LASVM applied to invariant problems

Gaëlle Loosli*, Léon Bottou**,
Stéphane Canu*
gaelle.loosli@insa-rouen.fr

** NEC Laboratories America
* Lab. Perception, Systèmes, Information - CNRS - FRE 2645

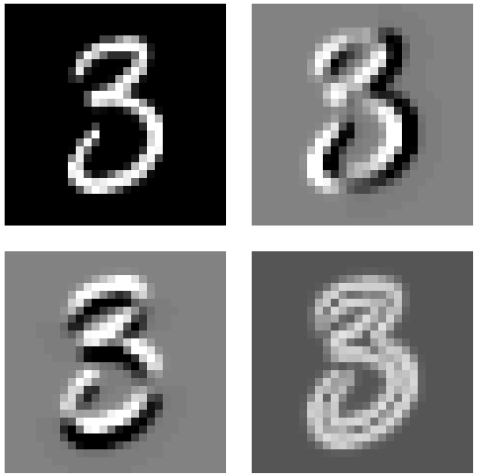
NIPS Workshop on Large Scale Kernel Machines

Roadmap

Definitions for deformations

If x represents a 3, then $T(x, d)$ should also represent a 3:

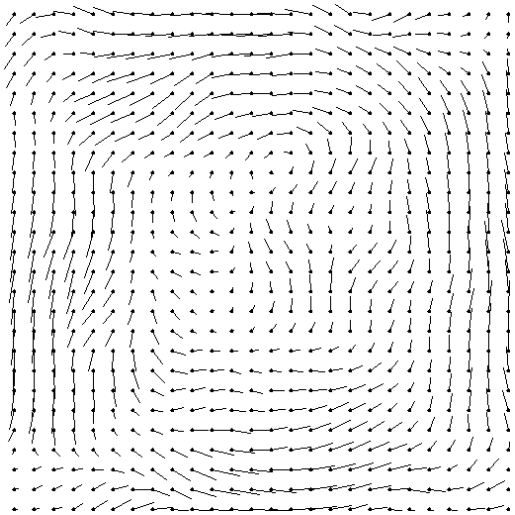
- Tangent vectors
- Deformation fields
  - Regular
  - Noisy (smoothed)

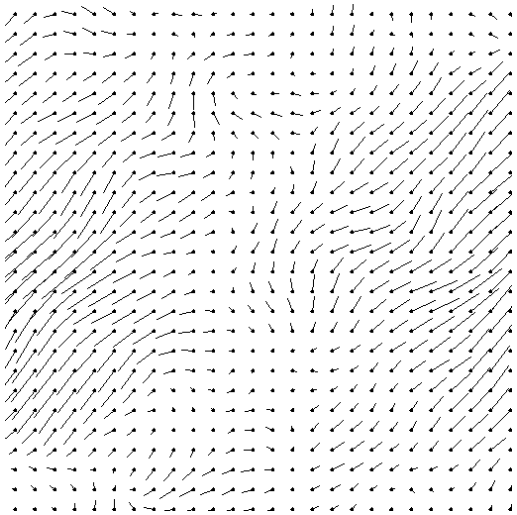


Definitions for deformations

If x represents a 3, then $T(x, d)$ should also represent a 3:

- Tangent vectors
- Deformation fields
  - Regular
  - Noisy (smoothed)



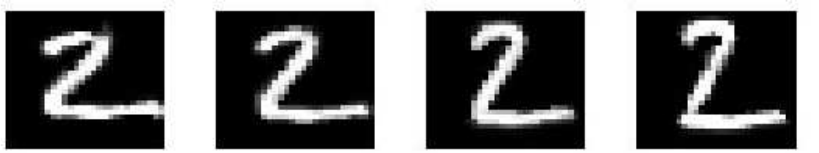


What to use for invariances?

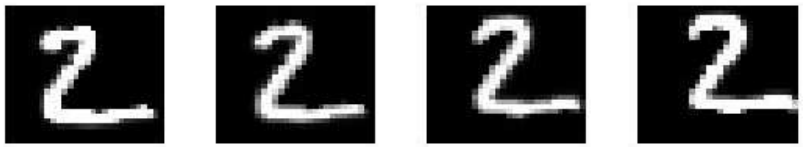
- Affine deformations (linear approximations)
- Thickening
- Elastic deformations

$$x_{affine}(i,j) = x(i,j) + \alpha_x * d_x * t_x(i,j) + \alpha_y * d_y * t_y(i,j)$$

Rotations with tangent vectors



Translations with tangent vectors



What to use for invariances?

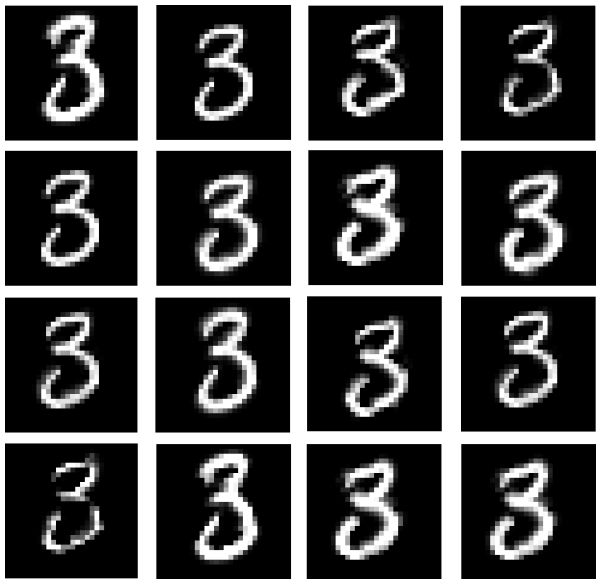
- Affine deformations (linear approximations)
- Thickening
- Elastic deformations

$$x_{thick}(i,j) = x(i,j) + \beta * \sqrt{t_x(i,j)^2 + t_y(i,j)^2}$$

What to use for invariances?

