(Linear) Metric Learning

Matt Kusner

(slides based on tutorial given by Kilian Weinberger)

What is similarity?

Who is most similar?

Similar gender?

Similar age?

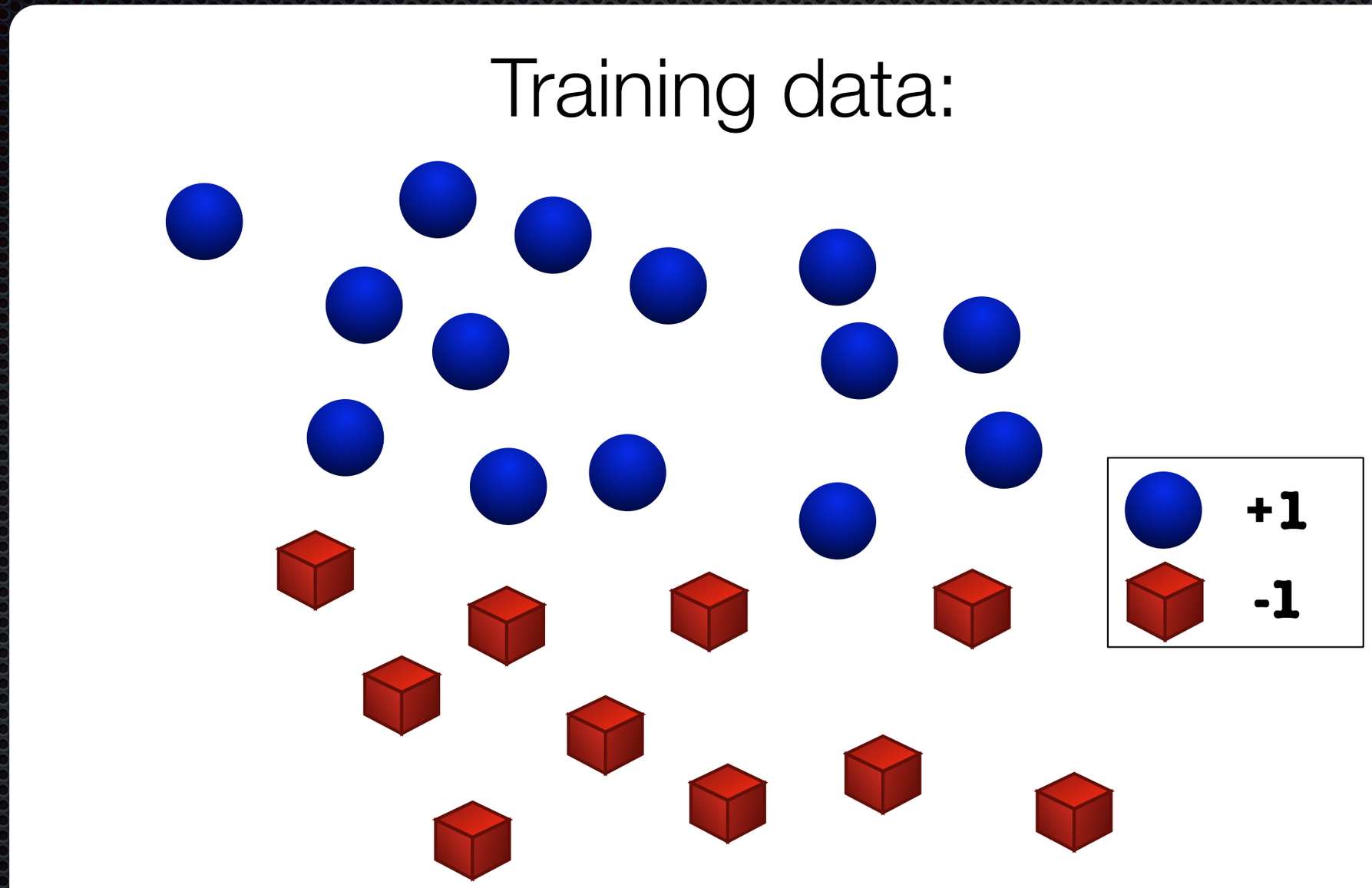
Similar hair cut?

Similarity
depends on
the context!

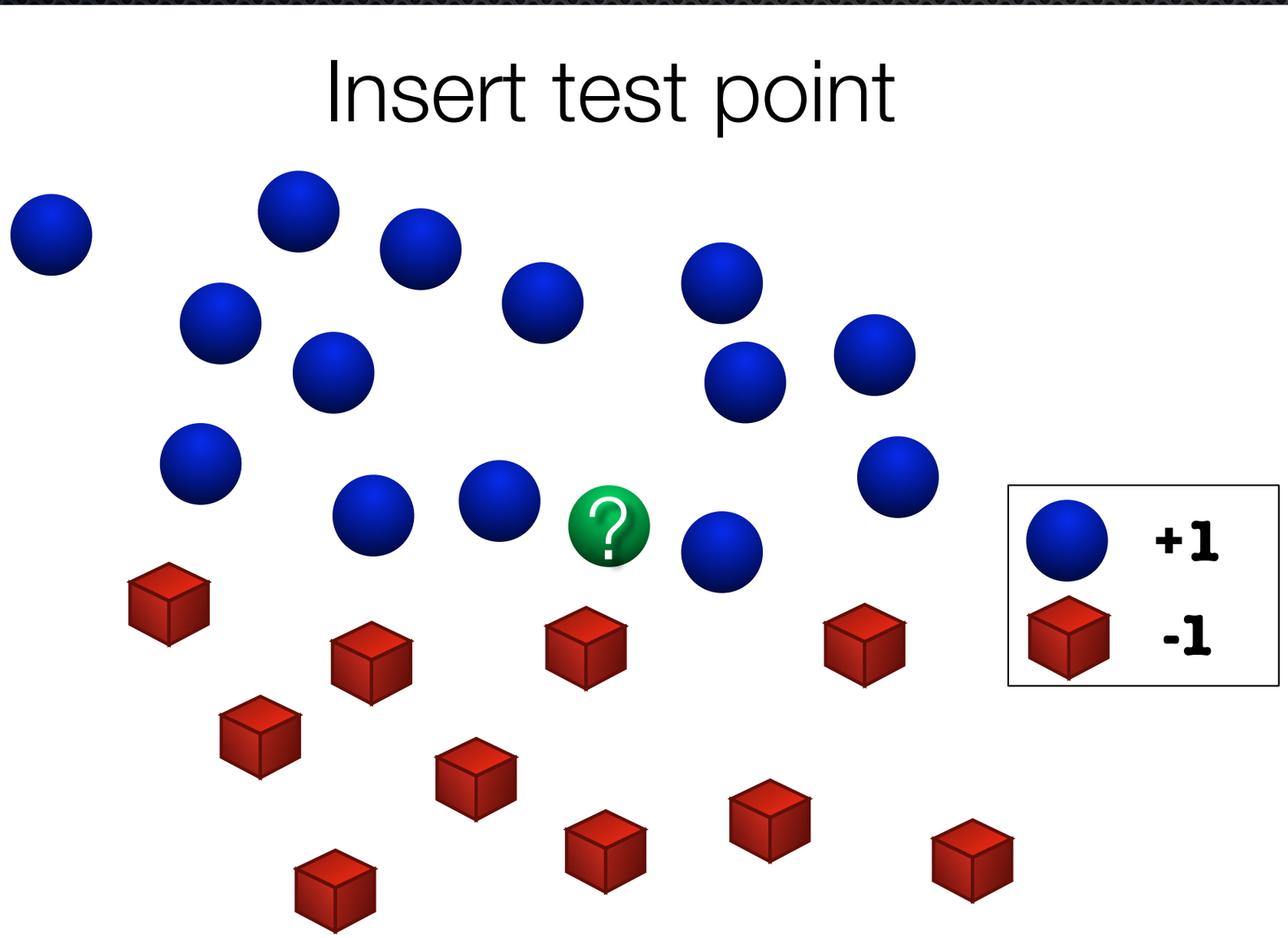
Motivation

- Many machine learning algorithms require a measure of (dis-) similarity (e.g. k-means, k-NN, SVMs, ...)
- Typically uninformed norms (e.g. **Euclidean distance**) are used as a measure of dissimilarity
- But we could incorporate our **knowledge** about the **task** and the **data**...
- Ideally we want a specific similarity measure for each task that is **semantically meaningful**.

