

# Classifiers: Naïve Bayes, evaluation

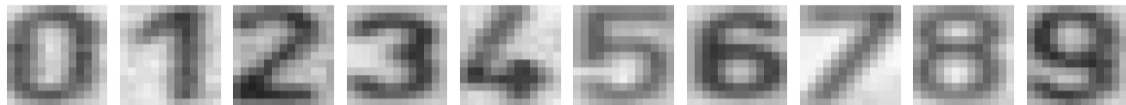
Tomáš Svoboda and Matěj Hoffmann

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

April 27, 2022

## Example: Digit recognition/classification



- ▶ **Input:** 8-bit image  $13 \times 13$ , 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



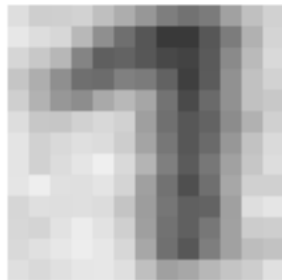
- ▶ **Input:** 8-bit image  $13 \times 13$ , pixel intensities 0 – 255. (0 means black, 255 means white)
- ▶ **Output:** Digit 0 – 9. Decision about the class, classification.
- ▶ **Features:** Pixel intensities ...

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



## Classification as a special case of statistical decision theory

- ▶ Attribute vector  $\vec{x} = [x_1, x_2, \dots]^T$ : pixels 1, 2, ...
- ▶ **State set  $\mathcal{S}$  = decision set  $\mathcal{D} = \{0, 1, \dots, 9\}$ .**
- ▶ **State = actual class, Decision = recognized class**
- ▶ **Loss function:** 
$$l(s, d) = \begin{cases} 0, & d = s \\ 1, & d \neq s \end{cases}$$



Optimal decision strategy:

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

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:

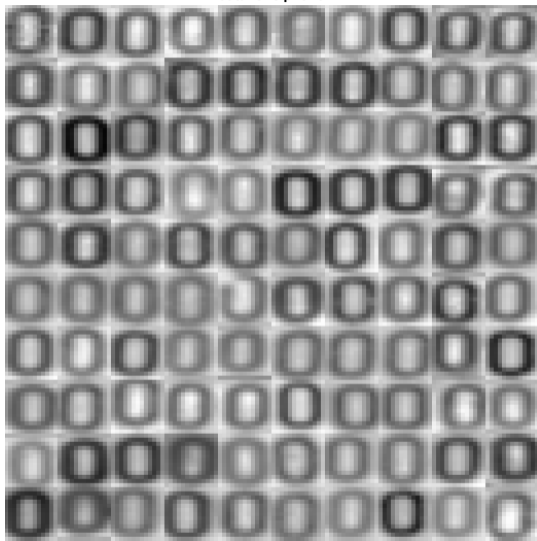
$$\delta^*(\vec{x}) = \arg \min_d \left( 1 - P(d|\vec{x}) \right) = \arg \max_d P(d|\vec{x})$$

