UNIVERSITA' POLITECNICA DELLE MARCHE

DII-DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

APPROXIMATION OF THE GRADIENT OF THE ERROR PROBABILITY FOR VECTOR QUANTIZERS

CLAUDIA DIAMANTINI, LAURA GENGA, DOMENICO POTENA

{DIAMANTINI, GENGA, POTENA} @DII.UNIVPM.IT

Contents

- Bvq: minimum risk classification
- Bvq2
  - Approximation(two code vectors)
  - Problem of Voronoi cusp
- Bvq3
  - better approximation(three code vectors)
- Experimental evaluation
- Conclusion and future work

Labeled Vector Quantizer

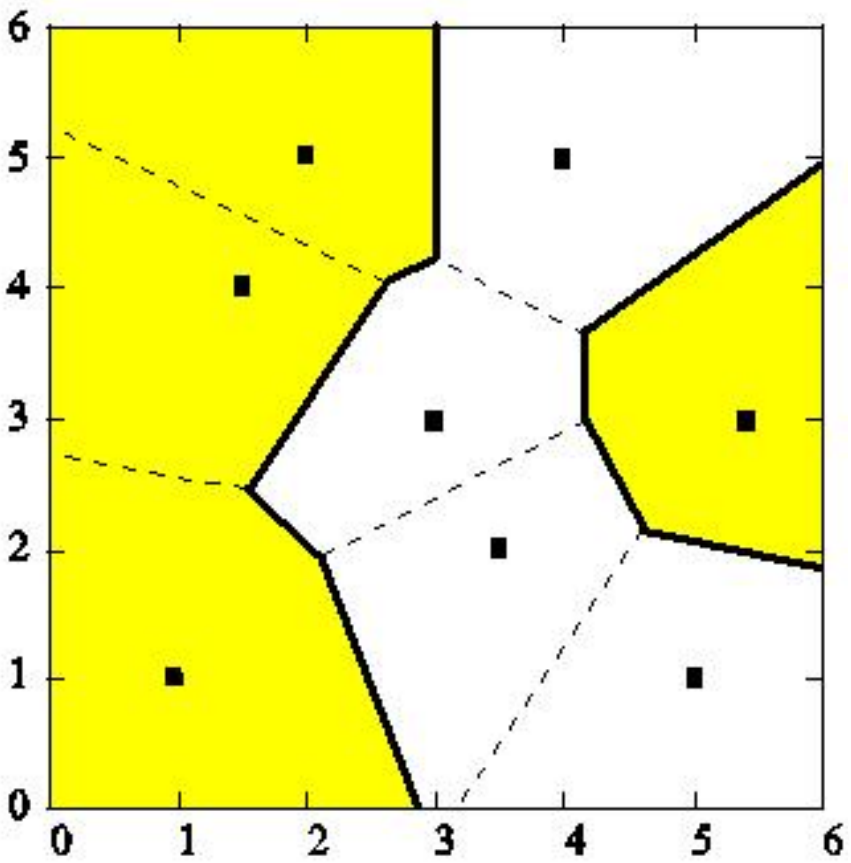
- A NN-VQ is a mapping

$$\Omega : \Re^n -> M$$

M={m_1, m_2,...., m_n} is the *codebook*

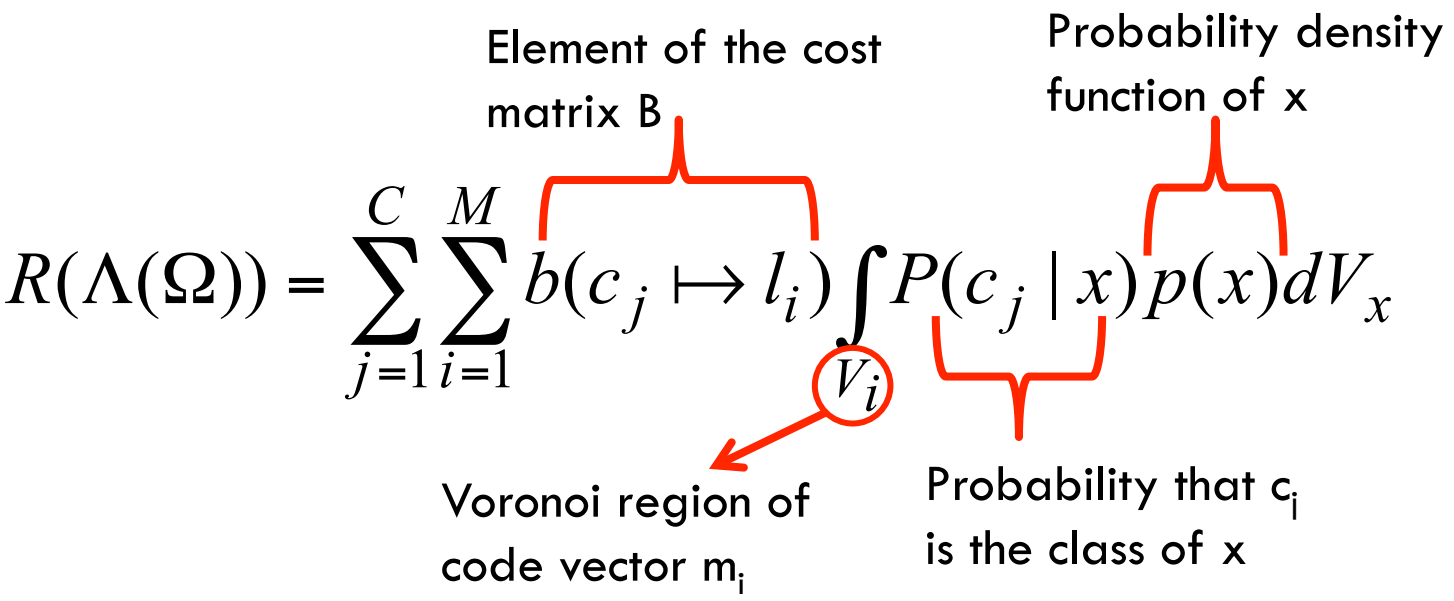
- A LVQ is a VQ equipped by a further mapping

$$\lambda(\Omega(x)) = l_i$$



The average misclassification risk

LVQ misclassification risk



Minimization of the risk

Goal: minimizing the risk by modifying the position of the code vectors

Code vector m_i at k-th iteration

Step size

$$m_i^{(k+1)} = m_i^{(k)} - \gamma^{(k)} \nabla_i R^{(k)}(\Phi^{(k)})$$