- Affine deformations (linear approximations)
- Thickening
- Elastic deformations

$$x_{deformed}(i,j) = x(i,j) + f_x(i,j) * t_x(i,j) + f_y(i,j) * t_y(i,j)$$



P. Y. Simard, Y. LeCun, J. S. Denker, and B. Victorri.

Transformation invariance in pattern recognition – tangent distance and tangent propagation.

International Journal of Imaging Systems and Technology, 11(3), 2000.

Patrice Y. Simard, Dave Steinkraus, and John C. Platt.

Best practices for convolutional neural networks applied to visual document analysis.

In *ICDAR '03: Proceedings of the Seventh International Conference on Document Analysis and Recognition*, page 958, Washington, DC, USA, 2003. IEEE Computer Society.

What to use for invariances?

- Affine deformations (linear approximations) [SLDV00]
- Thickening
- Elastic deformations [SSP03]

Random deformation

$$x_{transformed}(i,j) = x(i,j) + \alpha * (f_x(i,j) * \mathbf{t_x}(i,j) + f_y(i,j) * \mathbf{t_y}(i,j)) + \beta * \sqrt{\mathbf{t_x}(i,j)^2 + \mathbf{t_y}(i,j)^2}$$

What do we need?

- Tangent vectors - linear deformations (t_x, t_y)
- Deformation fields (regular or noisy) (f_x, f_y)
- Deformation strength parameters (α, β)
- We add 1 and 2 pixels translations any 8 directions

What to use for invariances?

- Affine deformations (linear approximations)
- Thickening
- Elastic deformations

Random deformation

$$x_{transformed}(i,j) = x(i,j) + \alpha * (\mathbf{f_x}(i,j) * t_x(i,j) + \mathbf{f_y}(i,j) * t_y(i,j)) + \beta * \sqrt{t_x(i,j)^2 + t_y(i,j)^2}$$

What do we need?

- Tangent vectors - linear deformations (t_x, t_y)
- Deformation fields (regular or noisy) (f_x, f_y)
- Deformation strength parameters (α, β)
- We add 1 and 2 pixels translations any 8 directions

What to use for invariances?

- Affine deformations (linear approximations)
- Thickening
- Elastic deformations

Random deformation

$$x_{transformed}(i,j) = x(i,j) + \alpha * (f_x(i,j) * t_x(i,j) + f_y(i,j) * t_y(i,j)) + \beta * \sqrt{t_x(i,j)^2 + t_y(i,j)^2}$$

What do we need?

- Tangent vectors - linear deformations (t_x, t_y)
- Deformation fields (regular or noisy) (f_x, f_y)
- Deformation strength parameters (α, β)
- We add 1 and 2 pixels translations any 8 directions

What to use for invariances?

- Affine deformations (linear approximations)
- Thickening
- Elastic deformations

Random deformation

$$x_{transformed}(i,j) = x(i,j) + \alpha * (f_x(i,j) * t_x(i,j) + f_y(i,j) * t_y(i,j)) + \beta * \sqrt{t_x(i,j)^2 + t_y(i,j)^2}$$

What do we need?

- Tangent vectors - linear deformations (t_x, t_y)
- Deformation fields (regular or noisy) (f_x, f_y)
- Deformation strength parameters (α, β)
- We add 1 and 2 pixels translations any 8 directions

How to incorporate invariances in learning?

- Invariances with SVM methods
  - Modify the cost function
  - Learn trajectories
  - Add some deformed points

Modify the cost function - Tangent kernels [HK02, CS02]

Changes the distance between elements using tangent distance

O. Chapelle and B. Schölkopf.

Incorporating invariances in nonlinear svms.
In Dietterich T. G.and Becker S. and Ghahramani Z., editors, *Advances in Neural Information Processing Systems*, volume 14, pages 609–616, Cambridge, MA, USA, 2002. MIT Press.

B. Haasdonk and D. Keysers.

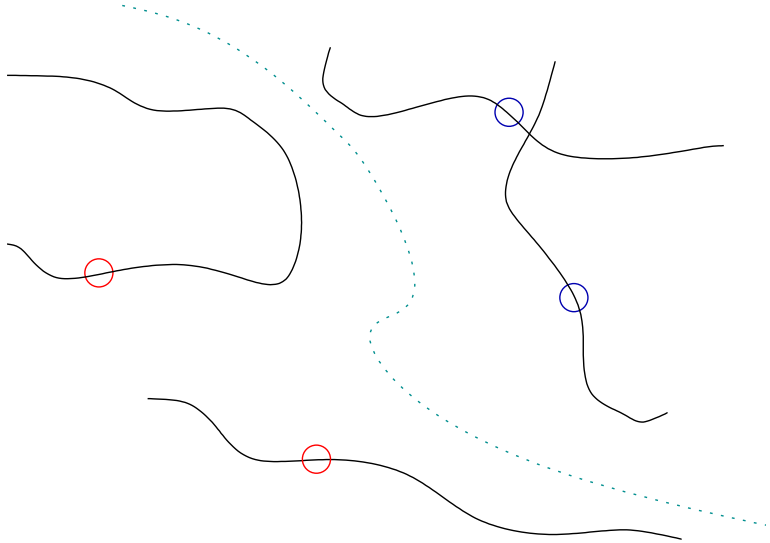
Tangent distance kernels for support vector machines.
In *International Conference on Pattern Recognition*, Quebec City, Canada, August 2002.

How to incorporate invariances in learning?