## Optimal (Bayes) Classification

$$\delta^*(\text{img}) = \arg \max_d P(d | \text{img})$$

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

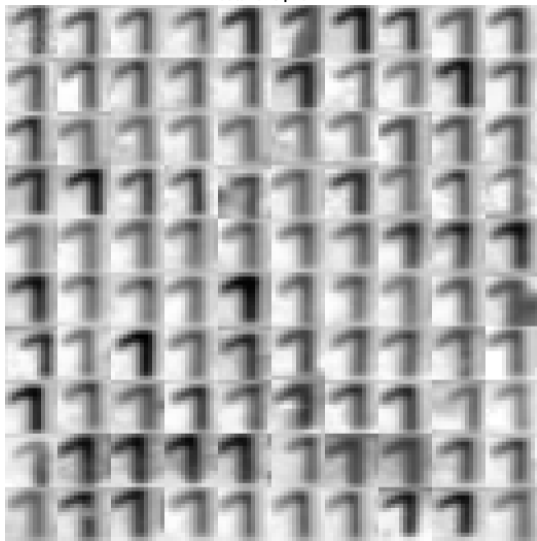
data for cipher 0



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

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

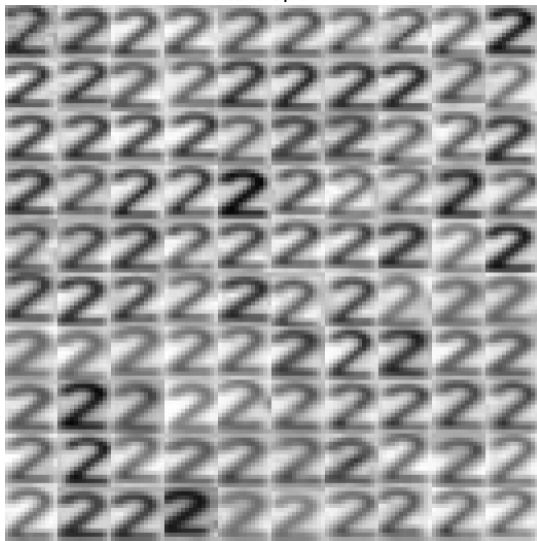
data for cipher 1



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

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

data for cipher 2



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



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

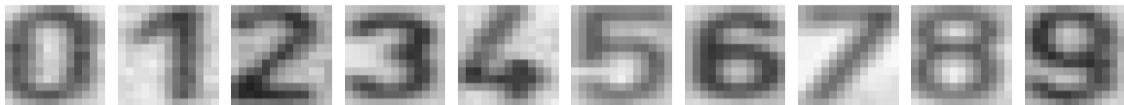
- ▶ Usually, we are not given  $P(s|\vec{x})$
- ▶ It has to be estimated from already classified examples – training data
- ▶ For discrete  $\vec{x}$ , training examples  $(\vec{x}_1, s_1), (\vec{x}_2, s_2), \dots, (\vec{x}_I, s_I)$ 
  - ▶ every  $(\vec{x}_i, s_i)$  is drawn independently from  $P(\vec{x},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

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

- ▶ 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  $13 \times 13$ , pixel intensities 0 – 255. (0 means black, 255 means white)

- A:  $169^{256}$
- B:  $256^{169}$
- C:  $13^{13}$
- D:  $169 \times 256$
- 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

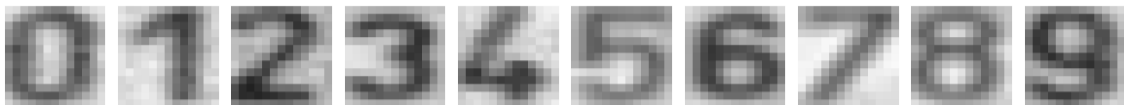
$$P(\vec{x}|s) = P(x[1]|s) \cdot P(x[2]|s) \cdot \dots$$

- ▶ Use simple Bayes law and maximize:

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

- ▶ 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

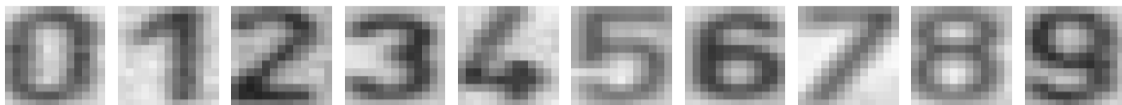


- ▶ **Input:** 8-bit image  $13 \times 13$ , pixel intensities 0 – 255. (0 means black, 255 means white)
- ▶ **Output:** Digit 0 – 9. Decision about the class, classification.
- ▶ **Features:** Pixel intensities ...

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  $s^* = \operatorname{argmax}_s P(s|\vec{x})$ .

## Example: Digit recognition/classification



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

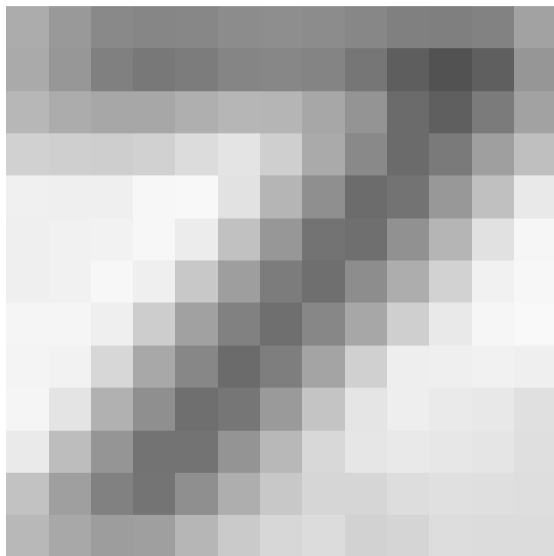
Collect **data** , ...