$$i = 1,2,....,M; \qquad k = 0,1,....$$

Gradient of the average risk

Parzen estimate

$$R(\Lambda(\Omega)) = \sum_{j=1}^{C}\sum_{i=1}^{M} b(c_j \mapsto l_i)\int_{V_i} P(c_j \mid x)p(x)dV_x$$

$$m_i^{(k+1)} = m_i^{(k)} - \gamma^{(k)}\nabla_i R^{(k)}(\Phi^{(k)})$$

Class of **z**

Sample of the training set

$$\nabla_i \boldsymbol{R}(\Omega) = \sum_{q=q_1}^{q_N} \frac{b(\boldsymbol{h} \mapsto l_q) - b(\boldsymbol{h} \mapsto l_i)}{\left\| m_i - m_q \right\|} \times \int_{S_{i,q}} (m_i - x)w(x - \boldsymbol{z})dS_x$$

Indexes of code vectors adjacent to m_i

Voronoi surface separating V_i and V_q

Parzen window

$$m_i^{(k+1)} = m_i^{(k)} - \gamma^{(k)} \nabla_i R^{(k)}(\Phi^{(k)})$$

$$\nabla_i \boldsymbol{R}(\Omega) = \sum_{q=q_1}^{q_N} \frac{b(\boldsymbol{h} \mapsto l_q) - b(\boldsymbol{h} \mapsto l_i)}{\left\| m_i - m_q \right\|} \times \int_{S_{i,q}} (m_i - x) w(x - z) dS_x$$

BVQ uses for *w(x)* an hyper cubic window

$$w(x) \begin{cases} \Delta^{-n} & \text{over an hyper cube centered in z} \\ 0 & \text{elsewhere} \end{cases}$$

