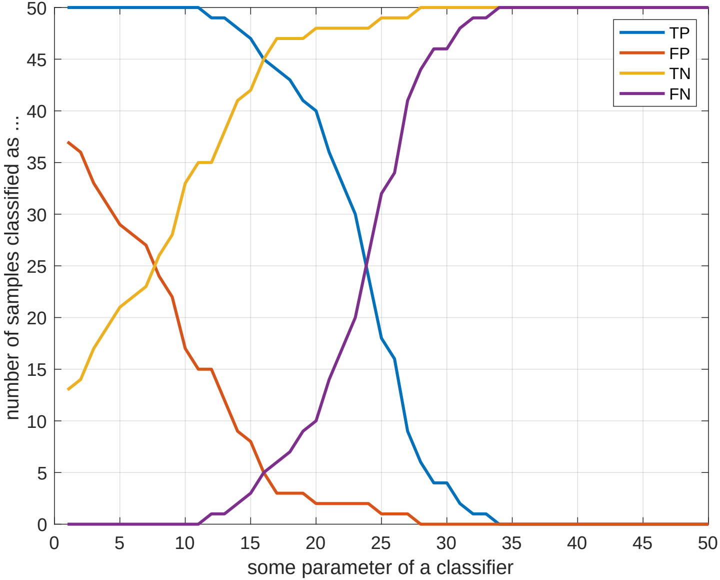
- How many selected items are relevant?
- Also called: Positive predictive value

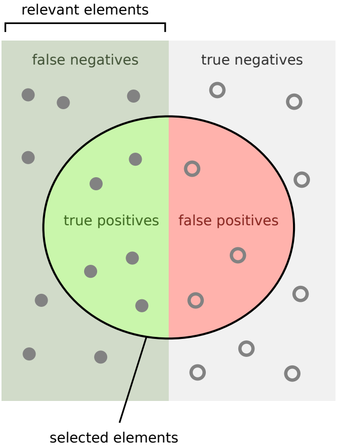
False positive rate (FPR)

- Probability of false alarm



Studying a classifier . . .





How many data samples x_i?

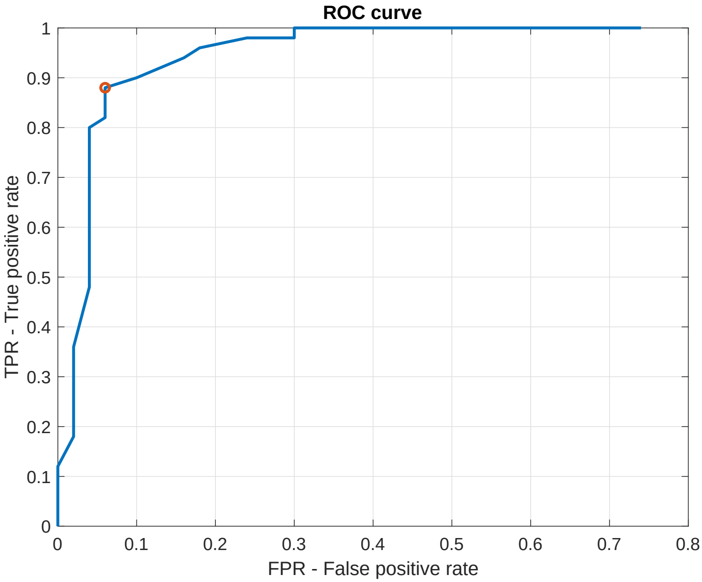
A 50

B 100

C 150

D 200

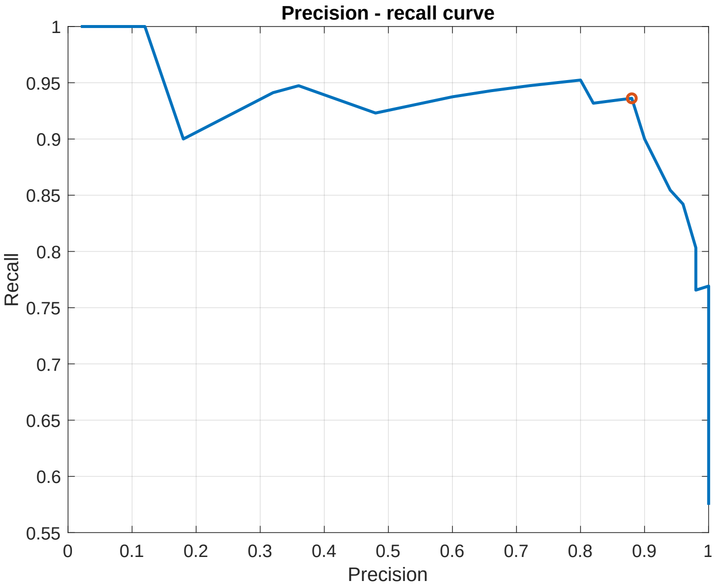
ROC – Receiver operating characteristics curve



$$\mathrm{TPR} = \frac{\mathrm{TP}}{P} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

$$\mathrm{FPR} = \frac{\mathrm{FP}}{N} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}$$