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

From images to  $\vec{x}$

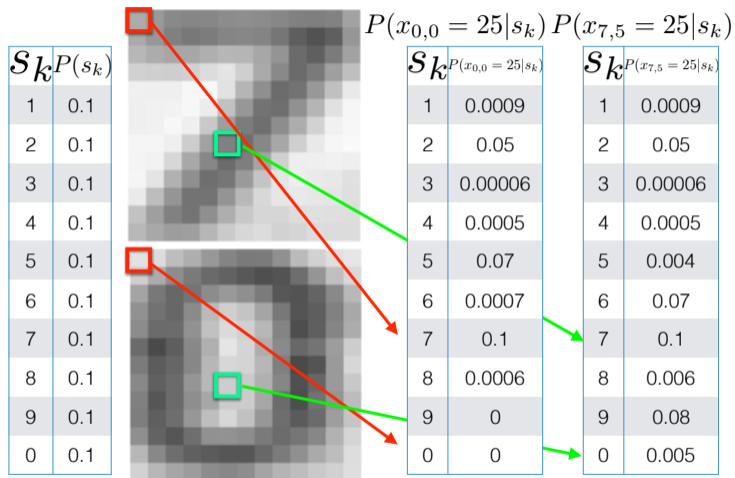


## Conditional probabilities, likelihoods



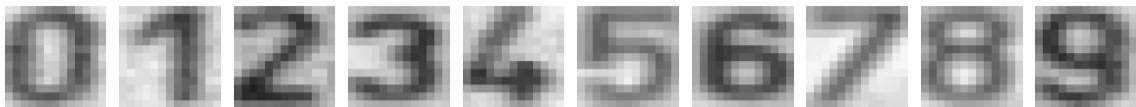
- ▶ Apriori digit probabilities  $P(s_k)$
- ▶ Likelihoods for pixels.  $P(x_{r,c} = l_i | s_k)$

# Conditional likelihoods

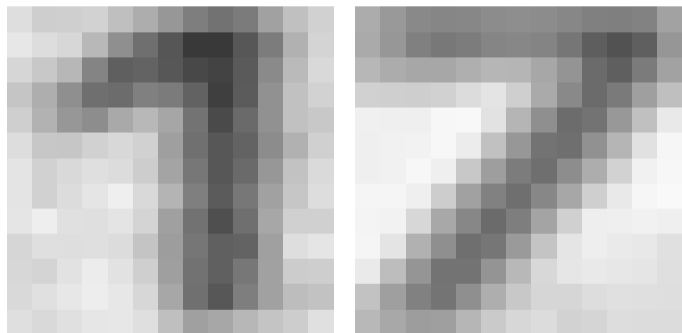




## Unseen events



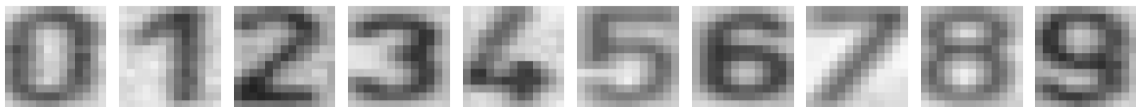
Images  $13 \times 13$ , intensities  $0 - 255$ , 100 exemplars per each class.



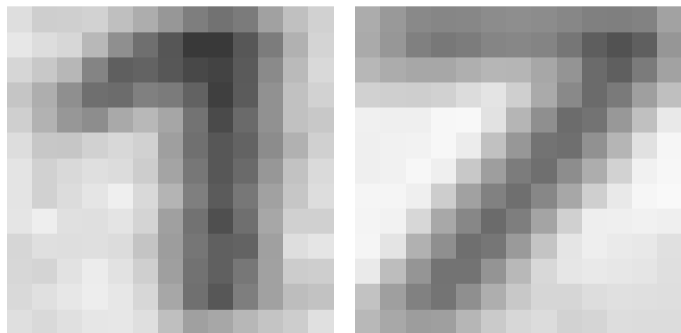
$$\begin{aligned} & \vdots = \vdots \\ P(x_{0,0} = 100 \mid s = 7) &= 0.05 \\ P(x_{0,0} = 101 \mid s = 7) &= 0 \\ P(x_{0,0} = 102 \mid s = 7) &= 0.06 \\ & \vdots = \vdots \end{aligned}$$

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

## Unseen events



Images  $13 \times 13$ , 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 event, how to decide?