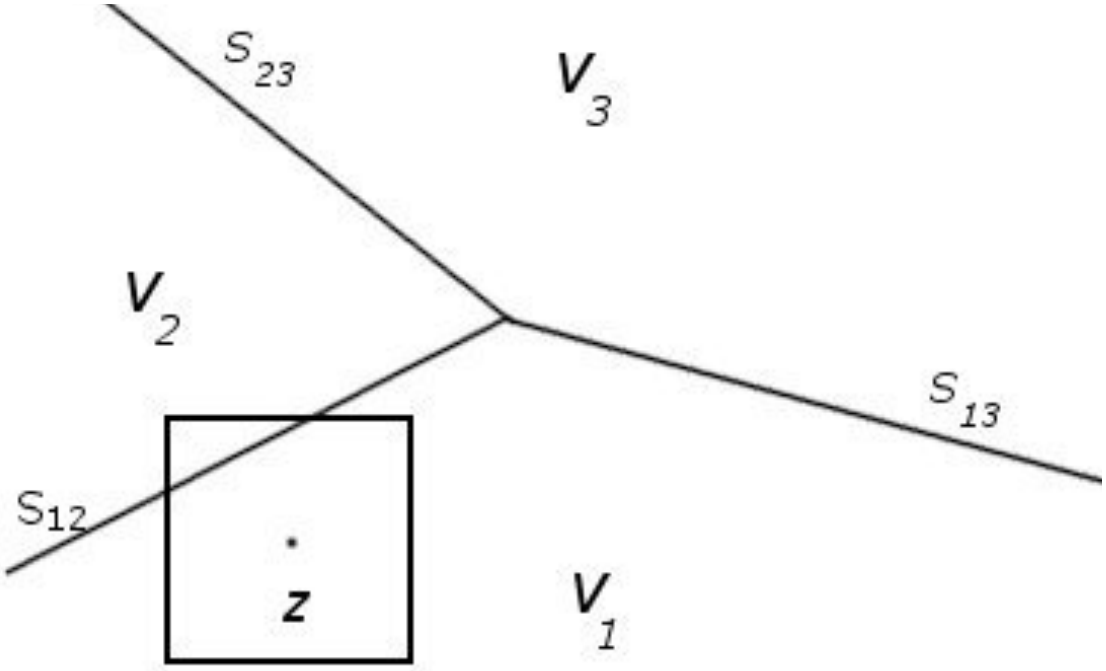
Problem: how to find S_{iq} that fall in the window?

Contents

- Bvq: minimum risk classification
- **Bvq2: an approximation**
  - **Problem of Voronoi cusp**
- Bvq3: an improvement(better approximation)
- Experimental evaluation
- Conclusion and future work