k-nearest neighbor classification

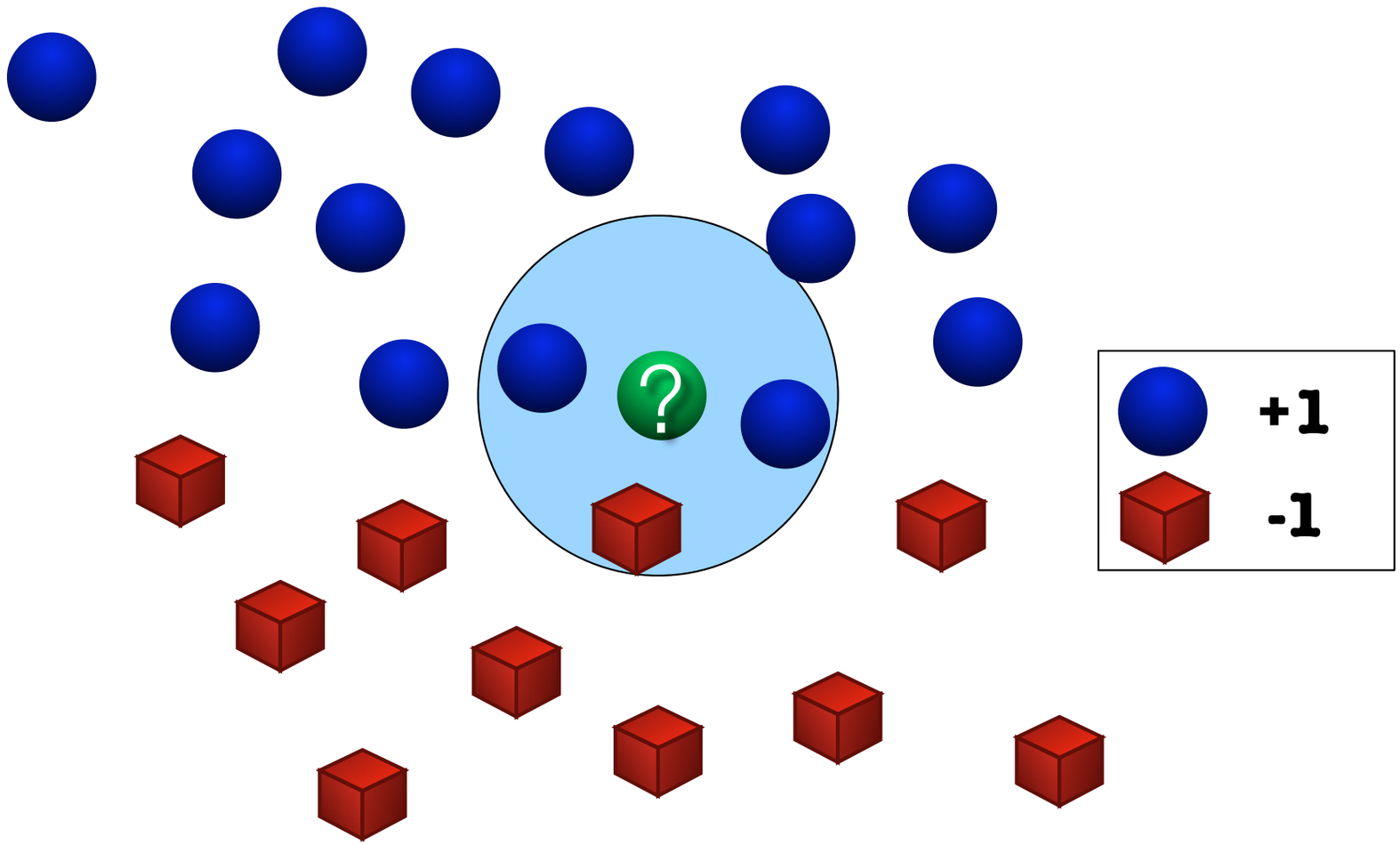


k-nearest neighbor classification



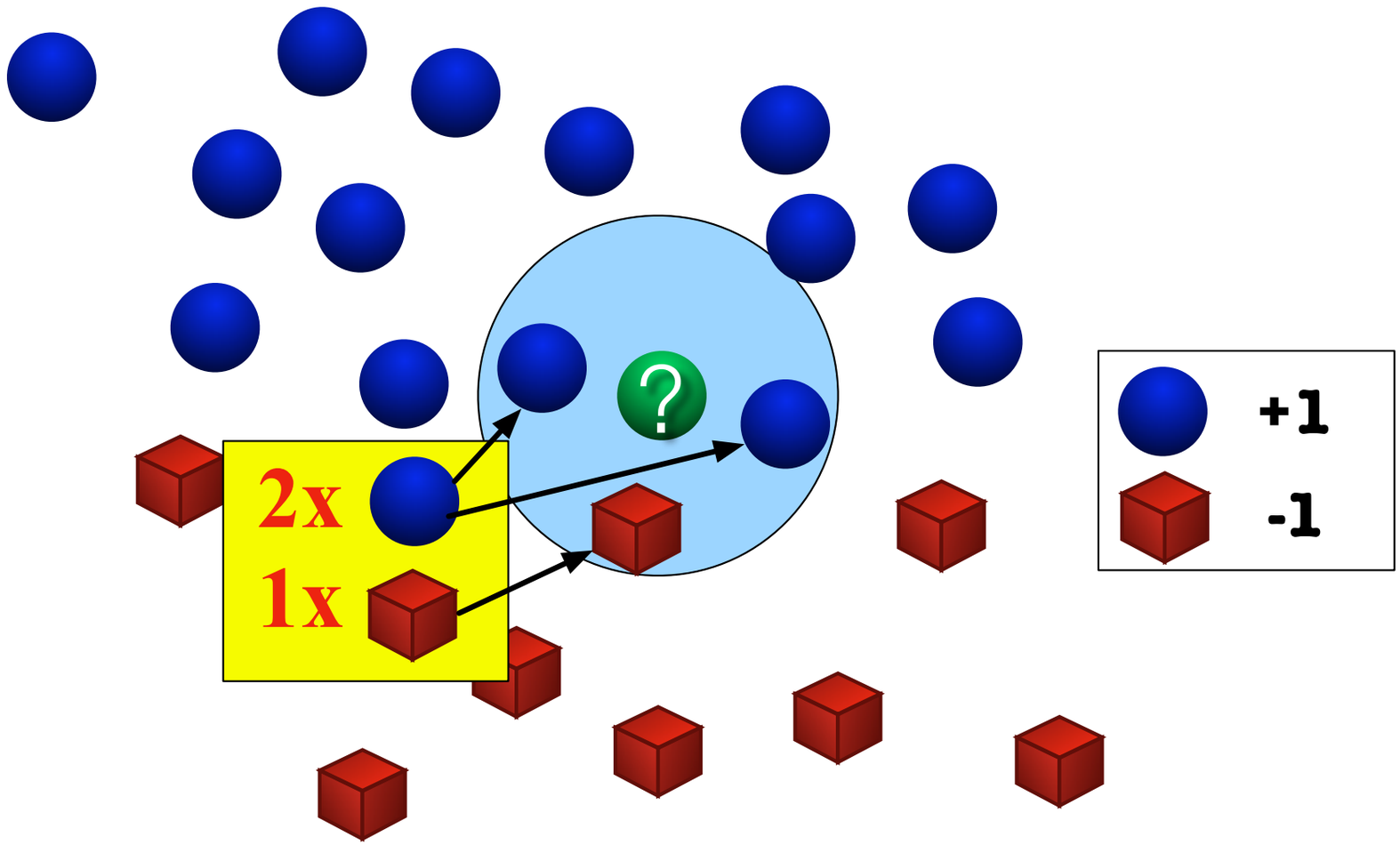
k-nearest neighbor classification

Find k nearest neighbors



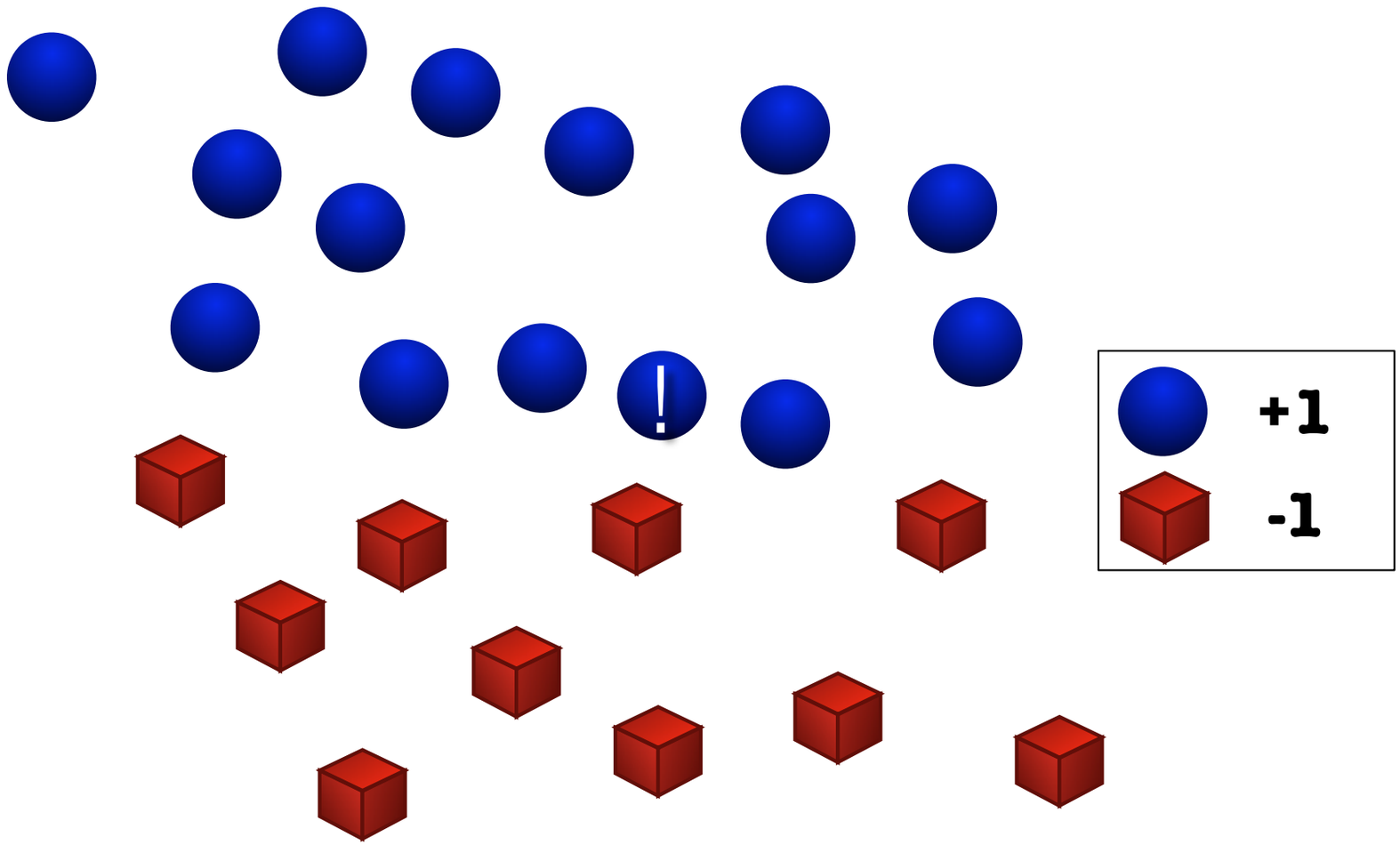
k-nearest neighbor classification

Let neighbors vote



k-nearest neighbor classification

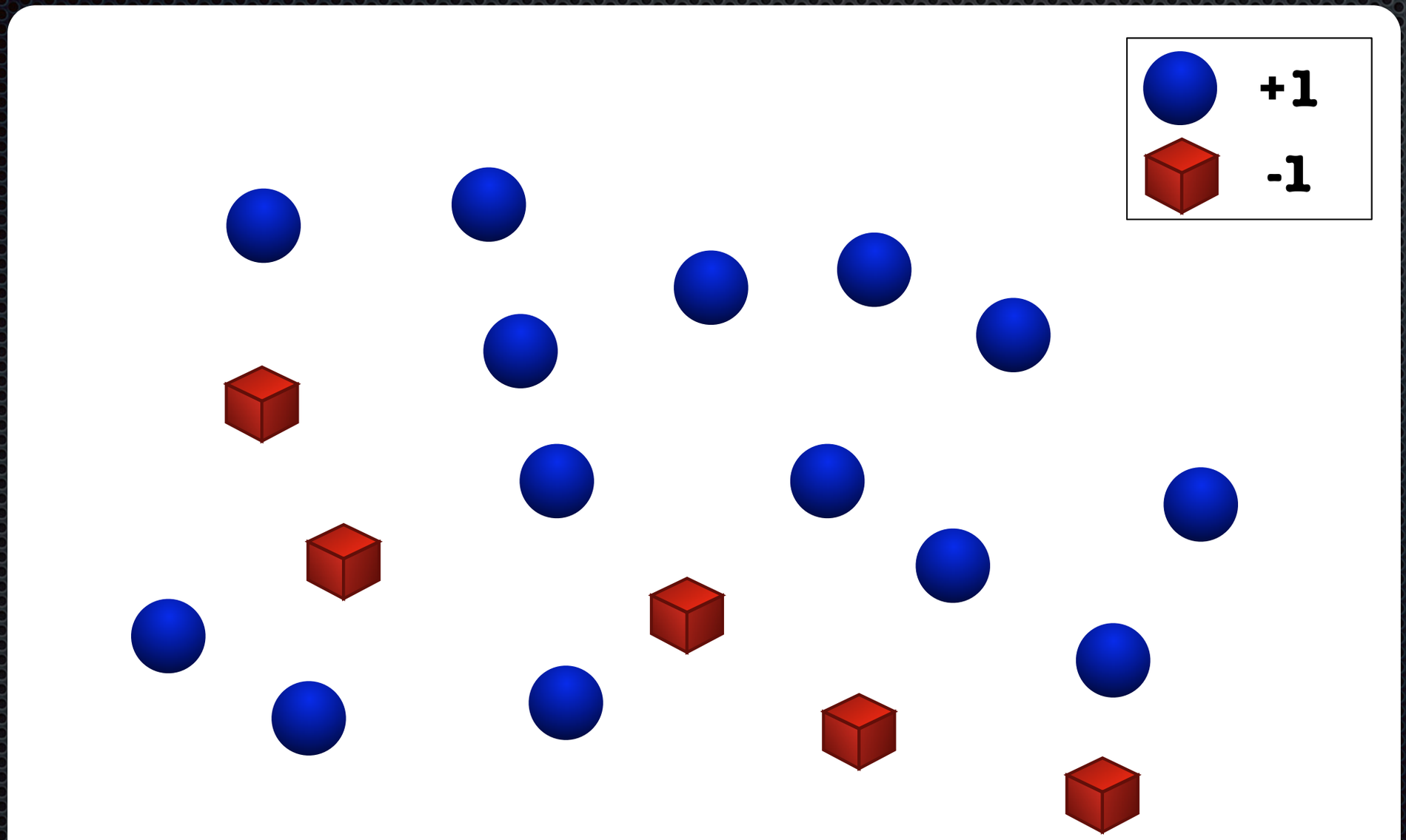
Assign most common label



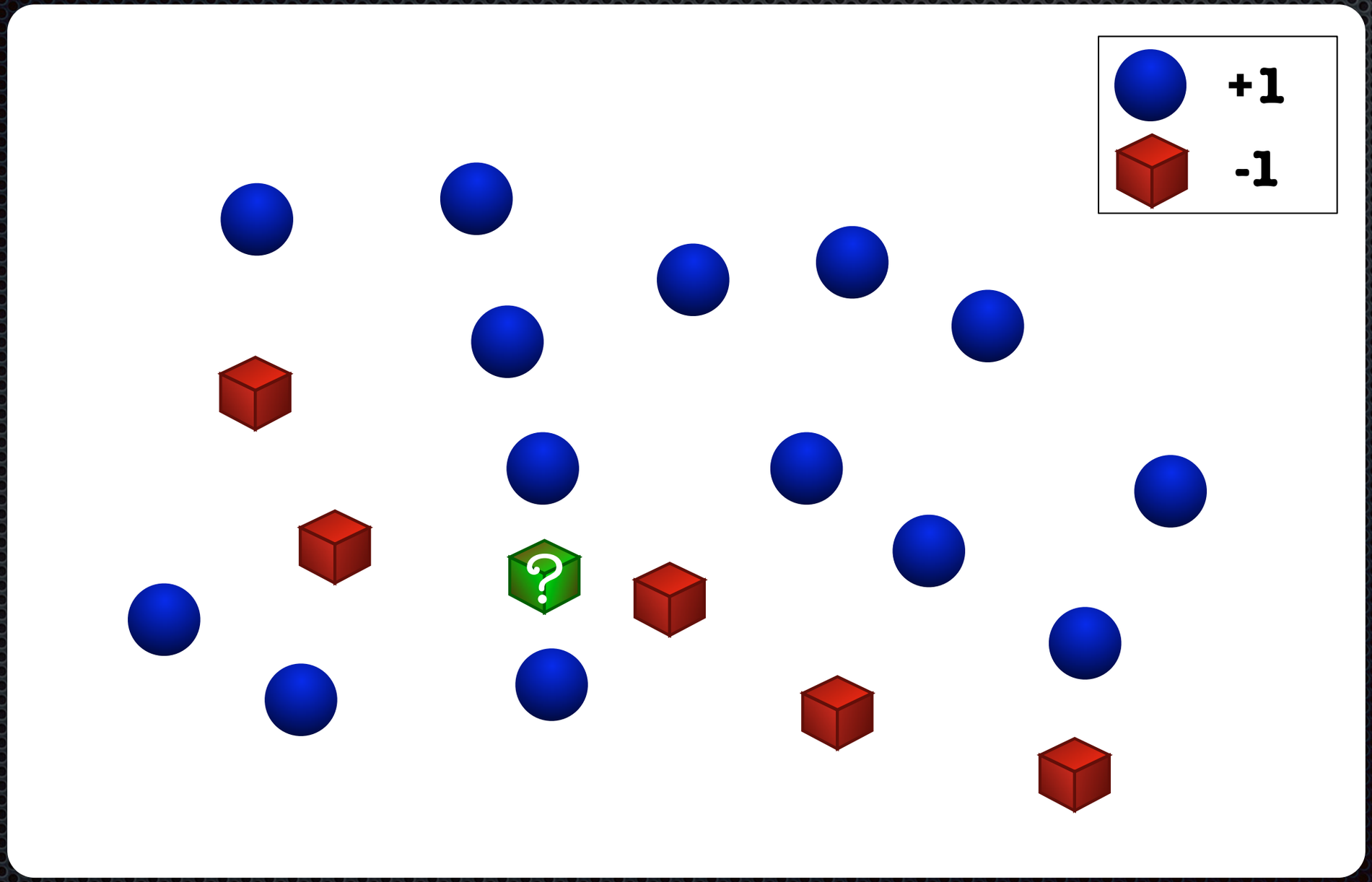
Strengths of k-NN classification

- straightforward to implement
- nonlinear decision boundaries
- complexity independent of # of classes
- asymptotic guarantees [Covert and Hart 1961]
- **relies entirely on underlying metric**