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

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

## 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.

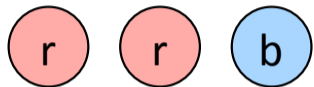
$$P_{\text{LAP}}(x) = \frac{c(x) + 1}{\sum_x [c(x) + 1]}$$

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

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

$$P_{\text{LAP}}(X) = ?$$

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 $k$

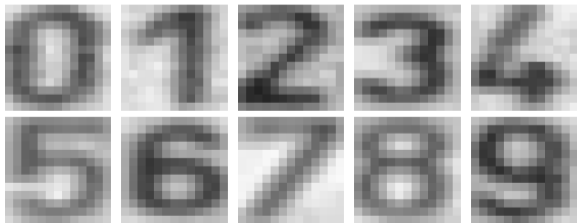
Pretend you see every sample  $k$  extra times:

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

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

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 $k$

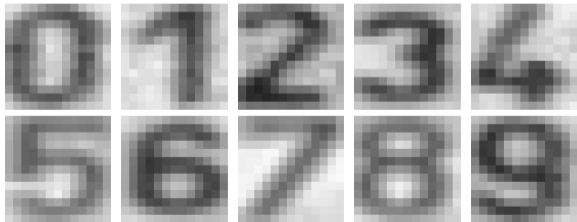
Pretend you see every sample  $k$  extra times:

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

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

For conditional, smooth each condition independently

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

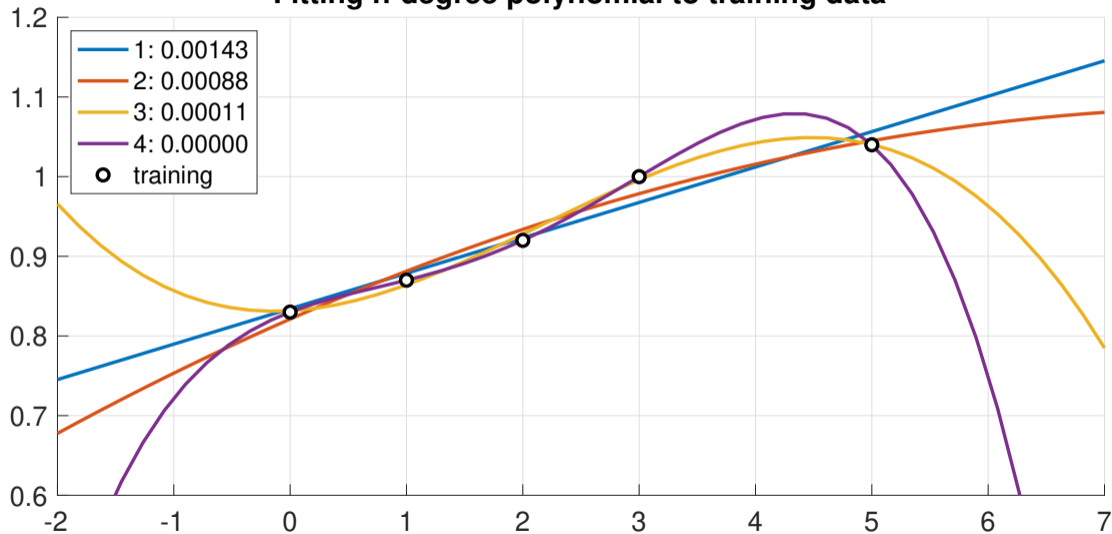


What is  $|X|$  equal to?