- Invariances with SVM methods
  - Modify the cost function
  - Learn trajectories
  - Add some deformed points

Learn trajectories - SDPM [GH04]

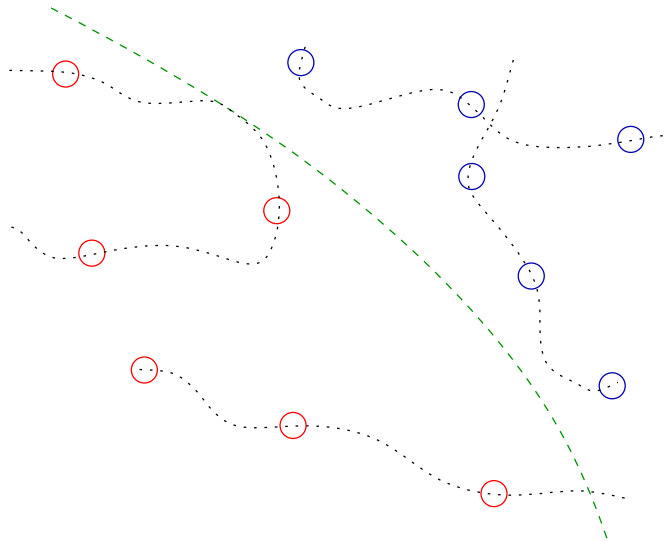
The idea is to classify the trajectories defined by a transformation with continuous parameter.



Thore Graepel and Ralf Herbrich.
Invariant pattern recognition by semi-definite programming machines.
In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

How to incorporate invariances in learning?

- Invariances with SVM methods
  - Modify the cost function
  - Learn trajectories
  - Add some deformed points



Database modification - Virtual SVM [DS02]

Adds deformed examples to the database

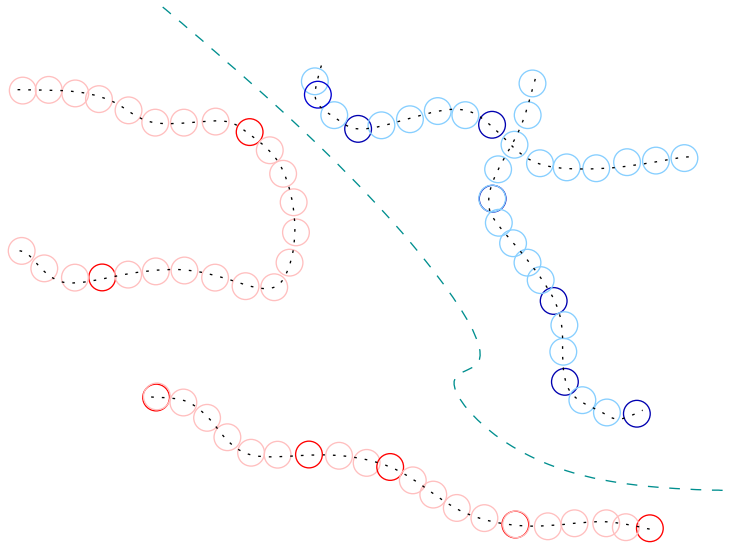
D. DeCoste and B. Schölkopf.
Training invariant support vector machines.
Machine Learning, 46:161–190, 2002.

How to incorporate invariances in learning?

- Invariances with SVM methods
  - Modify the cost function
  - Learn trajectories
  - Add some selected deformed points

Combination of trajectories and virtual points - [LCVJ05]

Uses discretized trajectories. The idea is to add to the database only virtual points that are SV



Gaëlle Loosli, Stéphane Canu, SVN Vishwanathan, and Alexander J.Smola.
Invariances in classification : an efficient svm implementation.
In *ASMDA 2005 -Applied Stochastic Models and Data Analysis*, 2005.

How to incorporate invariances in learning?

Good and bad

Method	Pro	Cons
Tangent distance	Efficient	Linear deformations only
Trajectories learning	All deformations	SDP - hard to solve
Virtual vectors	Simple, all deformations	Requires a lot of memory
Selected virtual vectors	Simple, all deformations	Not yet fast enough (simpleSVM limitations)

Objective

Apply the selected virtual vectors idea to a more adapted algorithm, namely LASVM

How to incorporate invariances in learning?

Good and bad

Method	Pro	Cons
Tangent distance	Efficient	Linear deformations only
Trajectories learning	All deformations	SDP - hard to solve
Virtual vectors	Simple, all deformations	Requires a lot of memory
Selected virtual vectors	Simple, all deformations	Not yet fast enough (simpleSVM limitations)

Objective

Apply the selected virtual vectors idea to a more adapted algorithm, namely LASVM

Helpful definitions and generalities

- Groups of points : I_A (active set) and I_0 (inactive set)
- Selection : pick a point from I_0 to transfer it to I_A
- Optimization : over I_A, may transfer a point from I_A to I_0
- Finalization : checks the final solution I_A

Helpful definitions and generalities

- Groups of points : I_A (active set) and I_0 (inactive set)
- Selection : pick a point from I_0 to transfer it to I_A
- Optimization : over I_A, may transfer a point from I_A to I_0
- Finalization : checks the final solution I_A

Helpful definitions and generalities

- Groups of points : I_A (active set) and I_0 (inactive set)
- Selection : pick a point from I_0 to transfer it to I_A
- Optimization : over I_A, may transfer a point from I_A to I_0
- Finalization : checks the final solution I_A

Helpful definitions and generalities

- Groups of points : I_A (active set) and I_0 (inactive set)
- Selection : pick a point from I_0 to transfer it to I_A
- Optimization : over I_A, may transfer a point from I_A to I_0
- Finalization : checks the final solution I_A

Helpful definitions and generalities

- Groups of points : I_A (active set) and I_0 (inactive set)
- Selection : pick a point from I_0 to transfer it to I_A
- Optimization : over I_A, may transfer a point from I_A to I_0
- Finalization : checks the final solution I_A

iterative SVM

- Initialize
- While selection is possible
  - Selection
  - Optimization
- Finalize

Algorithms - iterative methods

General loop

General	SimpleSVM [VSM03]	LASVM [BEWB05]
Selection	Takes the most violator point in I_0	Takes a point among the few next points (according to a criteria)
Optimization	Makes a full optimization over I_A	One SMO step between the candidate and a point from I_A (**Process**) and one SMO step between two points of I_A (**Reprocess**)
Finalization	Once no points are violators anymore, stops	Once all points are seen once, makes a full optimization over I_A (end of an *epoch*)

Antoine Bordes, Seyda Ertekin, Jason Weston, and Léon Bottou.
Fast kernel classifiers with online and active learning.
Journal of Machine Learning Research, 6:1579–1619, September 2005.