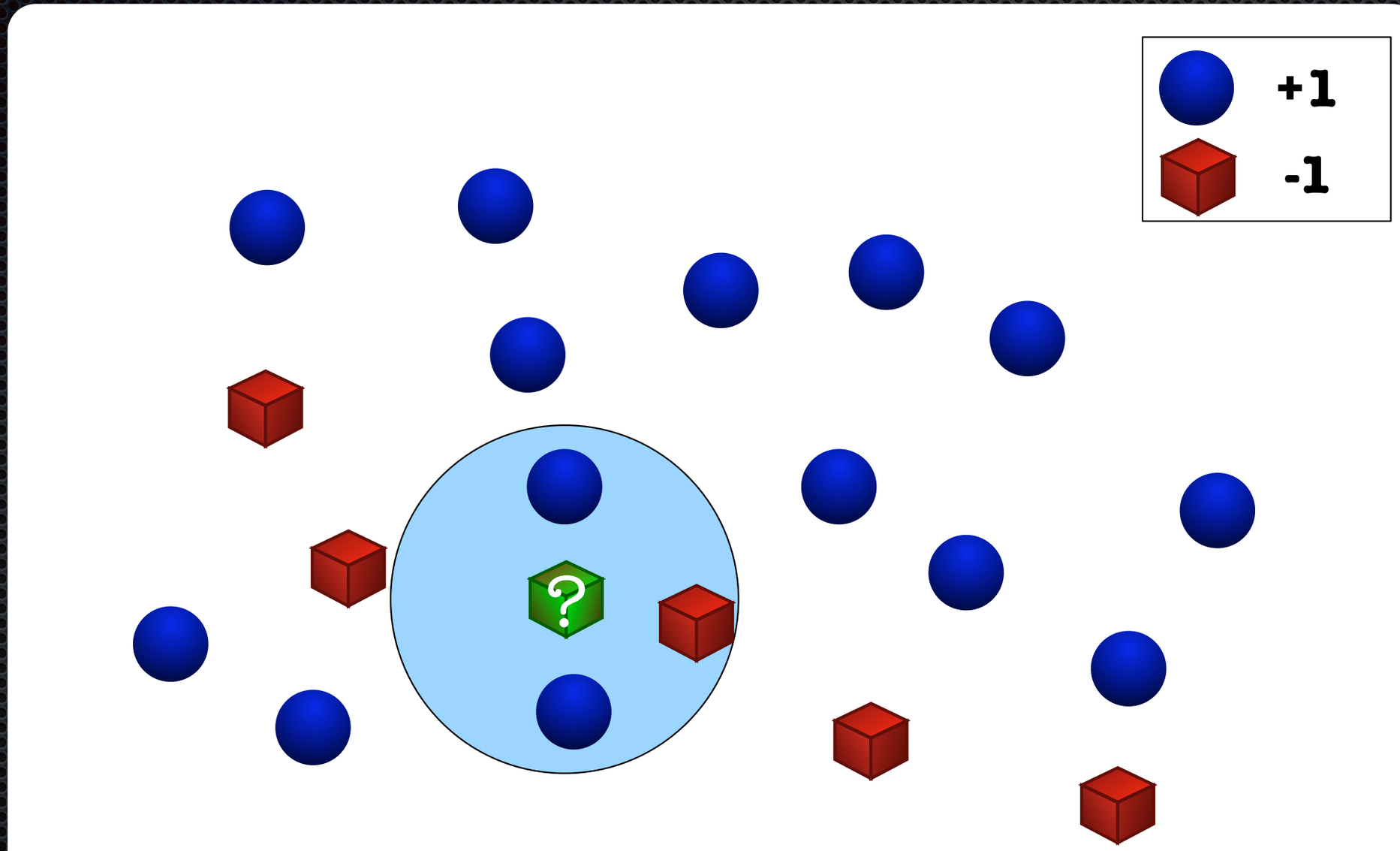
Problem with Euclidean distance



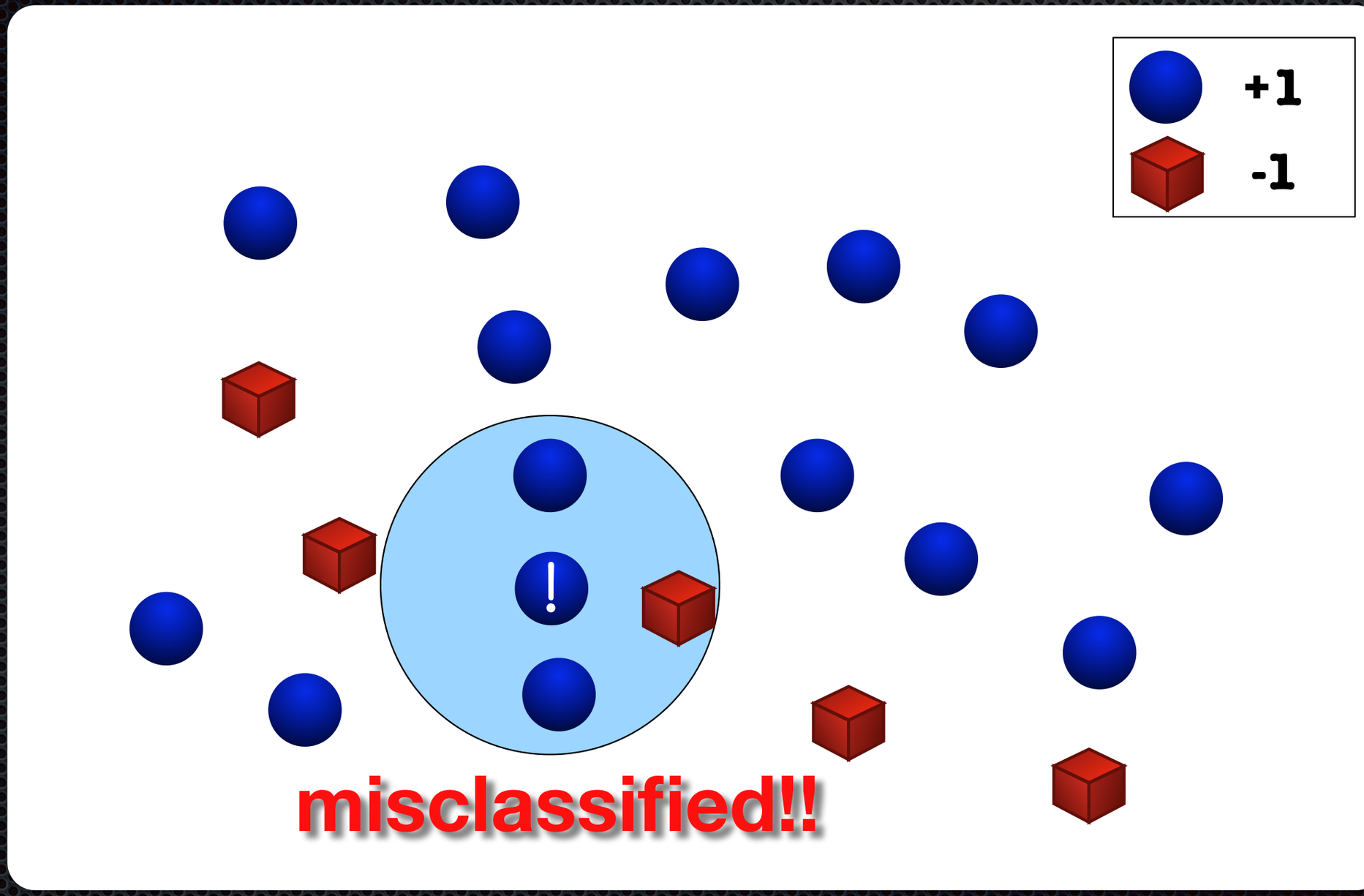
Problem with Euclidean distance



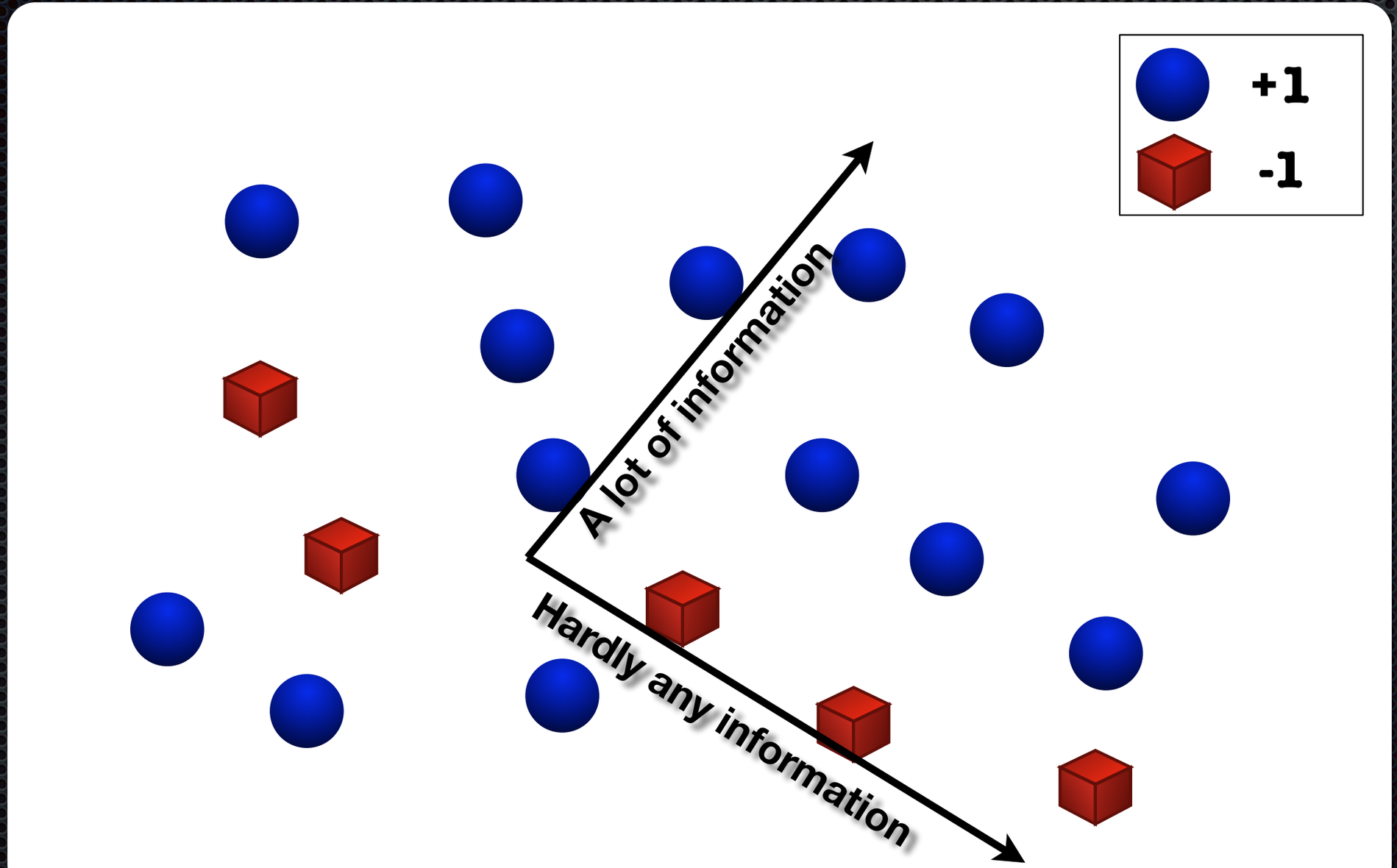
Problem with Euclidean distance



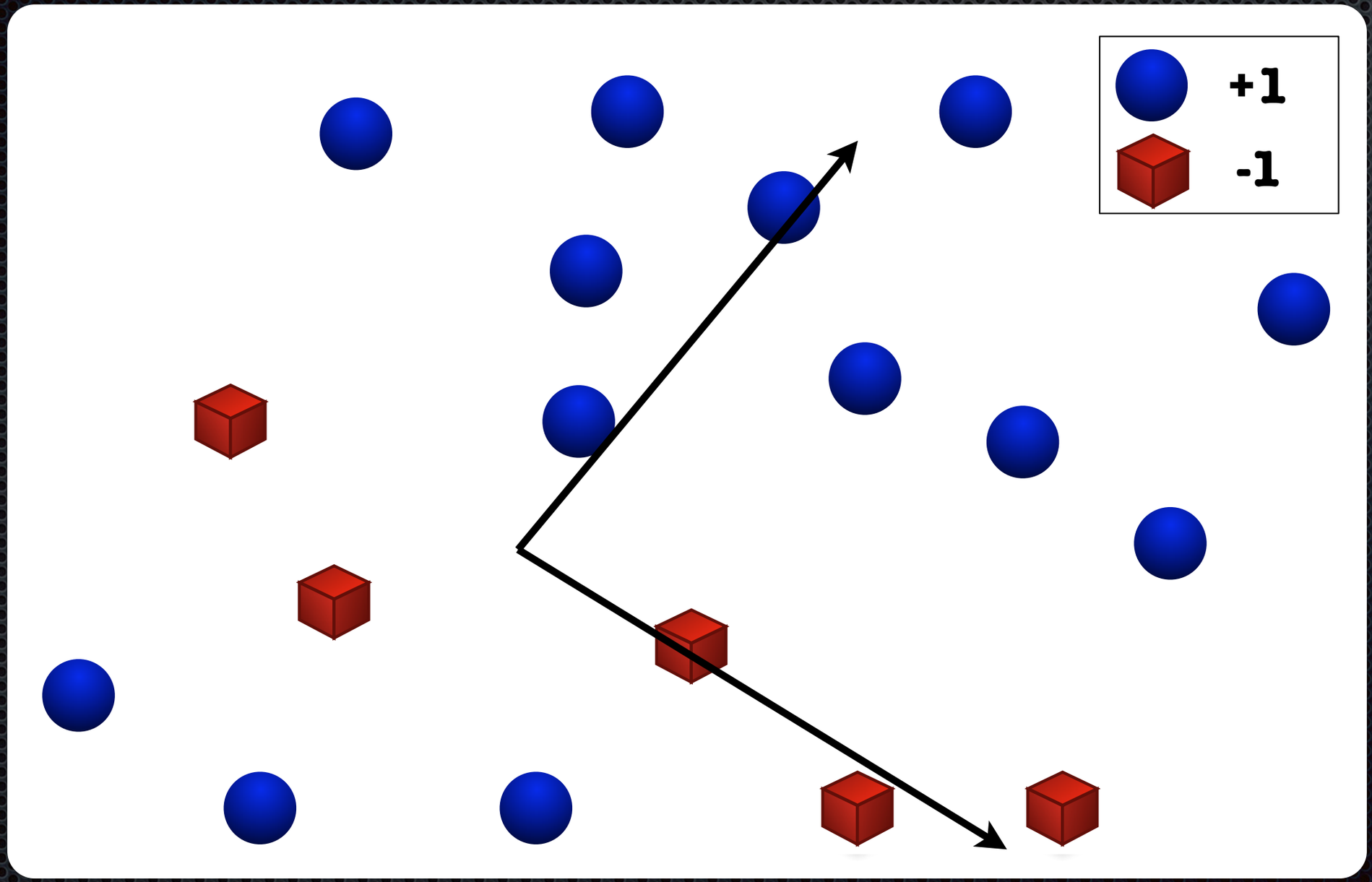
Problem with Euclidean distance



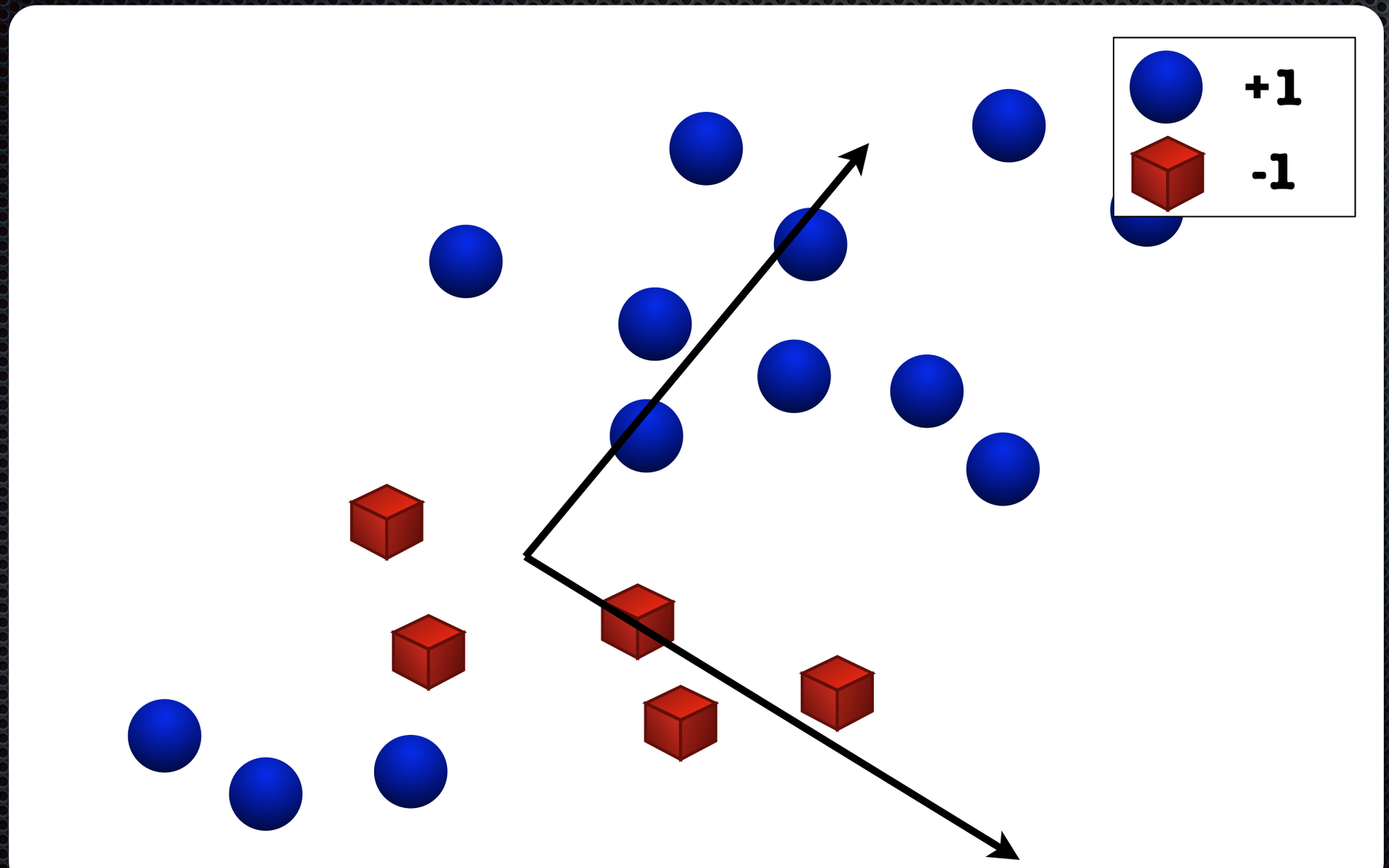
Problem with Euclidean distance



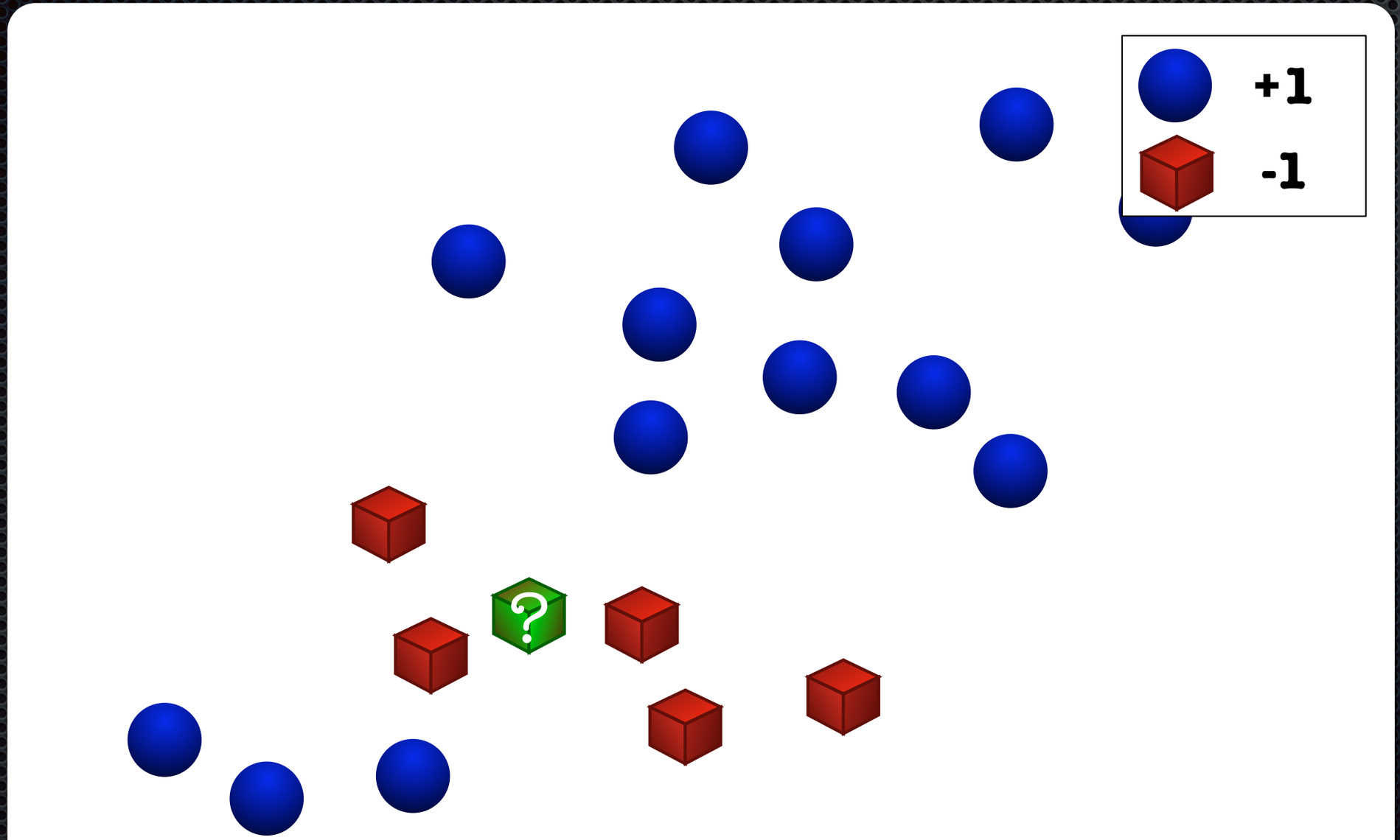
Cure: Amplify informative directions



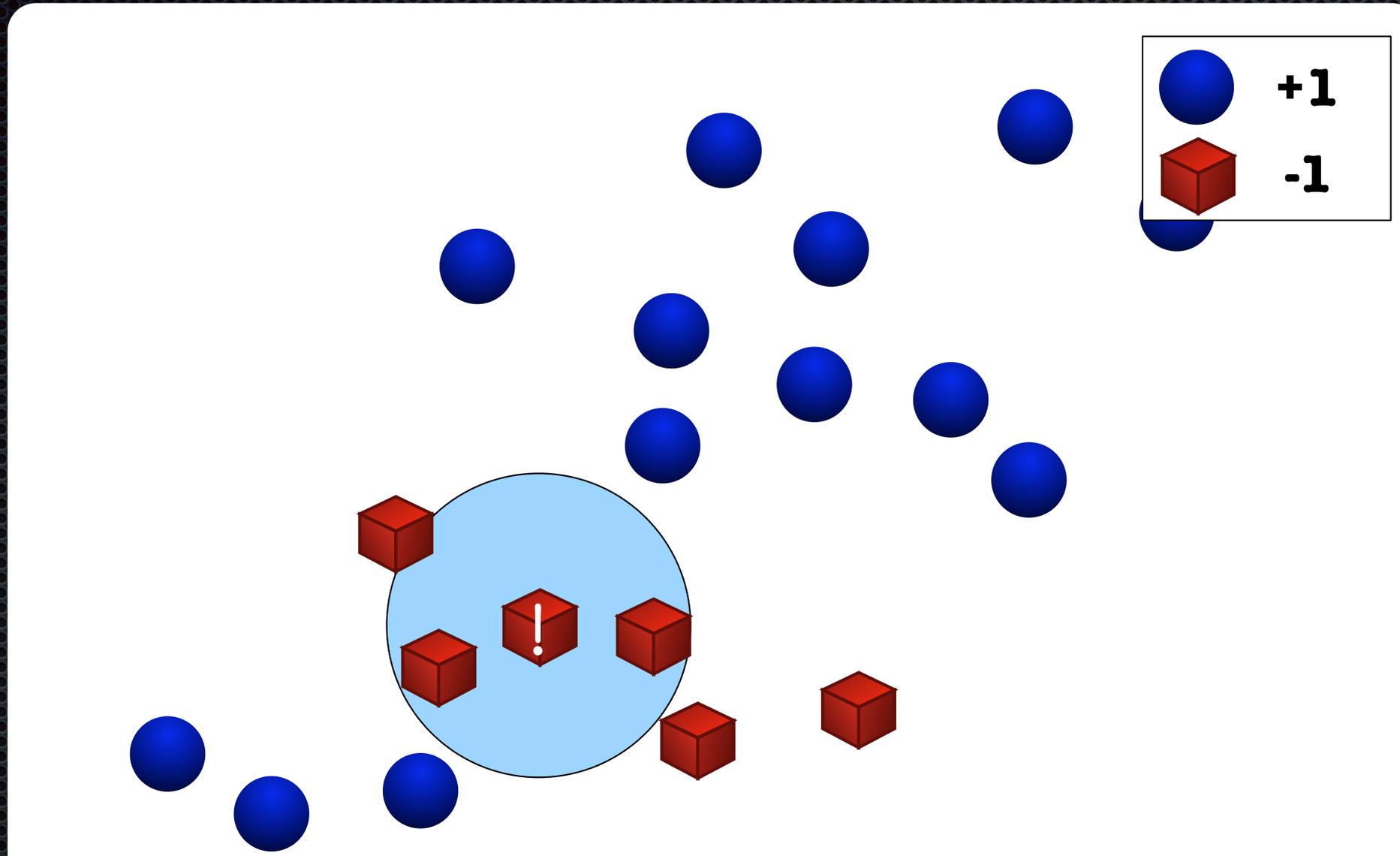
Dampen non-informative directions



Finally!!



Finally!!

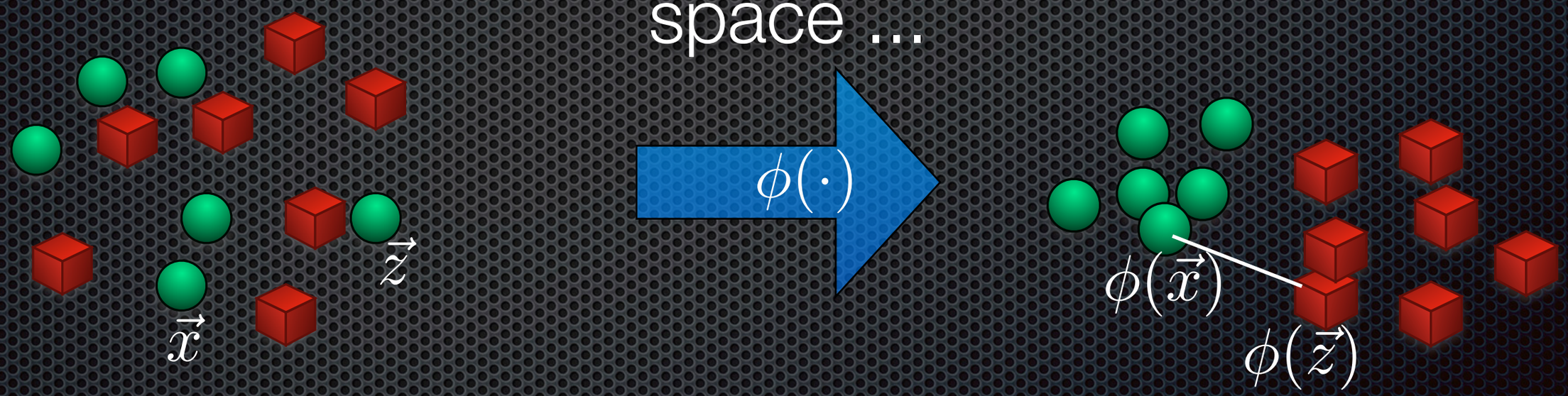


Mapping approach

Take input vectors ...

$$d(\vec{x}_i, \vec{x}_j) = \|\phi(\vec{x}_i) - \phi(\vec{x}_j)\|_2$$

... map them into some feature space ...



... and use Euclidean distance after mapping

Mapping approach

$$d(\vec{x}_i, \vec{x}_j) = \|\phi(\vec{x}_i) - \phi(\vec{x}_j)\|_2$$

Resulting metric is automatically a well-defined **pseudometric**.

requirements for pseudometric:

$d(\vec{x}, \vec{z}) \geq 0$	non-negativity