- A: 10
- B: 2
- C: 256
- D: None of the above

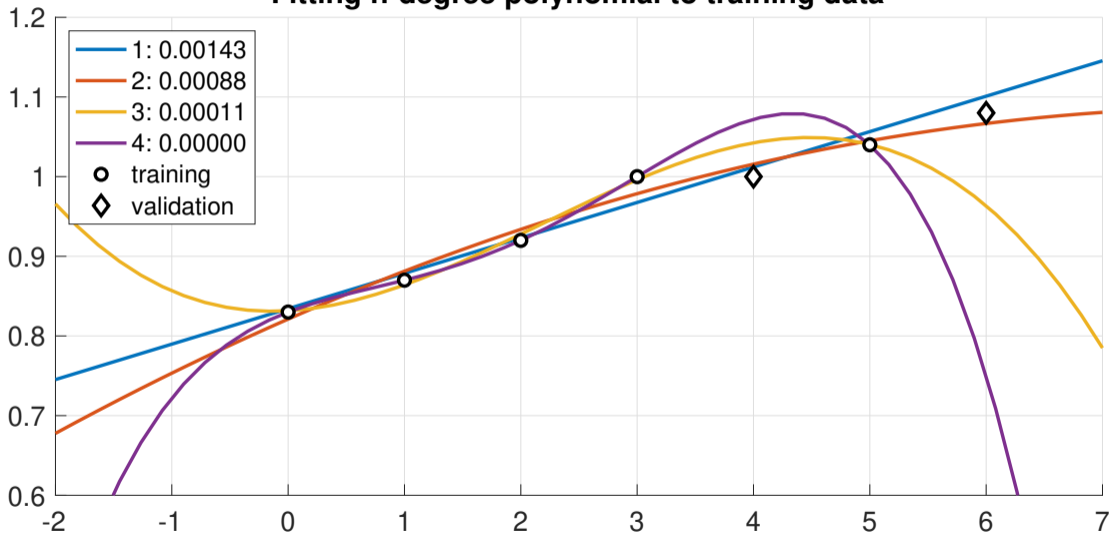
# What is the right degree of polynomial (hyperparameter of a regressor)

## Fitting n-degree polynomial to training data



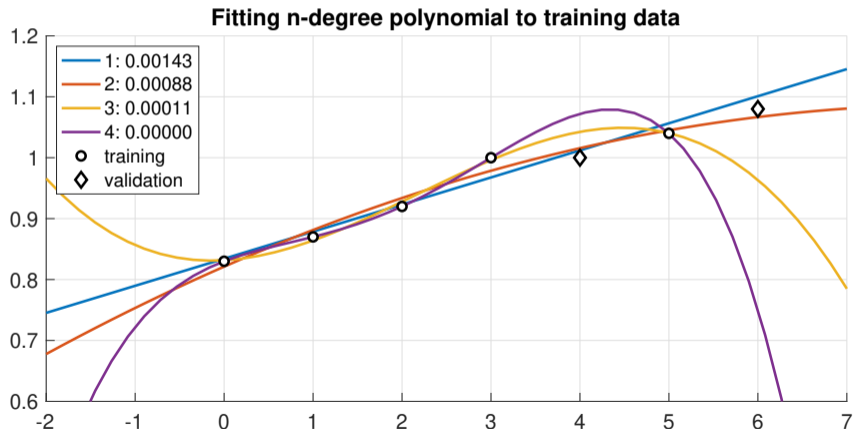
# What is the right degree of polynomial (hyperparameter of a regressor)

## Fitting n-degree polynomial to training data



# 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.



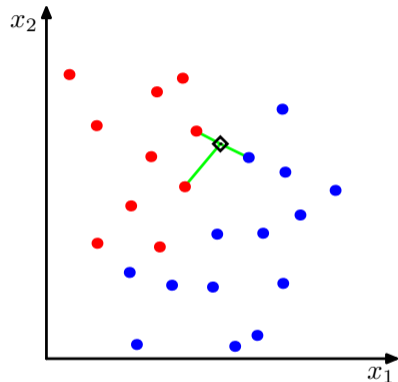
## $K$ - Nearest Neighbor and Bayes $j^* = \operatorname{argmax}_j P(s_j|\vec{x})$

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 (hyper-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)