BVQ2

Assumption: w(x) intersects only the piece of the decision border nearest to z

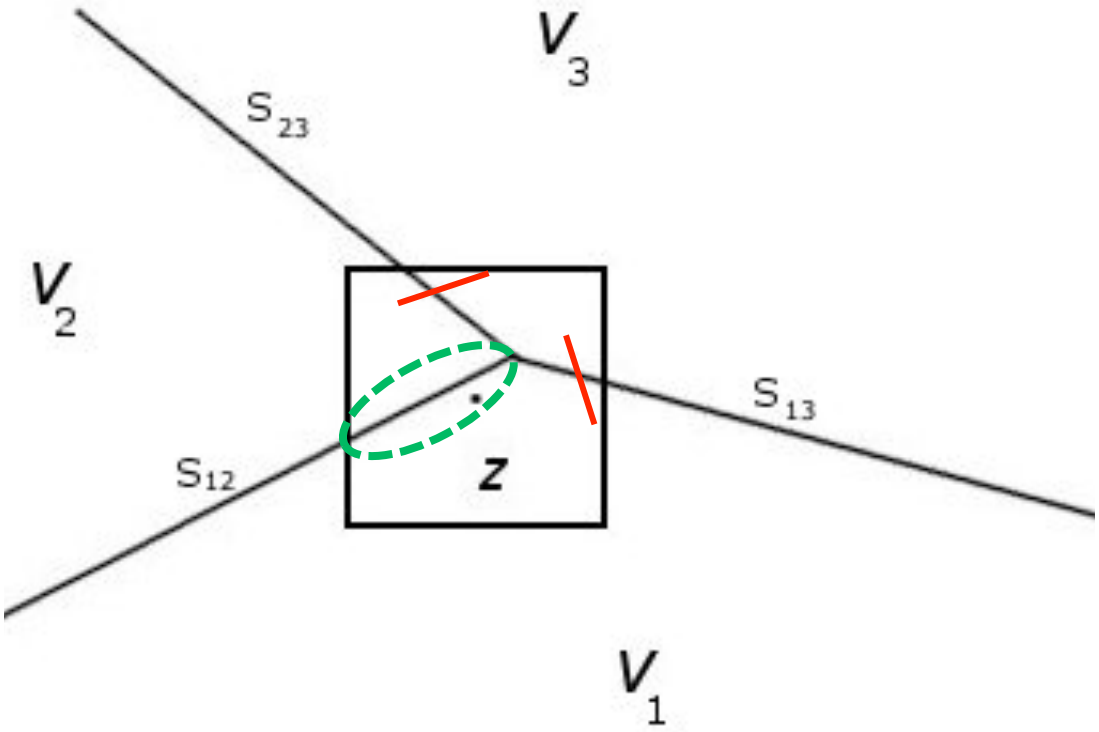


Approximated solution

The algorithm updates up to two code vectors at each iteration

- Easy implementation
- Good experimental results
- Reduction in the computational load
- Sometimes the approximation error is not negligible

Problem of Voronoi vertex



Contents

- Bvq: minimum risk classification
- Bvq2
  - an approximation(two code vectors)
  - Problem of Voronoi cusp
- **Bvq3**
  - **better approximation(three code vectors)**
- Experimental evaluation
- Conclusion and future work

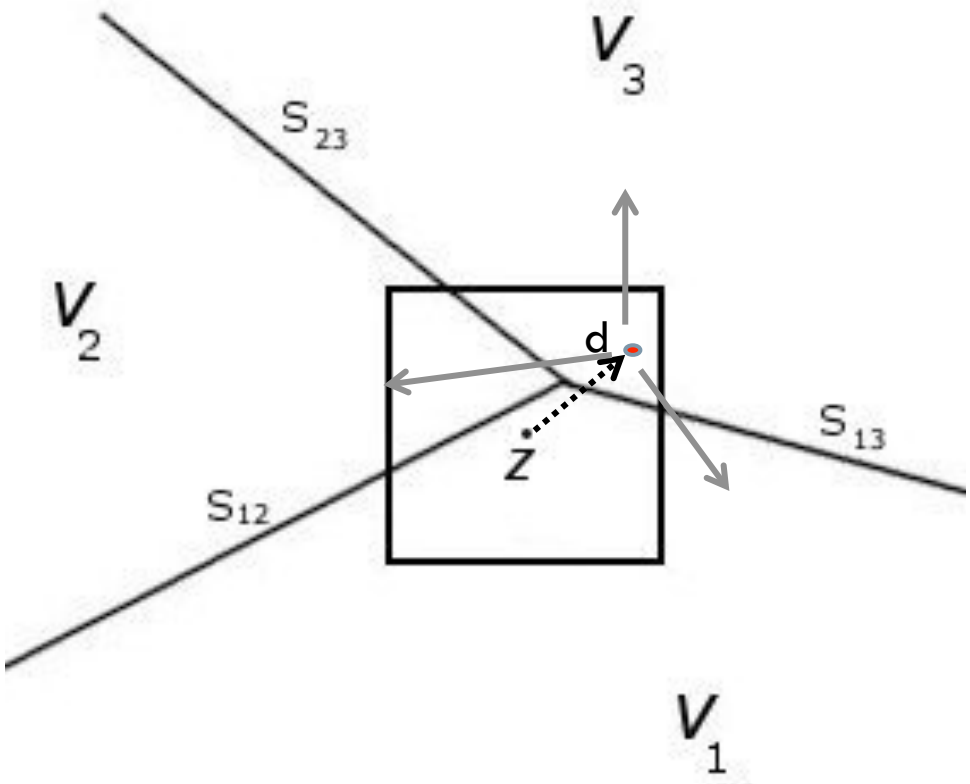
BVQ3

- *Update of the third neighbor*
  - *Better approximation of the gradient*
  - You must verify:
    - *the distance between z and S_{12}*
    - *the adjacency of the regions*

Problem: how to check the adjacency of Voronoi regions?