$d(\vec{x}, \vec{x}) = 0$	identity
$d(\vec{x}, \vec{z}) = d(\vec{z}, \vec{x})$	symmetry
$d(\vec{x}, \vec{z}) \leq d(\vec{x}, \vec{y}) + d(\vec{y}, \vec{z})$	triangular inequality

additional requirement for metric:

$d(\vec{x}, \vec{z}) = 0 \Leftrightarrow \vec{x} = \vec{z}$	identity of indiscernibles

Mahalanobis distance metric

The simplest mapping is a linear transformation

$$\phi : \vec{x} \rightarrow \mathbf{A}\vec{x}$$

$$d^2(\vec{x}, \vec{z}) = \|\mathbf{A}(\vec{x} - \vec{z})\|_2^2$$

$$= (\vec{x} - \vec{z})^\top \mathbf{A}^\top \mathbf{A}(\vec{x} - \vec{z})$$

$$= (\vec{x} - \vec{z})^\top \mathbf{M}(\vec{x} - \vec{z})$$

positive semi-definite

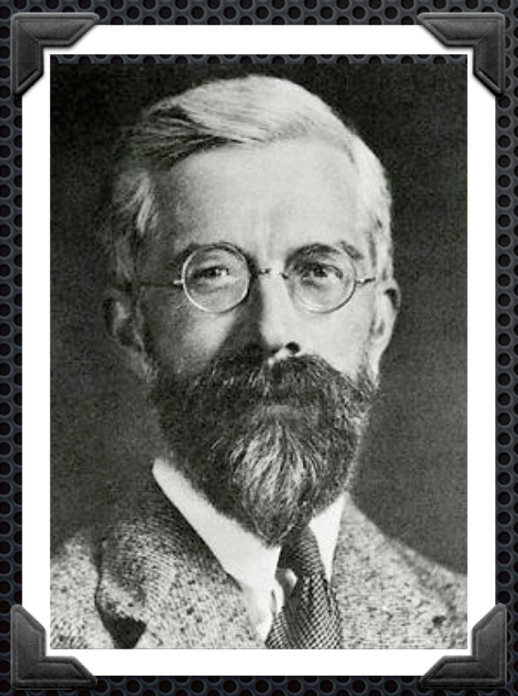
$$\mathbf{M} \succeq 0$$

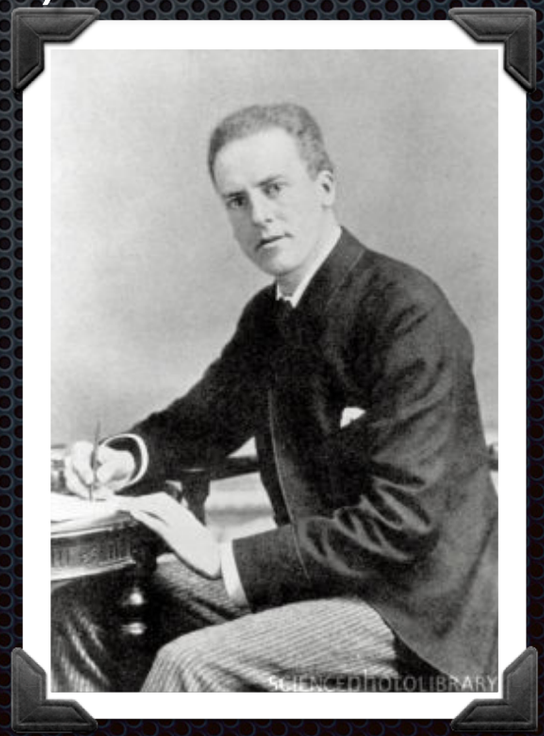
$$\mathbf{M} = \mathbf{A}^\top \mathbf{A}$$

(Generalization of Euclidean distance.)

Historic Metric Learning Algorithms

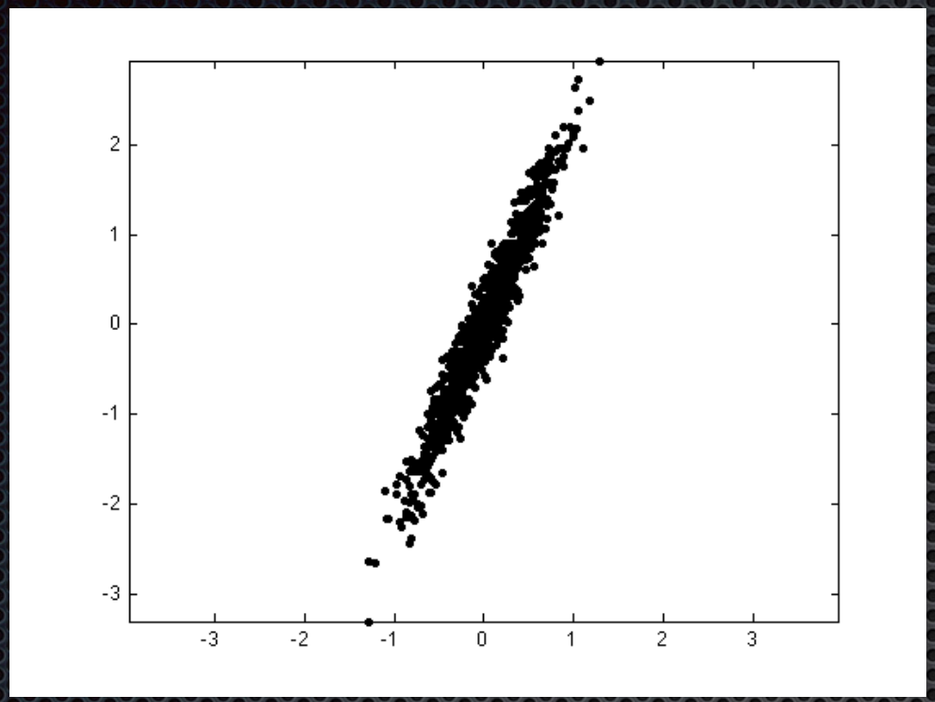
(all based on second-order statistics)

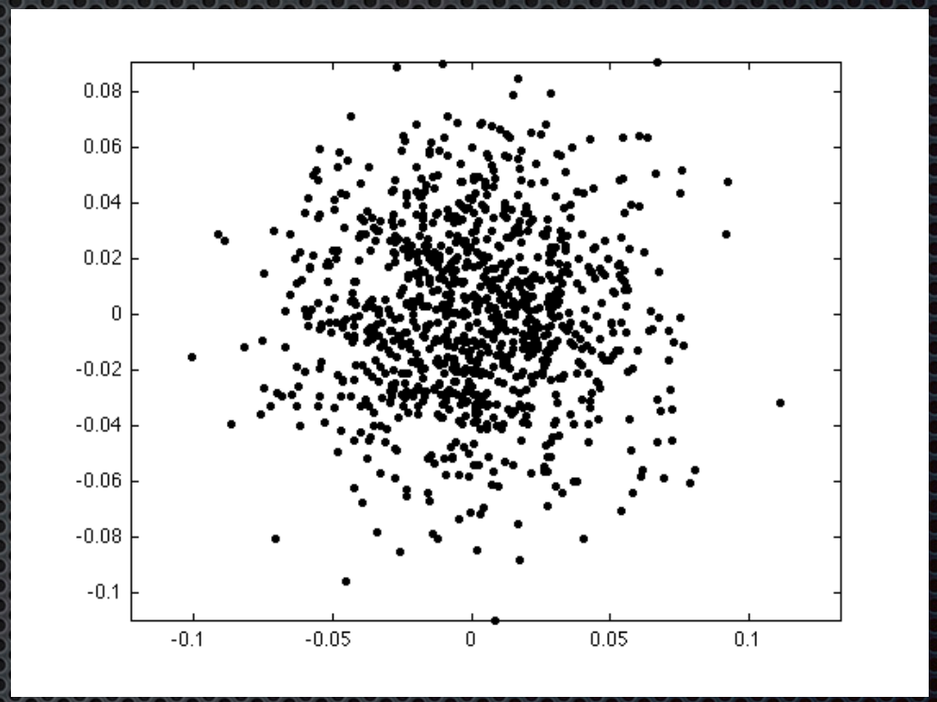




True Mahalanobis distance

(a.k.a. **whitening**)