## $k - NN$ for non-parametric density estimation

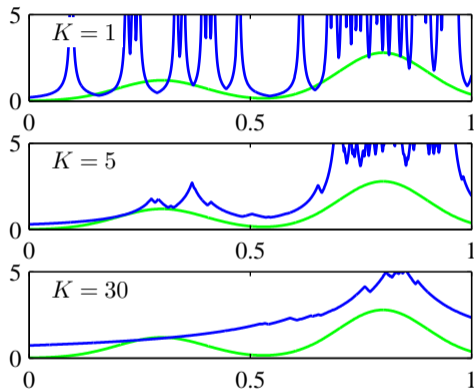
$$P(\vec{x}) = \frac{K}{NV}$$

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

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

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

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



# How to evaluate a classifier? Confusion table

Matching table for test set

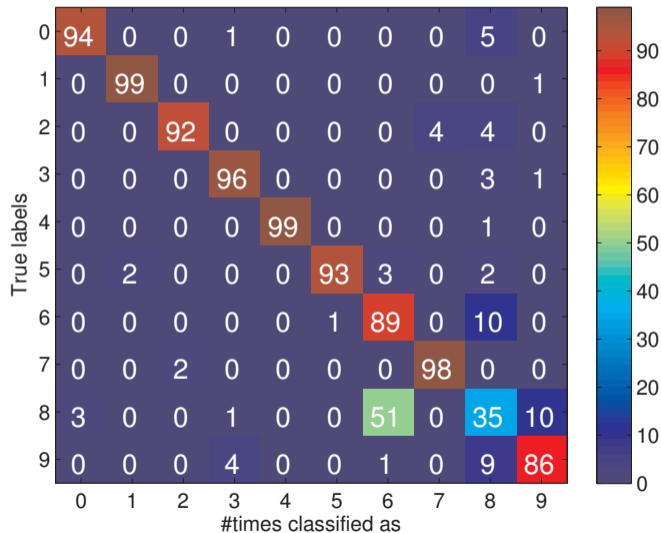


Figure from [6]



# Precision and Recall, and ...

Consider digit **detection** (is there a digit?) or SPAM/HAM classification.

**Recall** :

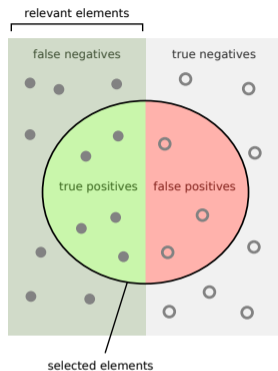
- ▶ How many relevant items are selected?
- ▶ Are we missing some items?
- ▶ Also called: **True positive rate** (TPR), sensitivity, hit rate ...

**Precision**

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

**False positive rate** (FPR)

- ▶ Probability of false alarm



How many selected items are relevant?



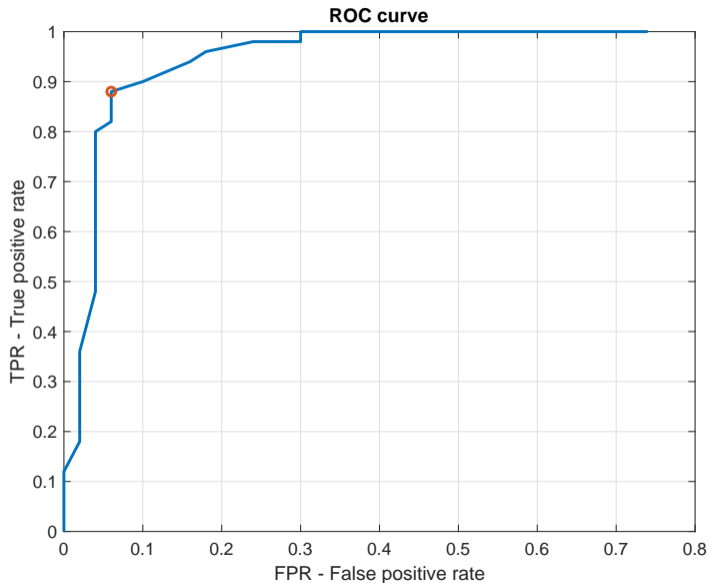
$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

How many relevant items are selected?



$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

# ROC – Receiver operating characteristics curve



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

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

## Product of many small numbers . . .

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

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

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

## Product of many small numbers . . .

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

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

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

## 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+Business 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](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, 1<sup>st</sup> edition, September 2007.

<http://visionbook.felk.cvut.cz/>.