S. V. N Vishwanathan, A. J. Smola, and M. Narasimha Murty.
SimpleSVM.
In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

LASVM specificities

Things to play with

- Selection criteria
- Number of **Reprocess** (level of optimization at each step)
- Number of epochs

- Complete - brute force
- Gradient
- Active - distance to the margins
- Auto-active

LASVM specificities

Things to play with

- Selection criteria
- Number of **Reprocess** (level of optimization at each step)
- Number of epochs

- Complete - brute force
- Gradient
- Active - distance to the margins
- Auto-active

Complete

Every point is selected as a candidate to the active set

LASVM specificities

Things to play with

- Selection criteria
- Number of **Reprocess** (level of optimization at each step)
- Number of epochs

- Complete - brute force
- Gradient
- Active - distance to the margins
- Auto-active

Gradient

Looks at the z next points, selects the most misclassified point

LASVM specificities

Things to play with

- Selection criteria
- Number of **Reprocess** (level of optimization at each step)
- Number of epochs

- Complete - brute force
- Gradient
- Active - distance to the margins
- Auto-active

Active

Selects a point if it is between the margins ($\pm\delta/2$)

LASVM specificities

Things to play with

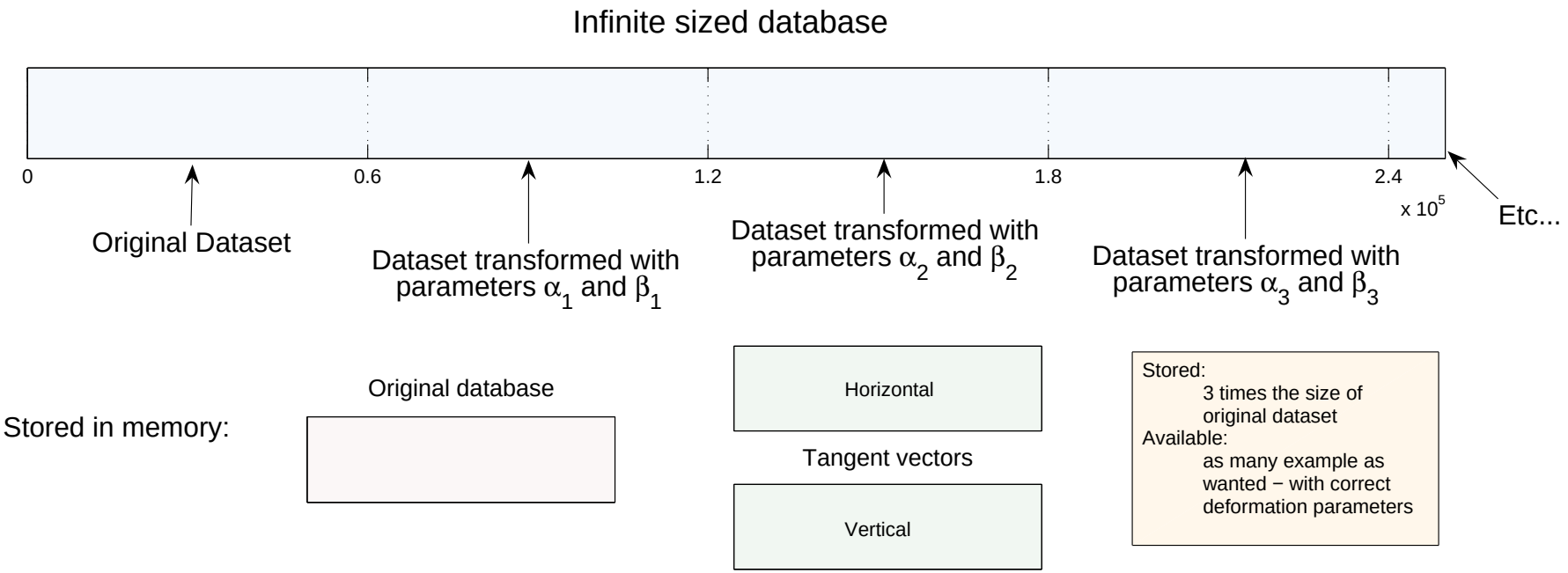
- Selection criteria
- Number of **Reprocess** (level of optimization at each step)
- Number of epochs

- Complete - brute force
- Gradient
- Active - distance to the margins
- Auto-active

Auto-active

Looks at the next points, keeps the points selected by the active rule, until 5 points are kept or 100 points checked. Selects the closest point to the decision boundary