$$M = C^{-1}$$

$$C = \sum_i (\vec{x}_i - \mu)(\vec{x}_i - \mu)^\top$$

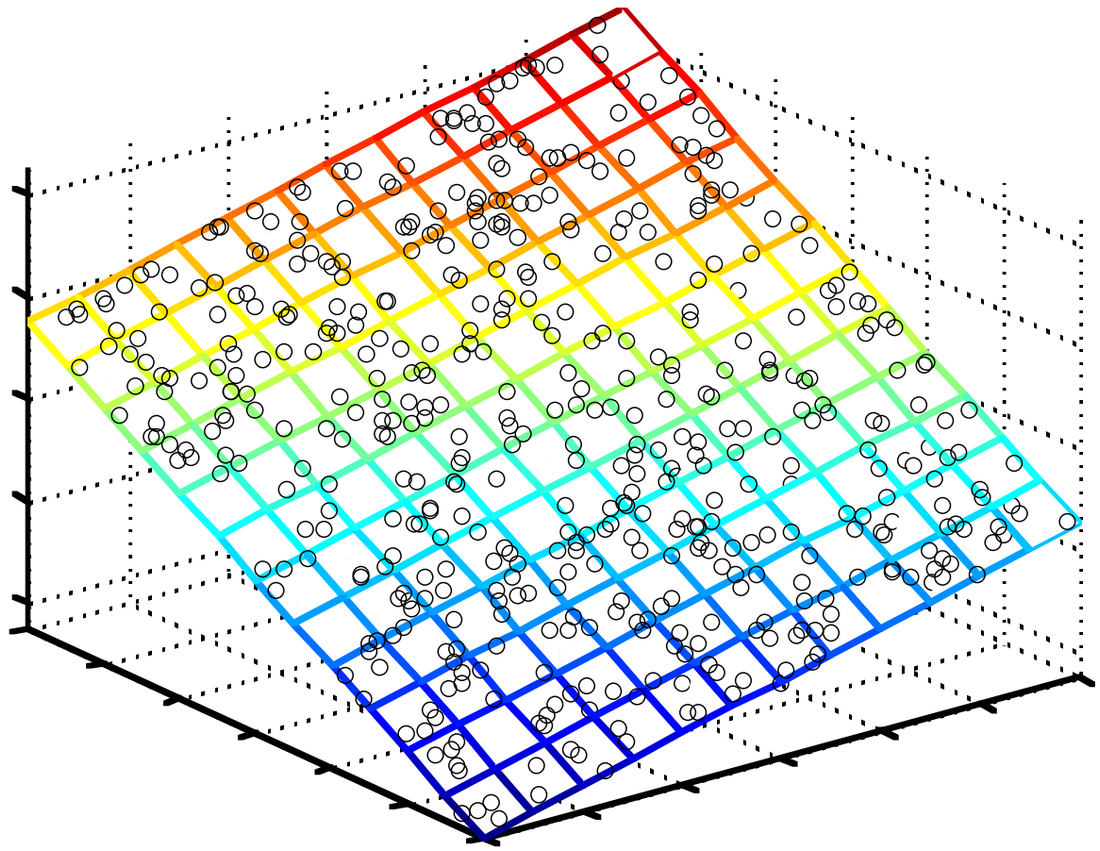
[P.C. Mahalanobis, 1936]

Principal Component Analysis

$$C = V \Delta V^\top$$

$$M = V V^\top$$

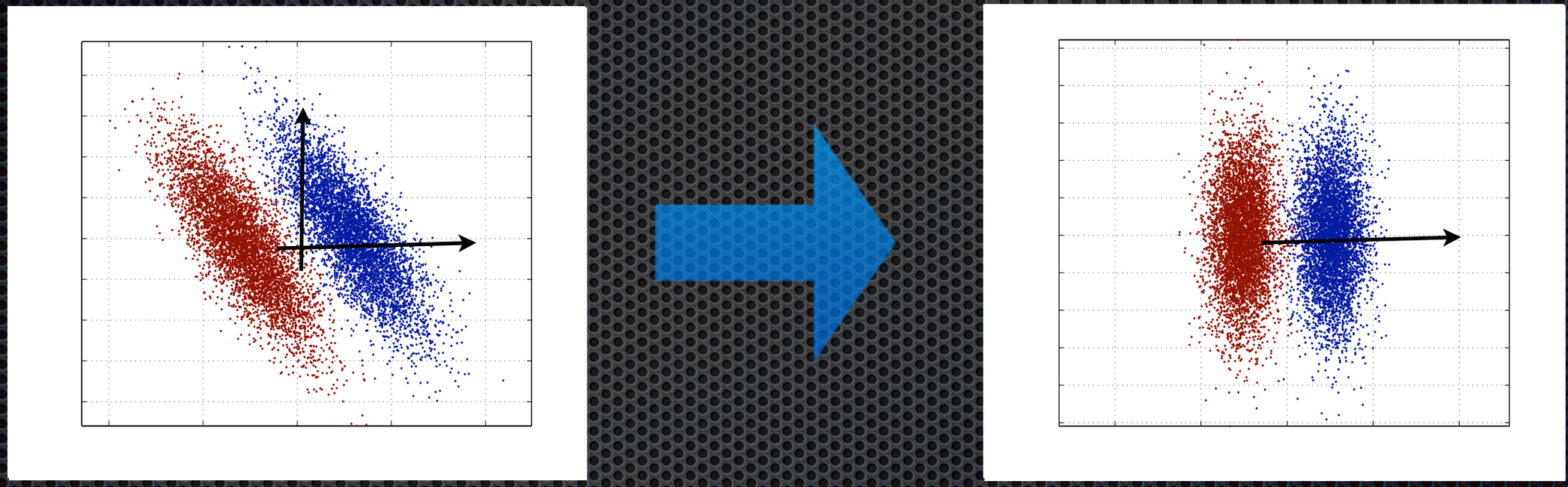
(Clip bottom k eigenvectors.)



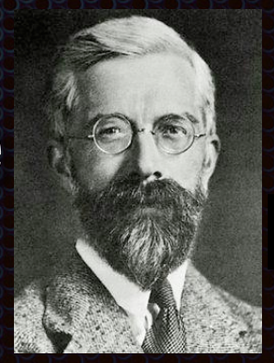
Project onto sub-space with maximum variance.

(Pearson 1901)

Liner Discriminant Analysis



Computes projection that best separates the different classes.



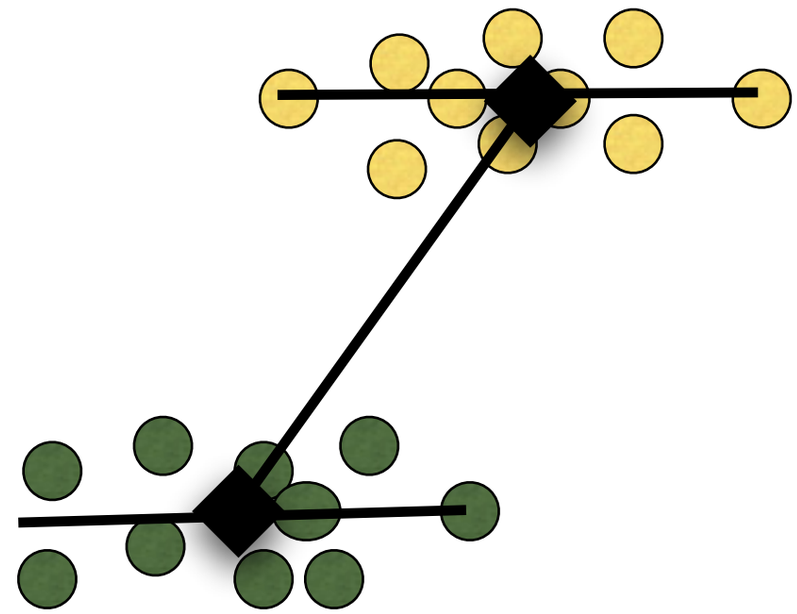
(Fisher 1936)

LDA: Optimization

between-class
covariance (weighted)

$$C_w^{-1} C_b = V \Delta V^\top$$

$$M \equiv V V^\top$$

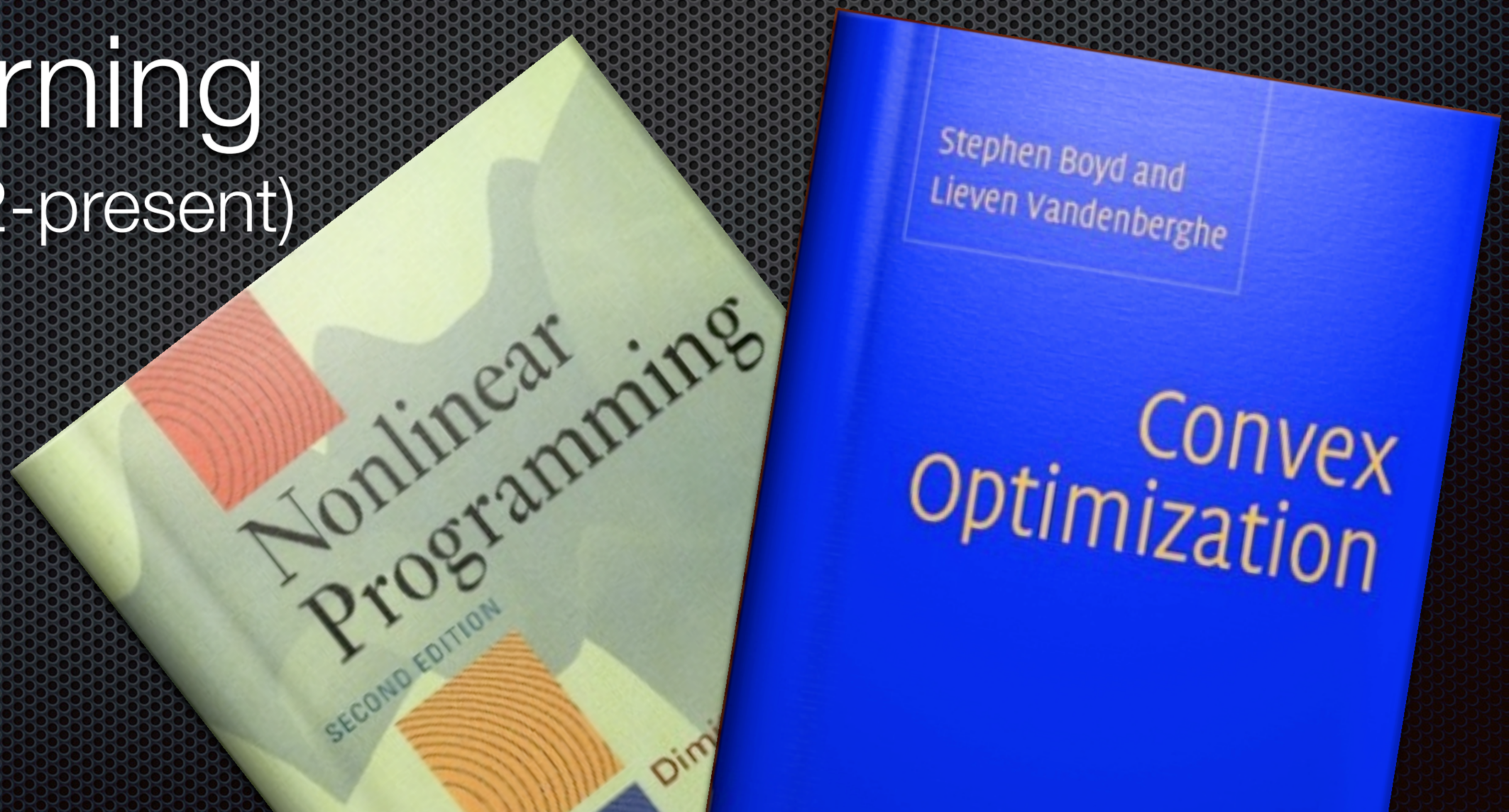


averaged within-class
covariance

PCA	Whitening	LDA
Learn “Mahalanobis” metric		
Based on second-order statistics		
Low dimensional projection		
Optimization problem		
Closed form eigenvalue solution		
unsupervised	unsupervised	supervised
not scale-free	noise sensitive	
Eigenvalue decomposition	Matrix inversion	Matrix inversion

(Clever combination of LDA + PCA + Whitening ≈ RCA [Shental et al. 2002])

Optimization based metric learning
(2002-present)



General Framework

- Constraint optimization problem
- Loss function:
  - Typically regularizer
- Constraints:
  - Enforce meaningful similarities
  - Positive semi-definiteness

Convex Optimization:

$$\min_{\mathbf{M}} \mathcal{L}(\mathbf{M})$$

$$\text{subject to:}$$

$$c_i(\mathbf{M}) \leq 0 \quad \forall c_i \in \mathcal{C}$$

$$\mathbf{M} \succeq 0$$

General Framework

- Constraint optimization problem
- Loss function:
  - Typically regularizer
- Constraints:
  - Enforce meaningful similarities
  - Positive semi-definiteness

Non-convex Optimization:

$$\min_{\mathbf{M}} \mathcal{L}(\mathbf{M})$$

$$\text{subject to:}$$

$$c_i(\mathbf{M}) \leq 0 \quad \forall c_i \in \mathcal{C}$$

$$\mathbf{M} \succeq 0$$

General Framework

- Constraint optimization problem
- Loss function:
  - Regularizer
- Approx. of kNN loss
  - Constraints:
  - Enforce meaningful similarities

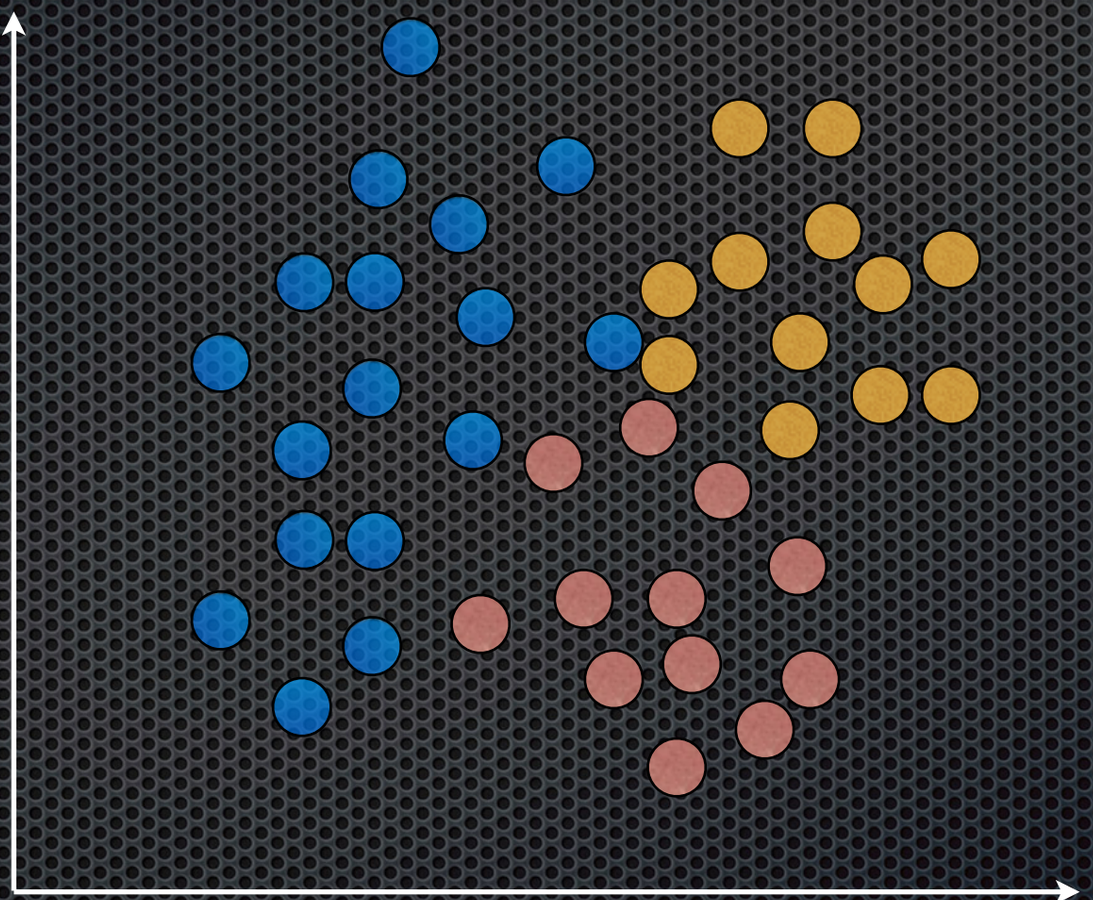
Unconstrained Optimization:

$$\min_{\mathbf{A}} \mathcal{L}(\mathbf{A}) + \sum_{c_i \in \mathcal{C}} c_i(\mathbf{A})$$

subject to:

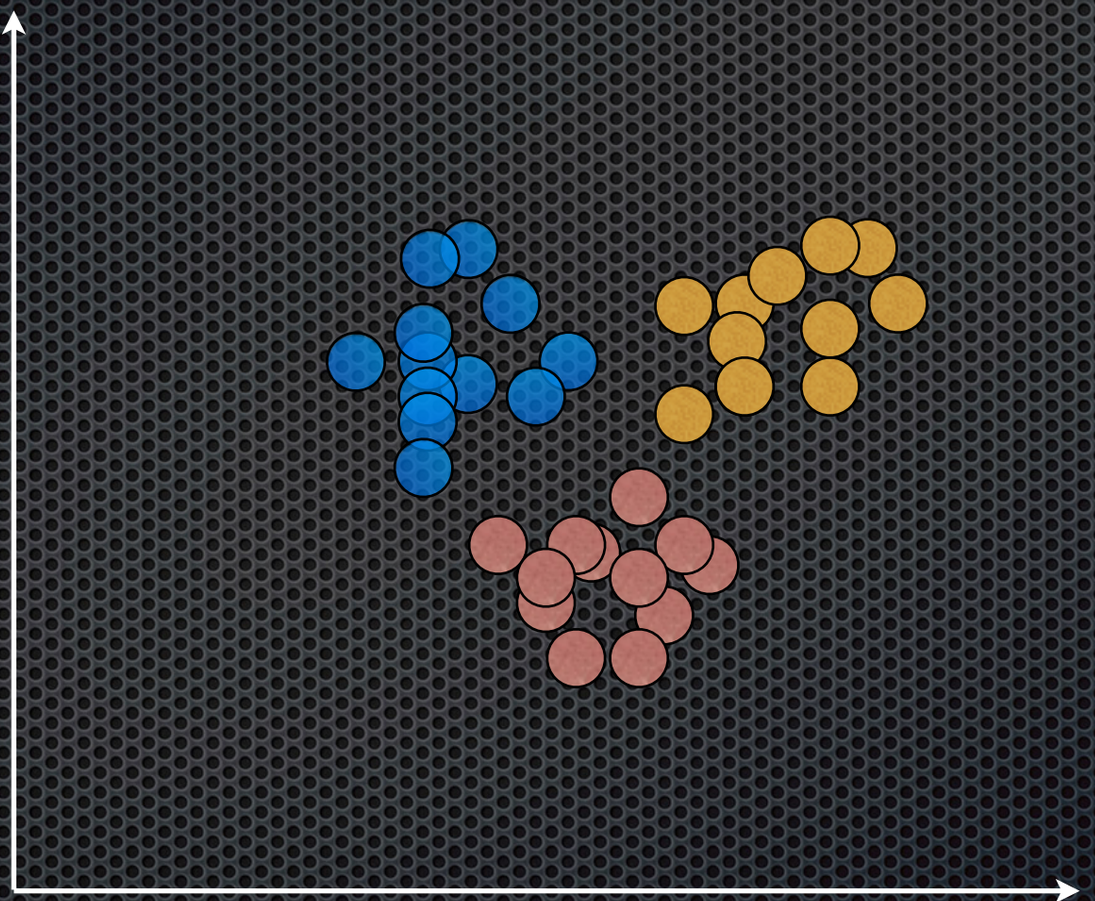
$$c_i(\mathbf{A}) \leq 0 \quad \forall c_i \in \mathcal{C}$$

Mahalanobis Metric for Clustering (MMC)



[Xing et al. (2002)]

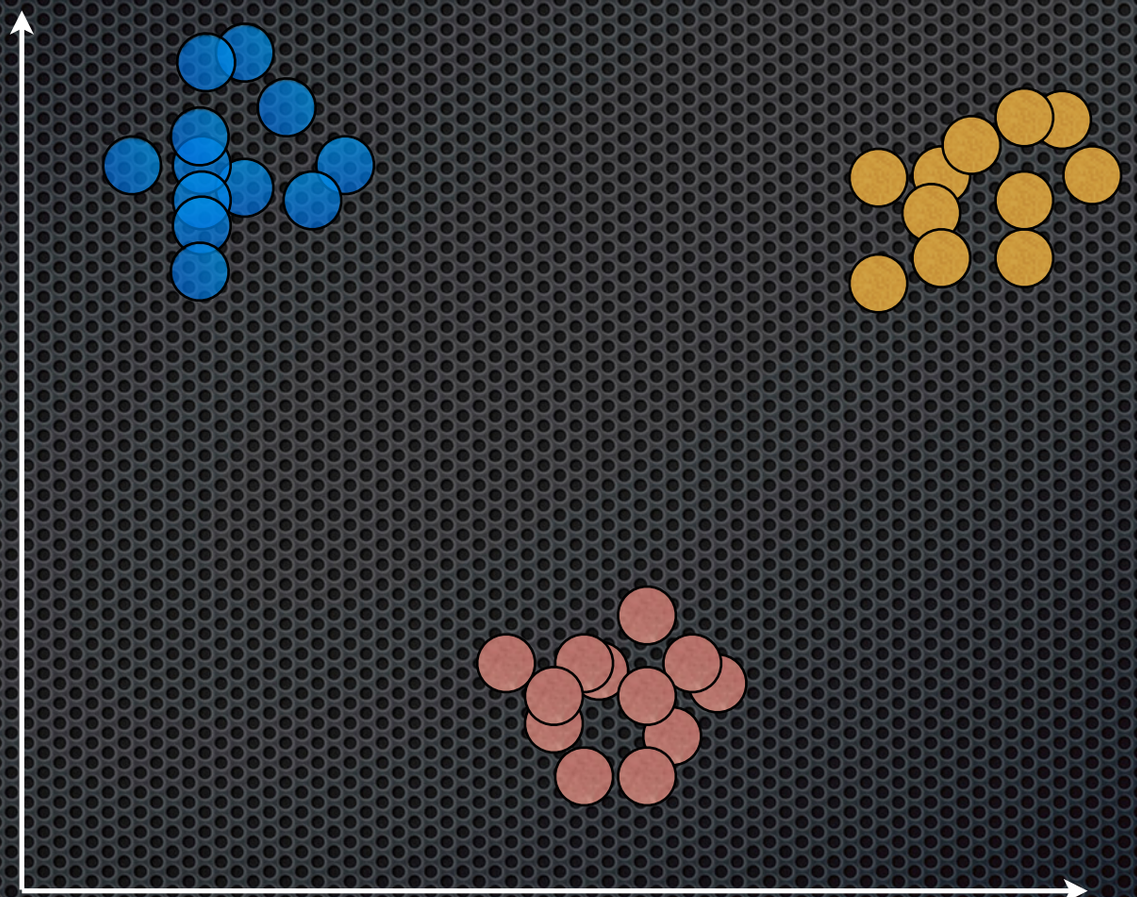
Mahalanobis Metric for Clustering (MMC)



Move **similarly** labeled inputs **together**

[Xing et al. (2002)]

Mahalanobis Metric for Clustering (MMC)



Move **different** labeled inputs **apart**

[Xing et al. (2002)]

Convex optimization problem

$$\mathcal{S} = \{(i,j) \mid \vec{x}_i \text{ and } \vec{x}_j \text{ are similarly labeled}\}$$

$$\mathcal{D} = \{(i,j) \mid \vec{x}_i \text{ and } \vec{x}_j \text{ are differently labeled}\}$$

$$\vec{x}_{ij} = (\vec{x}_i - \vec{x}_j)$$

$$\underset{\mathbf{M}}{\textbf{Maximize}} \sum_{(i,j)\in\mathcal{D}} \sqrt{\vec{x}_{ij}^{\top}\mathbf{M}\vec{x}_{ij}}$$

subject to:

$$\textbf{(1)} \sum_{(i,j)\in\mathcal{S}} \vec{x}_{ij}^{\top}\mathbf{M}\vec{x}_{ij} \leq 1$$

$$\textbf{(2)}\ \mathbf{M} \succeq 0.$$

Convex optimization problem

$$\underset{\mathbf{M}}{\text{Maximize}} \sum_{(i,j)\in\mathcal{D}} \sqrt{\vec{x}_{ij}^{\top}\mathbf{M}\vec{x}_{ij}}$$

pushing differently labeled inputs apart

$$\text{subject to:}$$

$$(1)\ \sum_{(i,j)\in\mathcal{S}} \vec{x}_{ij}^{\top}\mathbf{M}\vec{x}_{ij} \leq 1$$

$$(2)\ \mathbf{M} \succeq 0.$$

Convex optimization problem

$$\underset{\mathbf{M}}{\textbf{Maximize}} \sum_{(i,j)\in\mathcal{D}} \sqrt{\vec{x}_{ij}^{\top}\mathbf{M}\vec{x}_{ij}}$$

subject to:

$$(1)\ \sum_{(i,j)\in\mathcal{S}} \vec{x}_{ij}^{\top}\mathbf{M}\vec{x}_{ij} \leq 1$$

$$(2)\ \mathbf{M} \succeq 0.$$

pulling similar points together

Convex optimization problem

$$\underset{\mathbf{M}}{\textbf{Maximize}} \sum_{(i,j)\in\mathcal{D}} \sqrt{\vec{x}_{ij}^{\top}\mathbf{M}\vec{x}_{ij}}$$

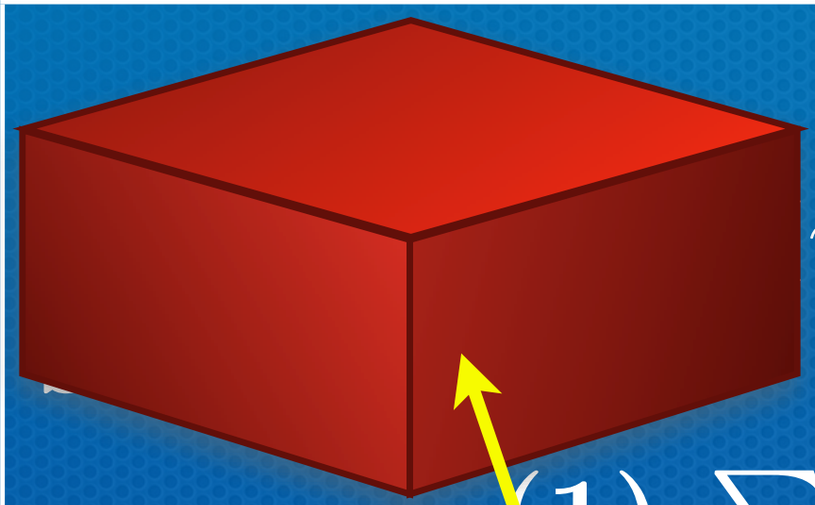
subject to:

$$(1)\ \sum_{(i,j)\in\mathcal{S}} \vec{x}_{ij}^{\top}\mathbf{M}\vec{x}_{ij} \leq 1$$

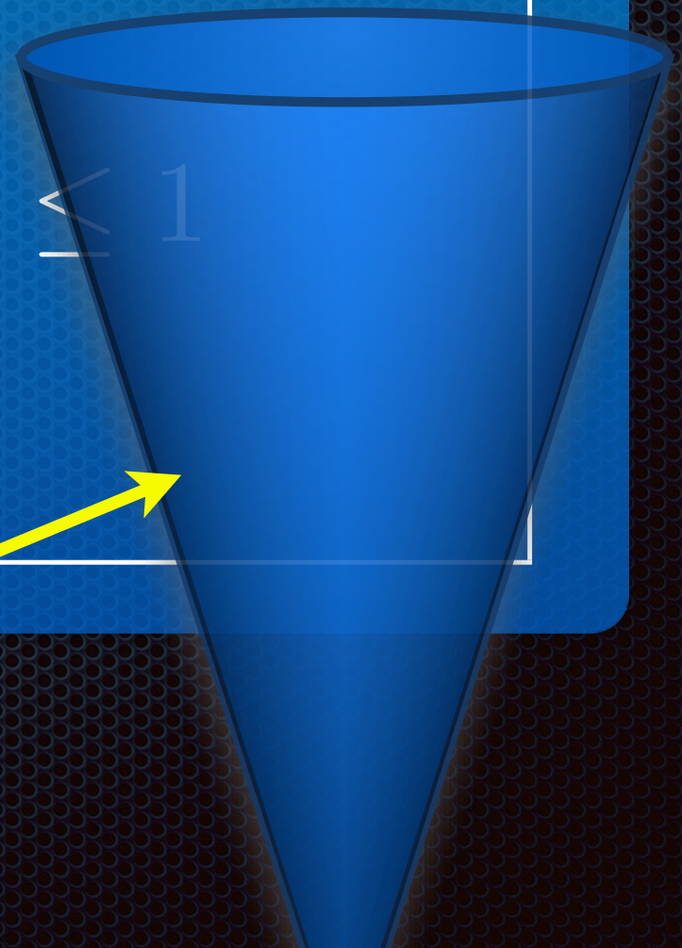
$$(2)\ \mathbf{M} \succeq 0.$$

ensuring positive semi-definiteness

Convex optimization problem



$$i,j)\in\mathcal{D}\ \sqrt{\vec{x}_{ij}^{\top}\mathbf{M}\vec{x}_{ij}}$$

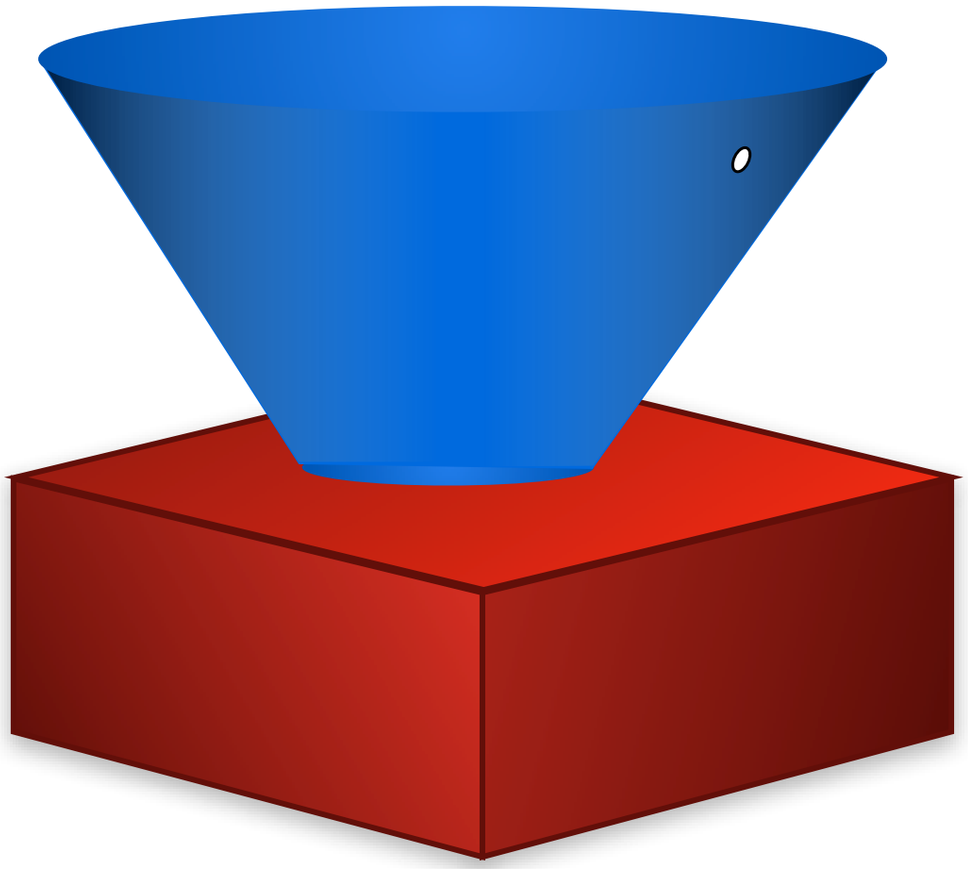


$$\textbf{(1)}\ \sum_{(i,j)\in\mathcal{S}}\vec{x}_{ij}^{\top}\mathbf{M}\vec{x}_{ij}\leq 1$$