Check of the adjacency

- Adding a Gaussian noise to z to find d
- Checking its neighbors
- Several attempts
  - Determined by *try_max* parameter



Contents

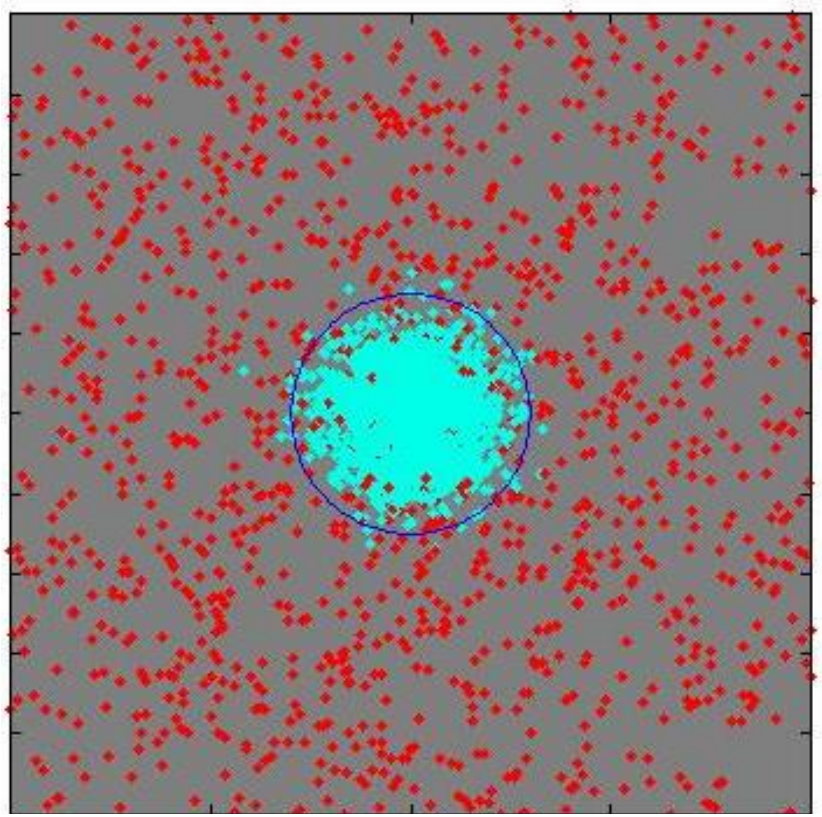
- Bvq: minimum risk classification
- Bvq2
  - an approximation(two code vectors)
  - Problem of Voronoi cusp
- Bvq3
  - better approximation(three code vectors)
- **Experimental evaluation**
- Conclusion and future work

Experiments

- One synthetic dataset – Gaussian
- Four real-world datasets from UCI
- Evaluation based on
  - Error probability
  - Execution time

Gaussian

- Each class represents an independent Gaussian distribution
  - Same probability and zero means
  - Different covariance matrixes:
    - I for class C1
    - 0.01 * I for class C2



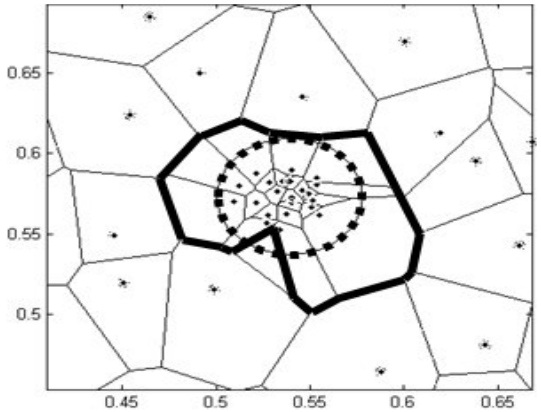
Parameters setting

		Parameter	Value
BVQ3	BVQ2	Window size (Δ)	0.0159
		n. Iterations	40000
		Step size (Y^0)	[0.00159-0.0159]
		n. of code vectors	[4-128]
		Try_max	[10,20,50]