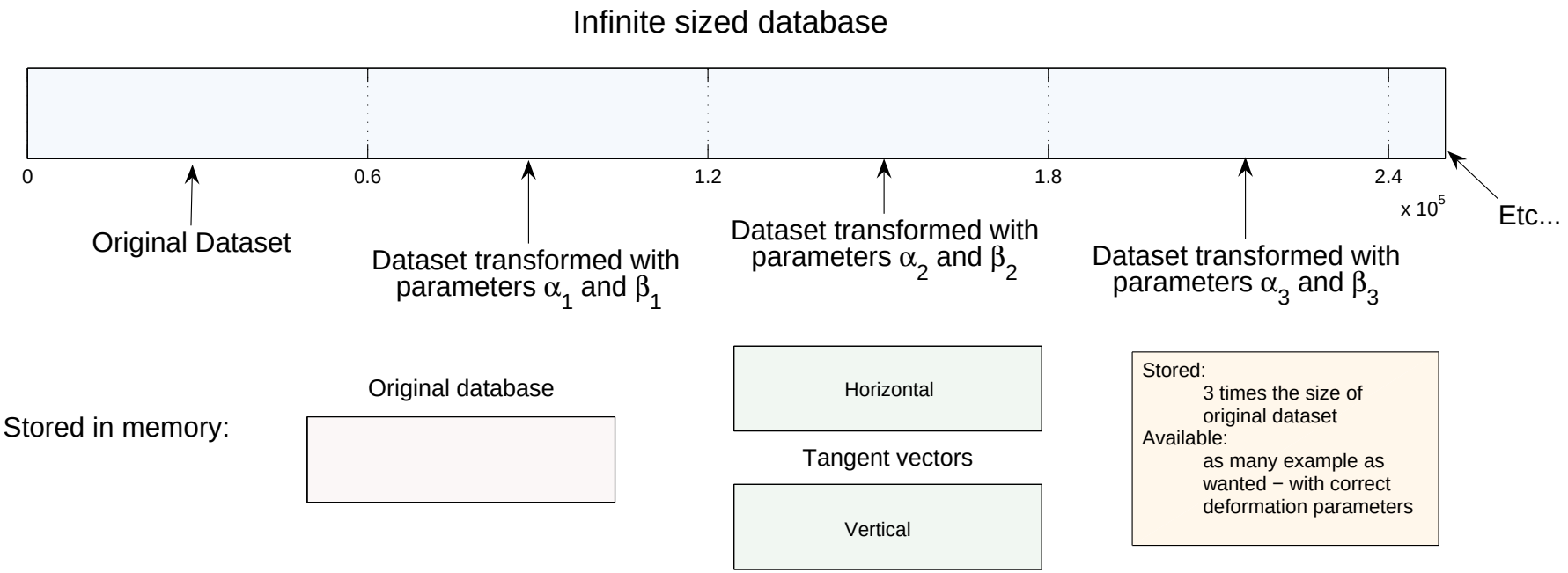
What is stored?



Memory usage

We only need to store 3 times the size of the original database. For convenience, we also store some pre-generated fields.

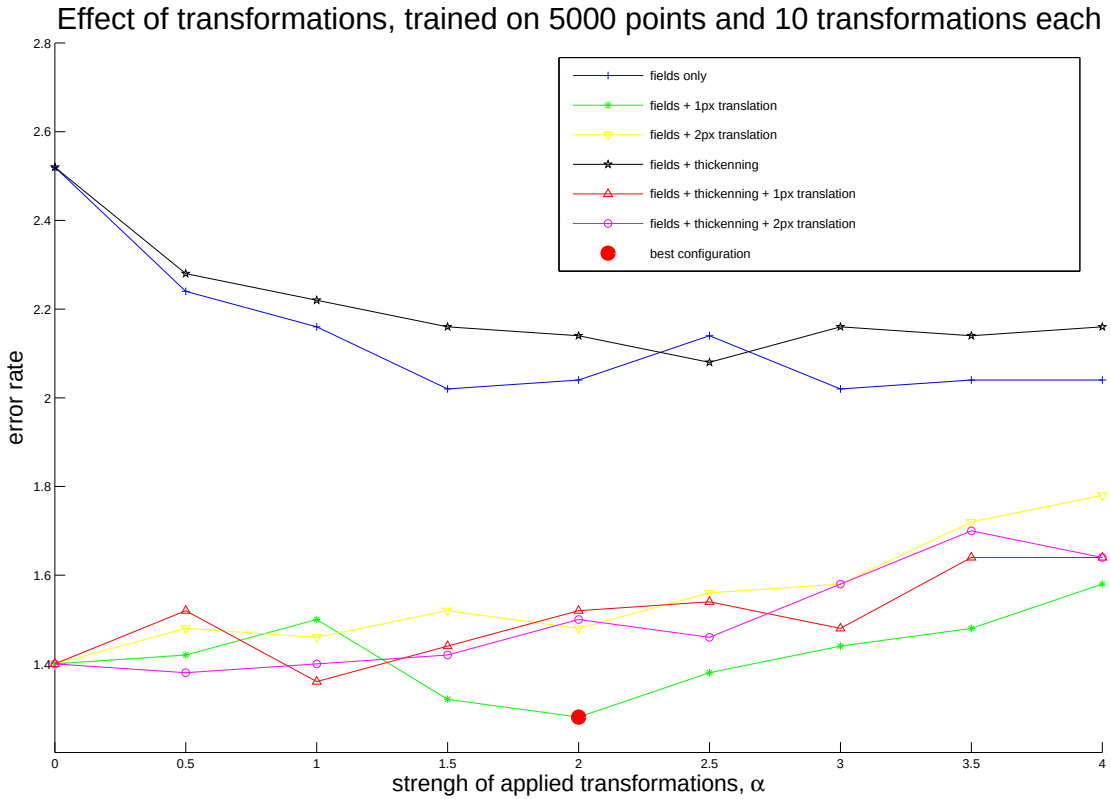
What is stored?



Memory usage

We only need to store 3 times the size of the original database. For convenience, we also store some pre-generated fields.