$$\textbf{(2)}\ \mathbf{M}\succeq 0.$$

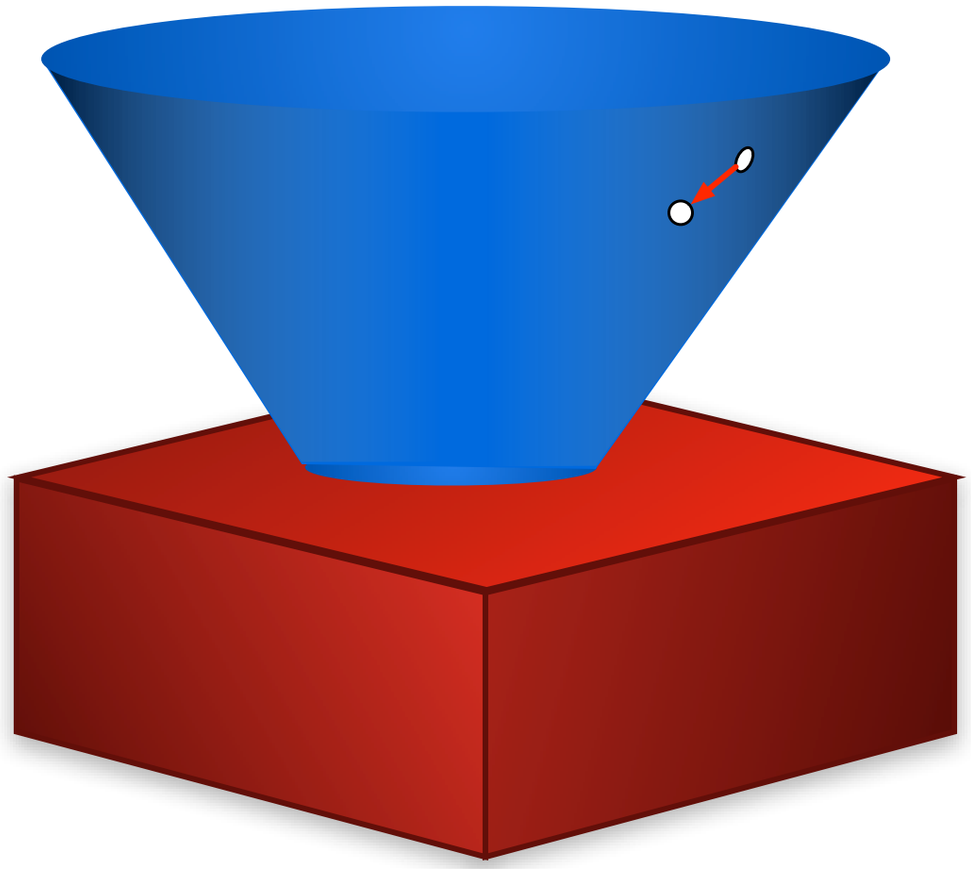
CONVEX

Gradient Alternating Projection



Gradient Alternating Projection

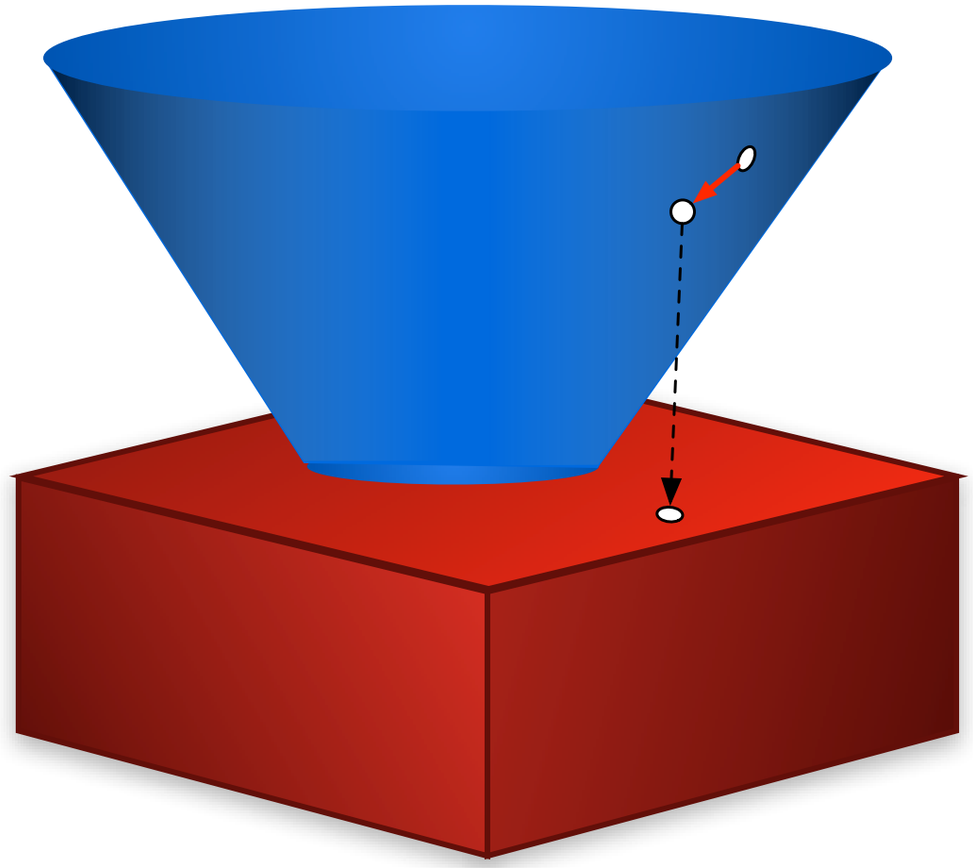
Take step along gradient.



Gradient Alternating Projection

Take step along gradient.

Project onto constraint satisfying sub-space.

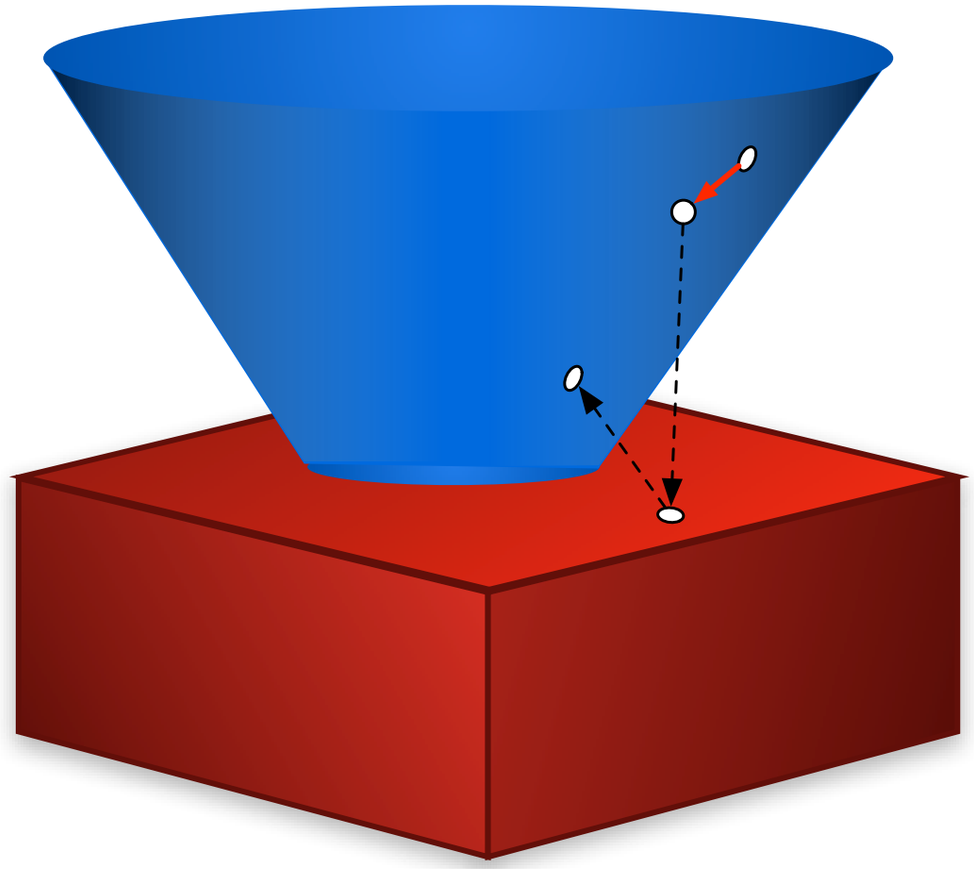


Gradient Alternating Projection

Take step along gradient.

Project onto constraint satisfying sub-space.

Project onto PSD cone.



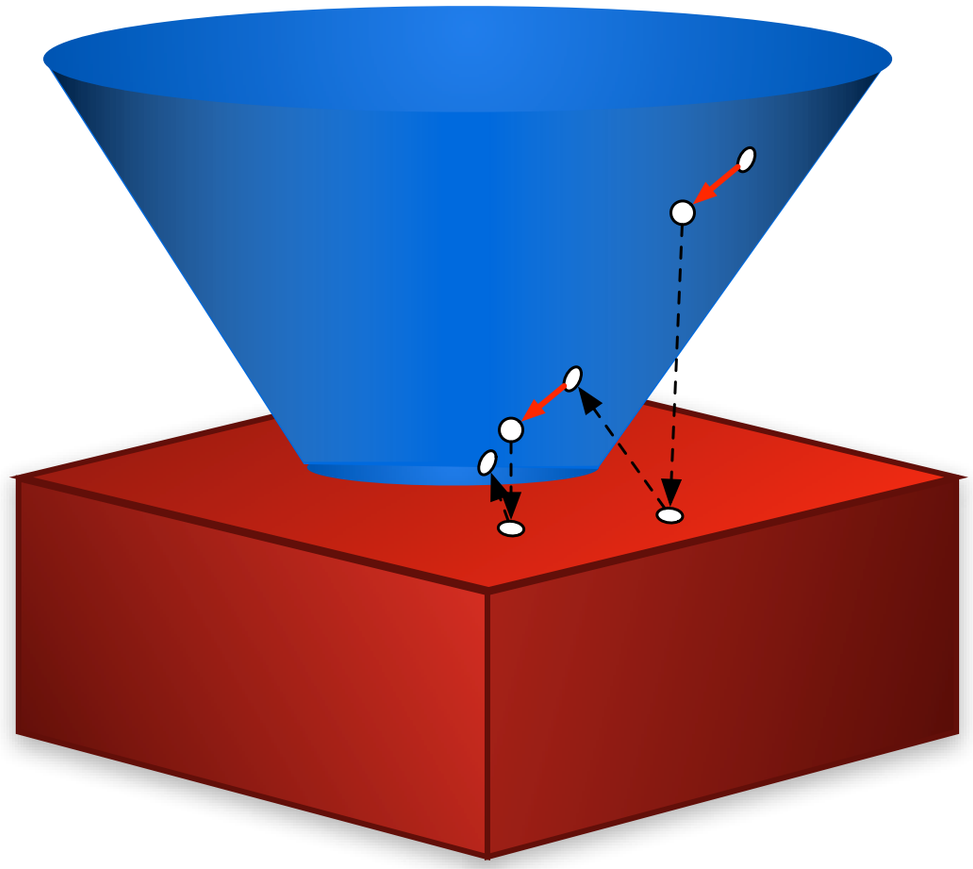
Gradient Alternating Projection

Take step along gradient.

Project onto constraint satisfy sub-space.

Project onto PSD cone.

REPEAT



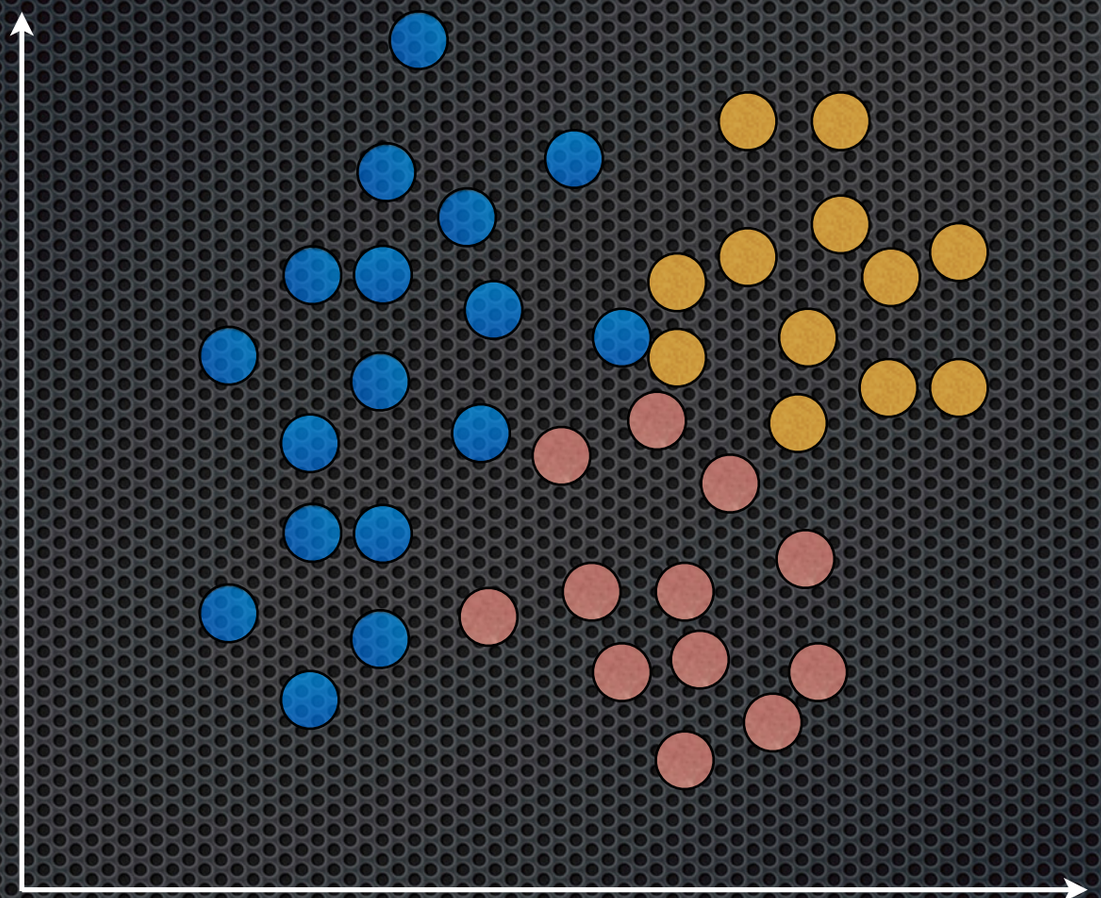
Algorithm is guaranteed to converge to optimal solution

Summary of Xing et al 2002

- **First Metric Learning SDP**
- Learns Mahalanobis metric
- Very suited for clustering
- Can be kernelized
- Optimization problem is convex
- Algorithm is guaranteed to converge
- Assumes data to be uni-modal
- Slow convergence (does not scale to large data)