- 10-fold cross validation

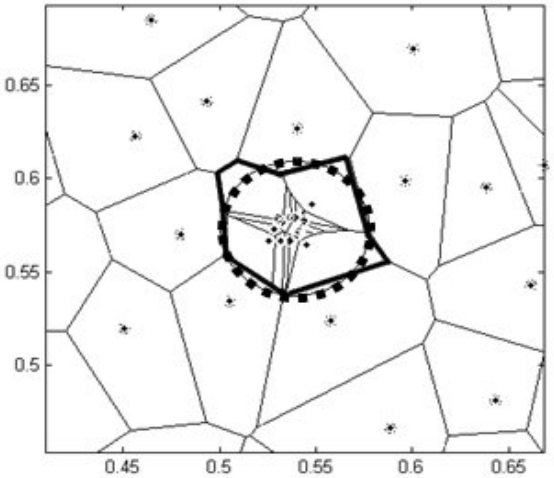
Gaussian: results

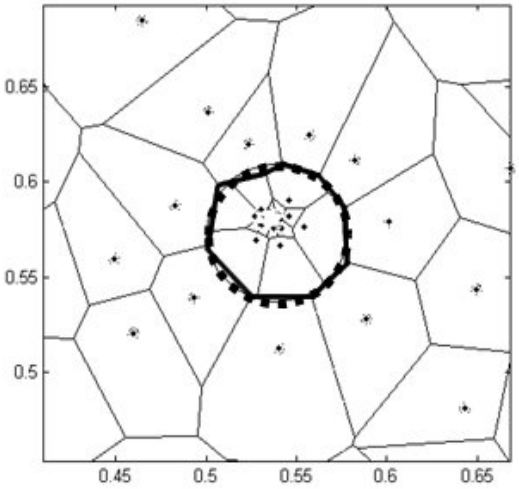
Initial code vectors configurations



Bayes Error: 0.027

BVQ2
Error probability:
0.0282





BVQ3
Error probability:
0.0277

Real-world datasets

- Datasets from UCI: Australian, Liver, Ionosphere and Mushroom
- Preprocessing: normalization

Dataset	dim	N	N_1	N_2
Australian	14	690	307	383
Liver	6	345	145	200
Mushroom	22	8124	4208	3916
Ionosphere	34	351	126	225

Parameters settings

		Parameter	Value
BVQ3	BVQ2	Window size (Δ)	Different for each dataset
		n. Iterations	50000
		Step size (Y^0)	[0.1-1]
		n. of code vectors	[4-128]
		Try_max	[10,20,30]