Precision – recall curve



$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

How to evaluate a multi-class classifier? Confusion table

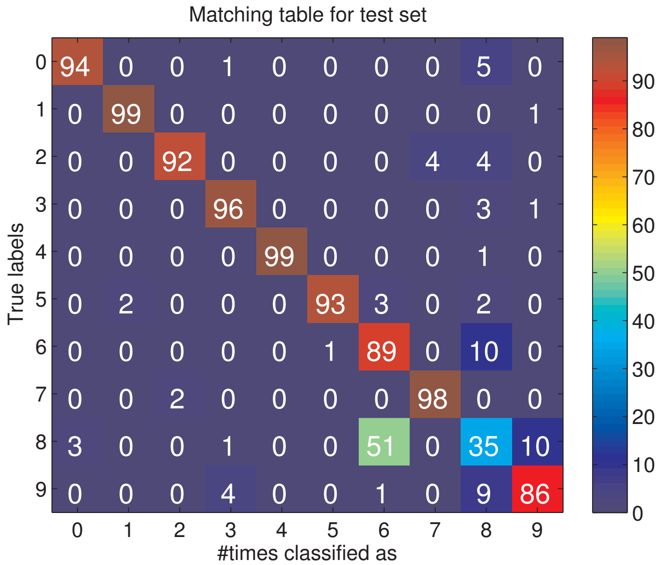


Figure from [6]

Product of many small numbers . . .

$$P(s|\vec{x}) = \frac{P(\vec{x}|s)P(s)}{P(\vec{x})} = \frac{P(s)}{P(\vec{x})} P(x[1]|s) \cdot P(x[2]|s) \cdot \ldots$$

$P(\vec{x})$ not needed, ...

$$\log(P(x[1]|s)P(x[2]|s)\cdots) = \log(P(x[1]|s)) + \log(P(x[2]|s)) + \cdots$$

Product of many small numbers ...

$$P(s|\vec{x}) = \frac{P(\vec{x}|s)P(s)}{P(\vec{x})} = \frac{P(s)}{P(\vec{x})} P(x[1]|s) \cdot P(x[2]|s) \cdot \ldots$$

$P(\vec{x})$ not needed, ...

$$\log(P(x[1]|s)P(x[2]|s)\cdots) = \log(P(x[1]|s)) + \log(P(x[2]|s)) + \cdots$$

References I

Further reading: Chapter 13 and 14 of [5]. Books [1] and [3] are classical textbooks in the field of pattern recognition and machine learning. This lecture has been also inspired by the 21st lecture of CS 188 at http://ai.berkeley.edu (e.g., Laplace smoothing). Many Matlab figures created with the help of [4].

[1] Christopher M. Bishop.
Pattern Recognition and Machine Learning.
Springer Science+Bussiness Media, New York, NY, 2006.
https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf.

[2] Yen-Chi Chen.
Lecture 7: Density estimation: k-nearest neighbor and basis approach.
Lecture within STAT 425: Introduction to Nonparametric Statistics, 2018.
http://faculty.washington.edu/yenchic/18W_425/Lec7_knn_basis.pdf.

References II

[3] Richard O. Duda, Peter E. Hart, and David G. Stork.
Pattern Classification.
John Wiley & Sons, 2nd edition, 2001.

[4] Vojtěch Franc and Václav Hlaváč.
Statistical pattern recognition toolbox.
http://cmp.felk.cvut.cz/cmp/software/stprtool/index.html.

[5] Stuart Russell and Peter Norvig.
Artificial Intelligence: A Modern Approach.
Prentice Hall, 3rd edition, 2010.
http://aima.cs.berkeley.edu/.

References III

[6] Tomáš Svoboda, Jan Kybic, and Hlaváč Václav.
Image Processing, Analysis and Machine Vision — A MATLAB Companion.
Thomson, Toronto, Canada, 1st edition, September 2007.
http://visionbook.felk.cvut.cz/.