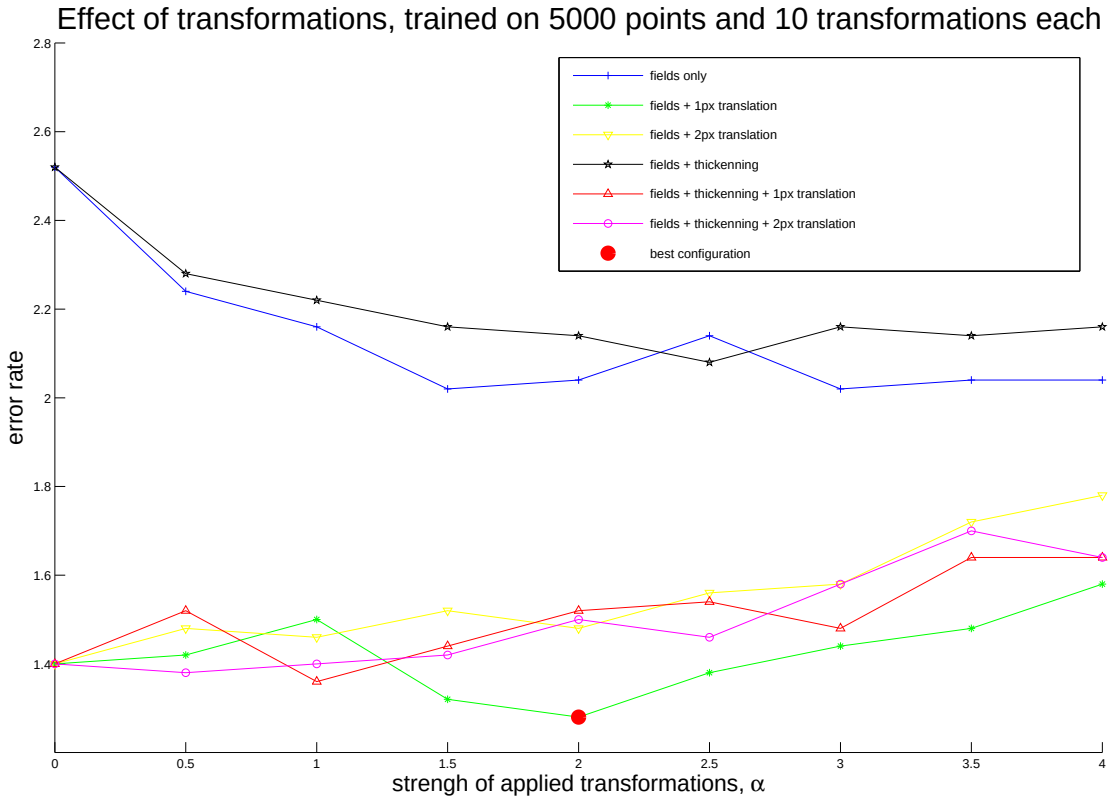
What are the relevant deformations for our task?



On the deformations

Translations are the most useful deformations for MNIST. Thickening does not help for MNIST database. $\beta = 0$

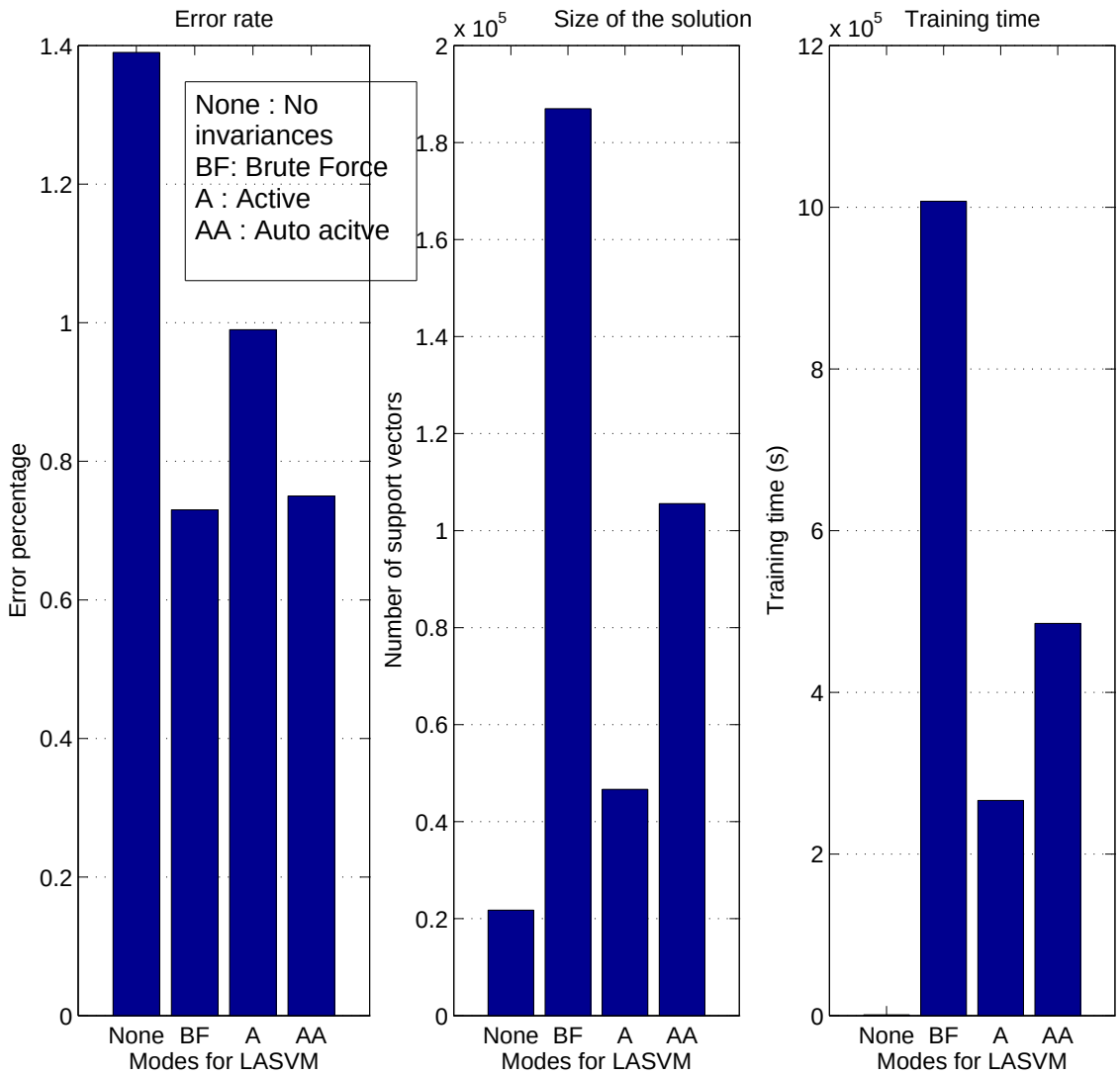
What are the relevant deformations for our task?