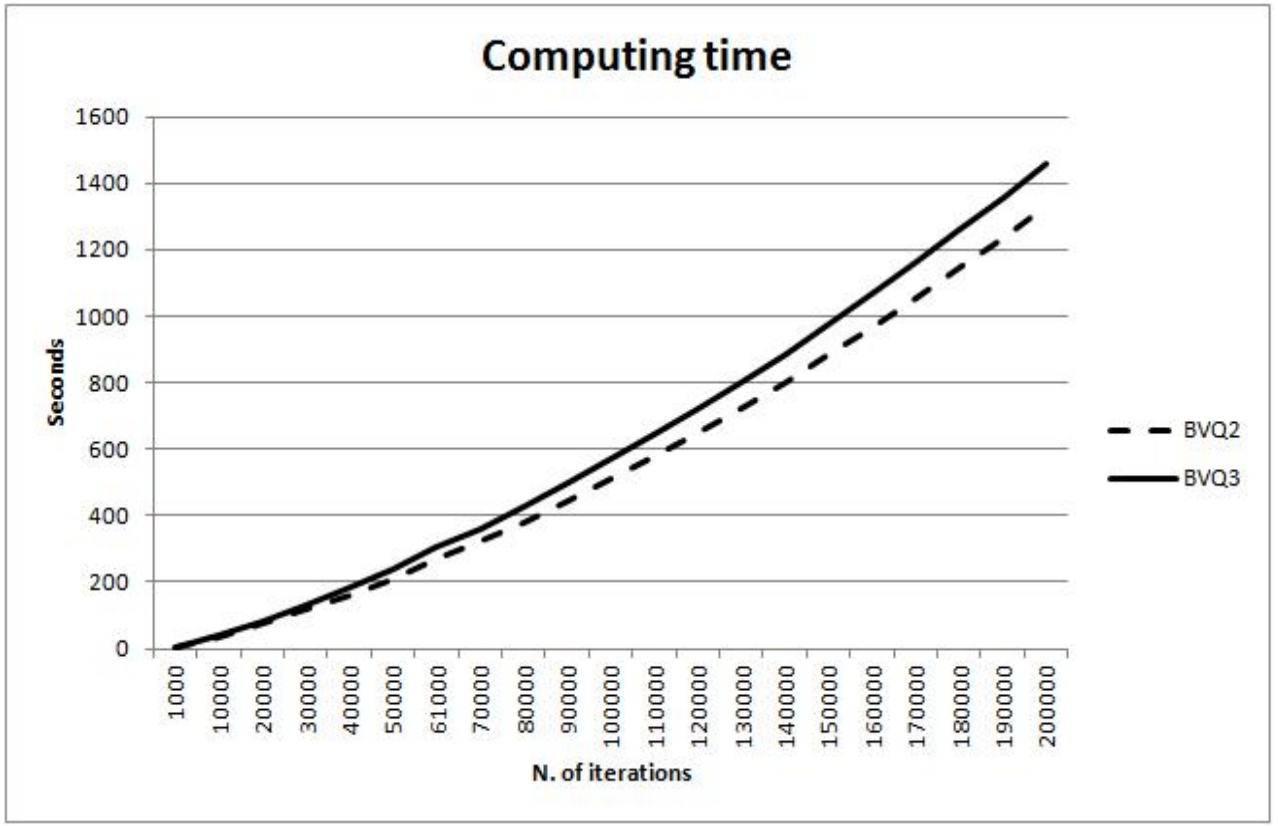
- 10-fold cross validation

Real-world datasets: results

		Number of code vectors					
Dataset		4	8	16	32	64	128
Australian	BVQ2	0.1406	0.1420	0.1478	0.1420	**0.1391**	0.1522
	BVQ3	0.1478	**0.1319**	0.1449	0.1478	0.1406	0.1391
Liver	BVQ2	0.3205	0.3200	**0.3190**	0.3505	0.3505	0.3700
	BVQ3	0.3267	**0.2919**	0.2981	0.3305	0.3324	0.3348
Mushroom	BVQ2	0.1001	0.0617	0.0327	0.0211	0.0113	**0.0065**
	BVQ3	0.0996	0.0373	0.0201	0.0049	**0.0017**	0.0046
Ionosphere	BVQ2	0.1480	0.1167	0.1082	0.1254	**0.0998**	0.1111
	BVQ3	0.1426	0.1194	0.0944	0.0898	0.0944	**0.0639**

Execution time

- Computing time: BVQ3 vs BVQ2



Contents

- Bvq: minimum risk classification
- Bvq2
  - an approximation(two code vectors)
  - Problem of Voronoi cusp
- Bvq3
  - better approximation(three code vectors)
- Experimental evaluation
- **Conclusion and future work**

BVQ2 vs BVQ3

- Main differences:
  - Extraction of the third neighbor
  - Check of the adjacency of the third regions
- The complexity order is the same for the two versions
- BVQ3 has experimentally obtained the best results

Future works

- Future works:
  - Extend the experiments to real data, in particular on datasets:
    - Multiclass
    - Imbalanced
  - Analysis of parameters relation

Thank you for attention!

BVQ Algorithm

Let label(**x**) be the function that returns the class of **x**.

1. Set the value of Δ, γ^0, the number of iterations n_{max} and the maximum number of attempts *try_max*;
2. Initialize code vectors m_i.....m_n;
3. Set $flag_{BVQ3} = 0$;
4. For k = 1 to n_{max} do
   1. Randomly pick a training pair $(z^{(k)}, h^{(k)})$ from the training set;
   2. Find the first three code vectors m_1, m_2 ,m_3 nearest to $z^{(k)}$, such that
      $$\|m_1 - z^{(k)}\|^2 \leq \|m_2 - z^{(k)}\|^2 \leq \|m_3 - z^{(k)}\|^2$$
   3. If (label(m_1) $\neq$ label(m_2) and $\left\|z^{(k)} - z^{(k)}_{1,2}\right\| \leq \Delta/2$)
      1. If (label(m_1) $\neq$ label(m_3) or label(m_2) $\neq$ label(m_3))
         1. Set *try=0;*
         2. While ($flag_{BVQ3}=0$ and *try* < *try_max*)
            1. Set $d=z^{(k)}$ + *noise;*
            2. *try* = *try* + *1;*
            3. If $w(d - z^{(k)}) \neq 0$
               1. Find the first 3 code vectors q_1,q_2,q_3 nearest to d, such that: $\|q_1 - d\|^2 < \|q_2 - d\|^2 < \|q_3 - d\|^2$
               2. If ($q_1 = m_3$ and ($q_2 = m_1$ or $q_2 = m_2$))
                  1. update m_1, m_2 ,m_3 using the formula in (10);
                  2. $flag_{BVQ3} = 1$;
      2. else update m_1 and m_2 using the formula in (9);