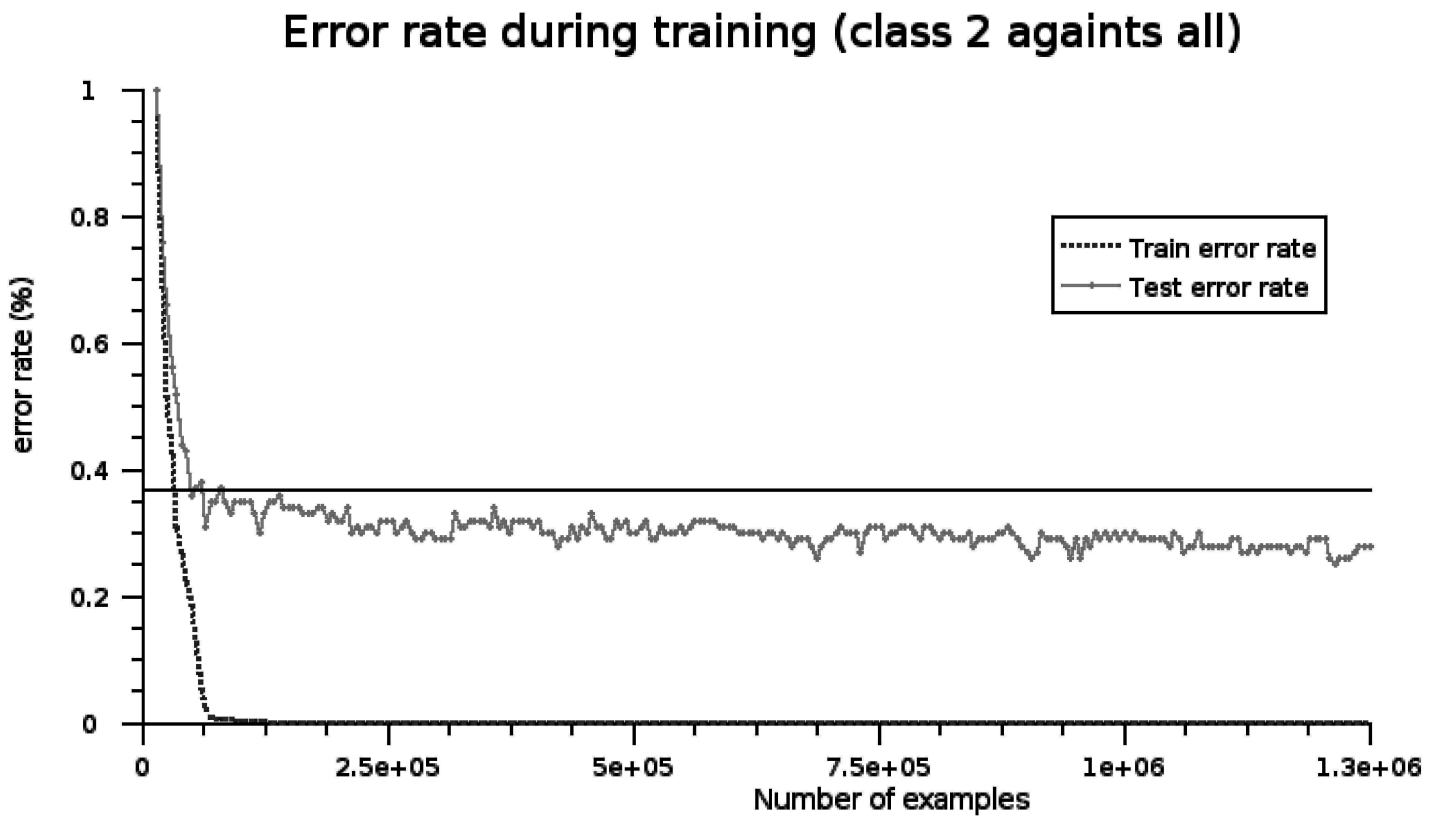
On the deformations

Translations are the most useful deformations for MNIST. Thickening does not help for MNIST database. $\beta = 0$

Which mode will we use?



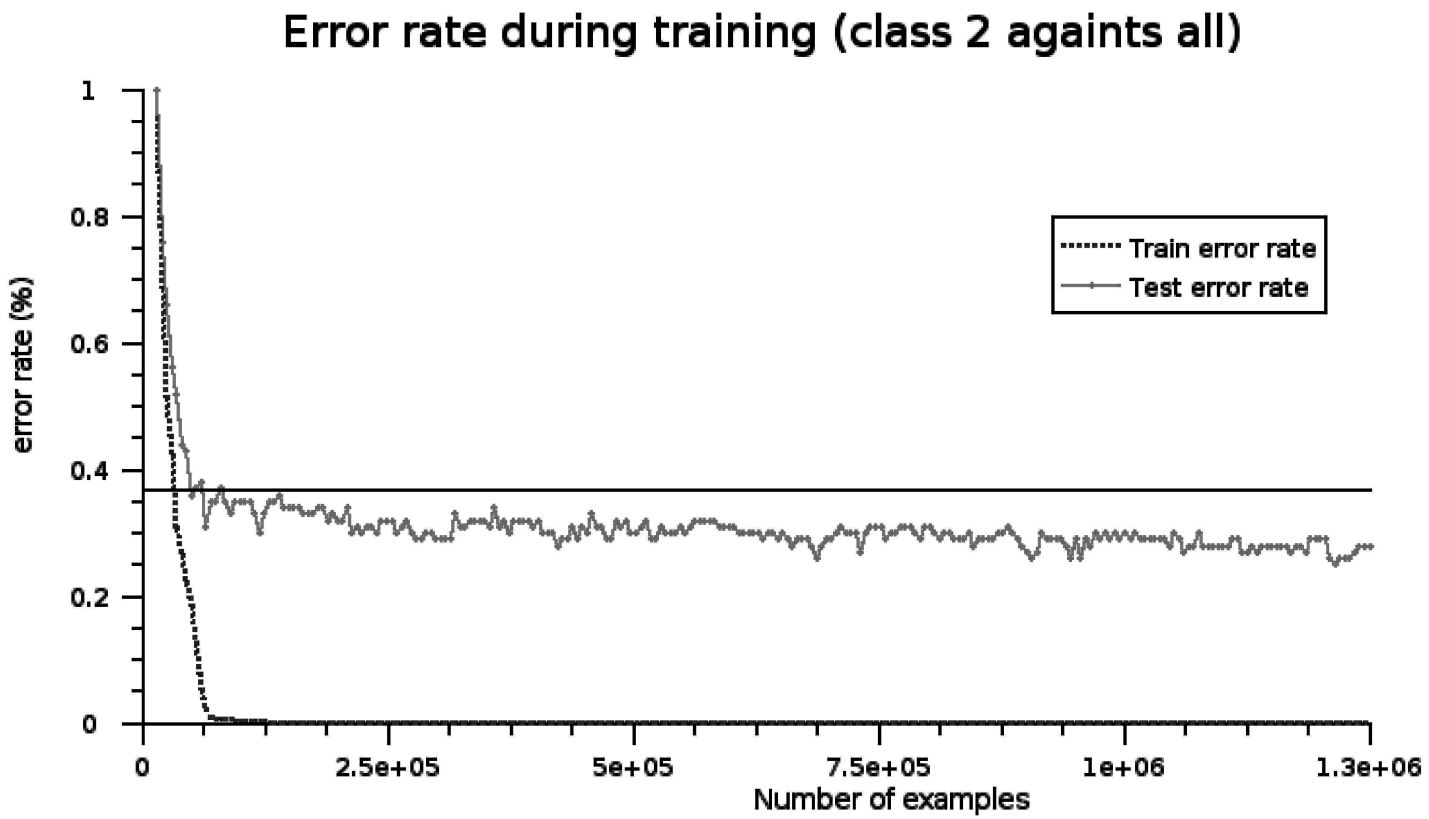
does it help to use those deformations?



On the efficiency

Adding some deformed examples slowly (but surely) improves the solution.

does it help to use those deformations?



On the efficiency

Adding some deformed examples slowly (but surely) improves the solution.

Optimal results

- Which transformations : all but thickening
- Training data size : 8 millions
- Solution sizes : about 120000 SV for the 10 classifiers
- Computational time : 8 days
- Performance : 0.67 %

Optimal results

- Which transformations : all but thickening
- Training data size : 8 millions
- Solution sizes : about 120000 SV for the 10 classifiers
- Computational time : 8 days
- Performance : 0.67 %

Optimal results

- Which transformations : all but thickening
- Training data size : 8 millions
- Solution sizes : about 120000 SV for the 10 classifiers
- Computational time : 8 days
- Performance : 0.67 %

Optimal results

- Which transformations : all but thickening
- Training data size : 8 millions
- Solution sizes : about 120000 SV for the 10 classifiers
- Computational time : 8 days
- Performance : 0.67 %

Optimal results

- Which transformations : all but thickening
- Training data size : 8 millions
- Solution sizes : about 120000 SV for the 10 classifiers
- Computational time : 8 days
- Performance : 0.67 %

NB: the machine used is a dual opteron with 16GB RAM (6.5GB cache used for kernel)...

Conclusion

- We wanted to solve invariance problem
- We needed to solve large SVM to do so

As a result...

- We ran some very large SVMs (the largest until now?)
- We obtained fairly good results (accuracy (we thought convolution networks are still much better))

Conclusion

- We wanted to solve invariance problem
- We needed to solve large SVM to do so

As a result...

- We ran some very large SVMs (the largest until now?)
- We obtained fairly good results (accuracy (we thought convolution networks are still much better)

Conclusion

- We wanted to solve invariance problem
- We needed to solve large SVM to do so

As a result...

- We ran some very large SVMs (the largest until now?)
- We obtained fairly good results in accuracy (even though convolution networks are still much better)

Conclusion

- We wanted to solve invariance problem
- We needed to solve large SVM to do so

As a result...

- We ran some very large SVMs (the largest until now?)
- We obtained fairly good results in accuracy (even though convolution networks are still much better)

Conclusion

- We wanted to solve invariance problem
- We needed to solve large SVM to do so

As a result...

- We ran some very large SVMs (the largest until now?)
- We obtained fairly good results in accuracy (even though convolution networks are still much better)

Perspective

Challenge

Yann's challenge on stereo pictures?