POLA

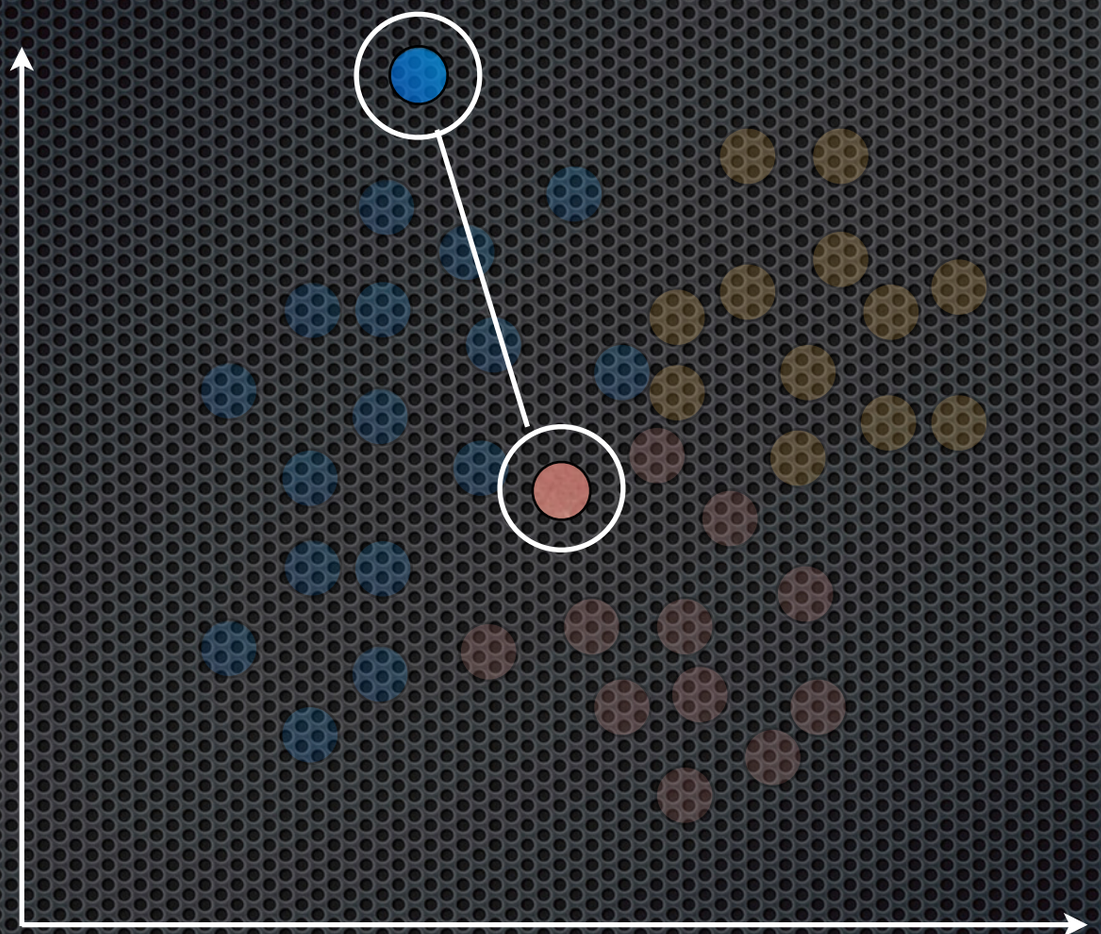
(Pseudo-metric online learning algorithm)



[Shalev-Shwartz et al.; ICML 2004]

POLA

(Pseudo-metric online learning algorithm)

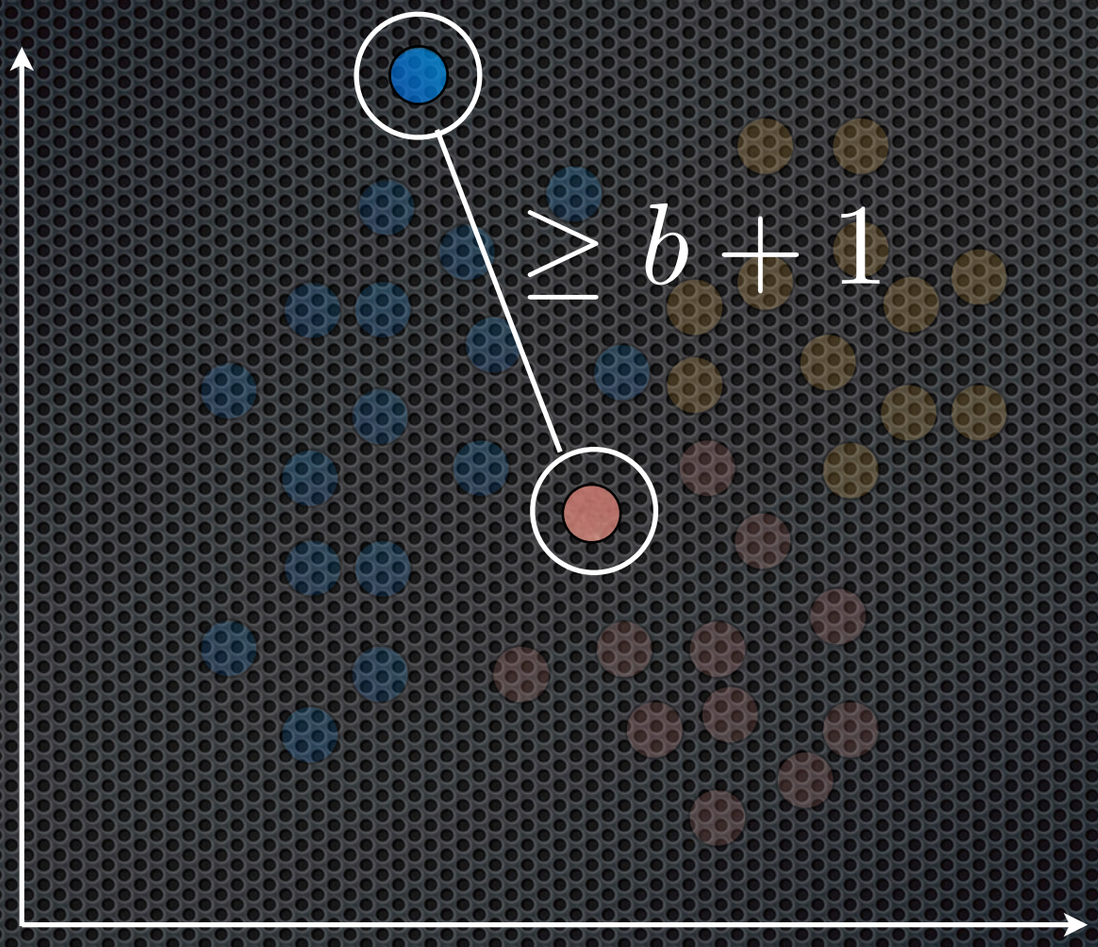


This time the inputs are accessed two at a time.

[Shalev-Shwartz et al.; ICML 2004]

POLA

(Pseudo-metric online learning algorithm)

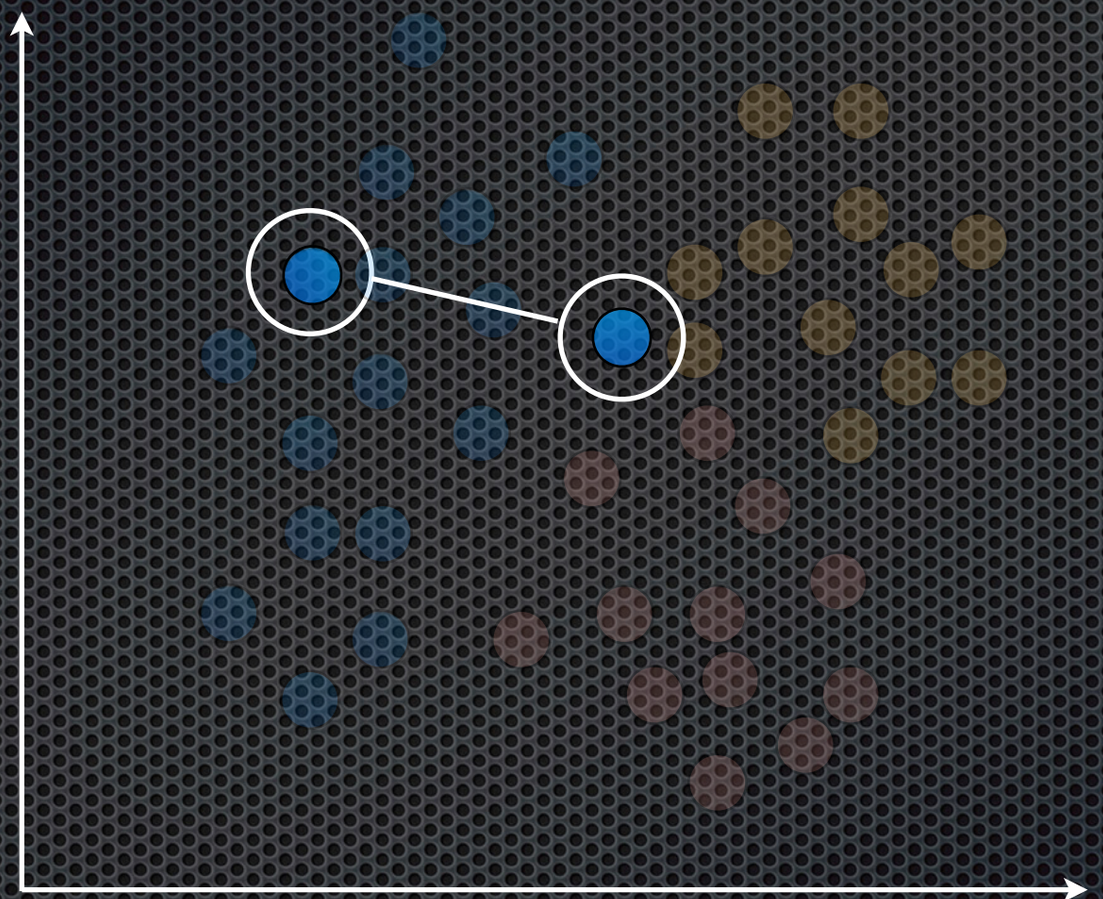


This algorithm is **Differently** labeled inputs are **separated**.

[Shalev-Shwartz et al.; ICML 2004]

POLA

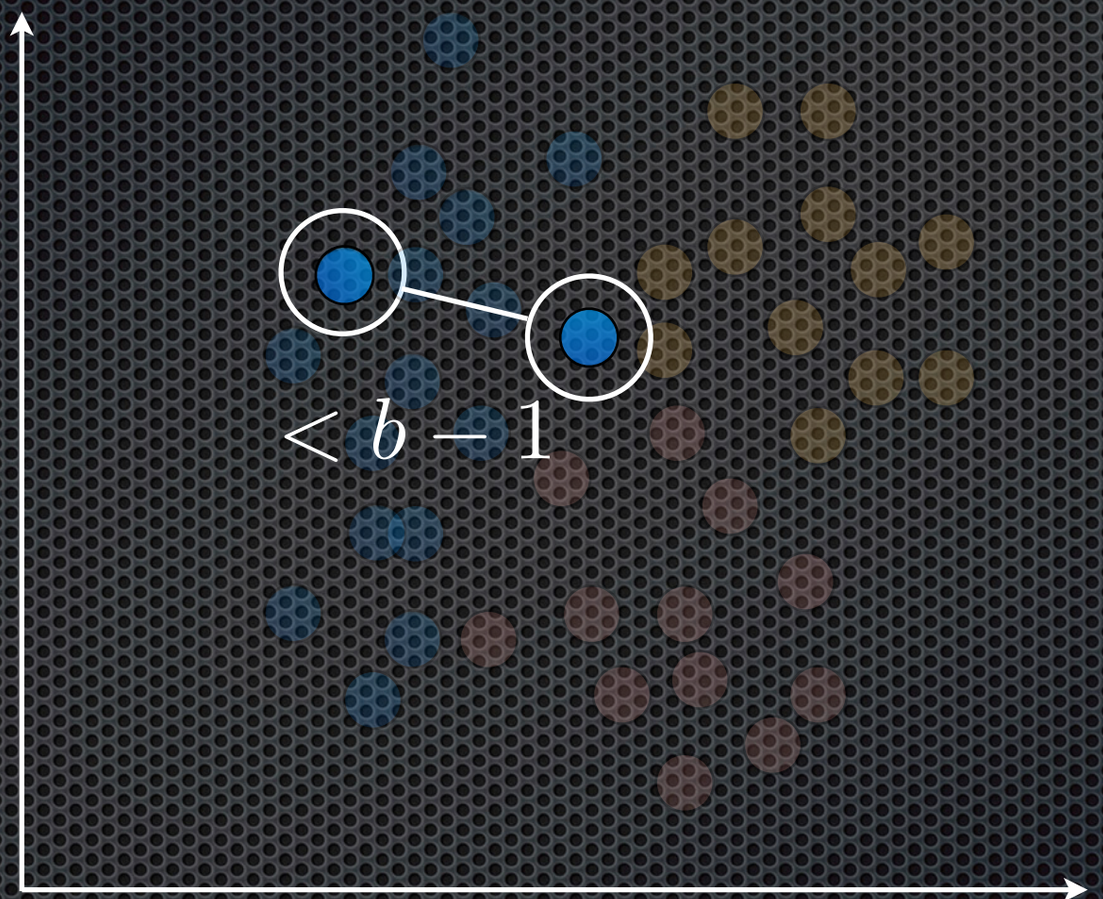
(Pseudo-metric online learning algorithm)



[Shalev-Shwartz et al.; ICML 2004]

POLA

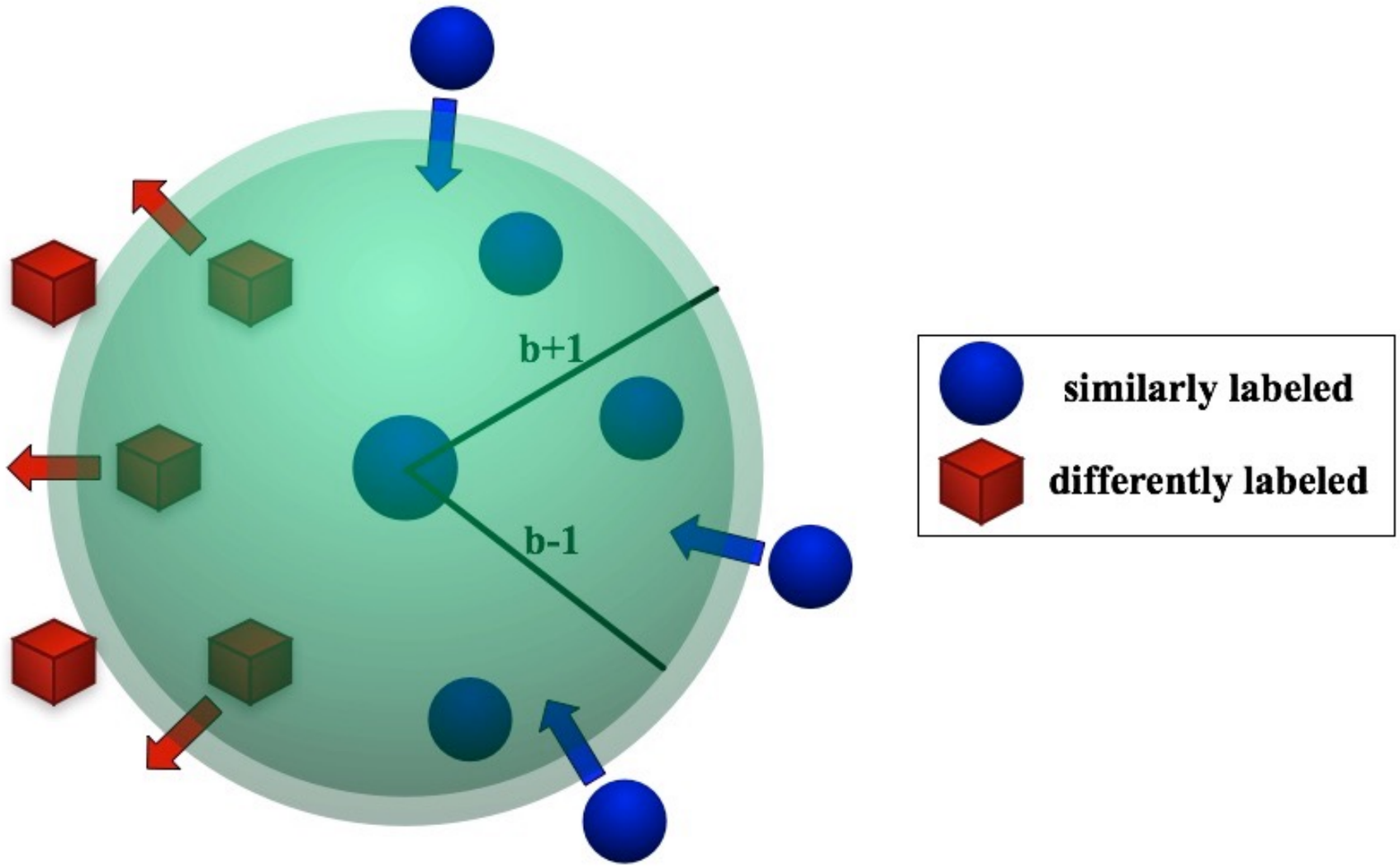
(Pseudo-metric online learning algorithm)



Similarly labeled inputs are **moved closer**.

[Shalev-Shwartz et al.; ICML 2004]

Margin



Convex optimization

At each time t, we get two [illegible]s: $\vec{u}_t$, $\vec{v}_t$

[illegible]nt 1:

$$y_t(b_t - \vec{x}^{\top} \vec{a}_t) \geq$$

Constraint 2:

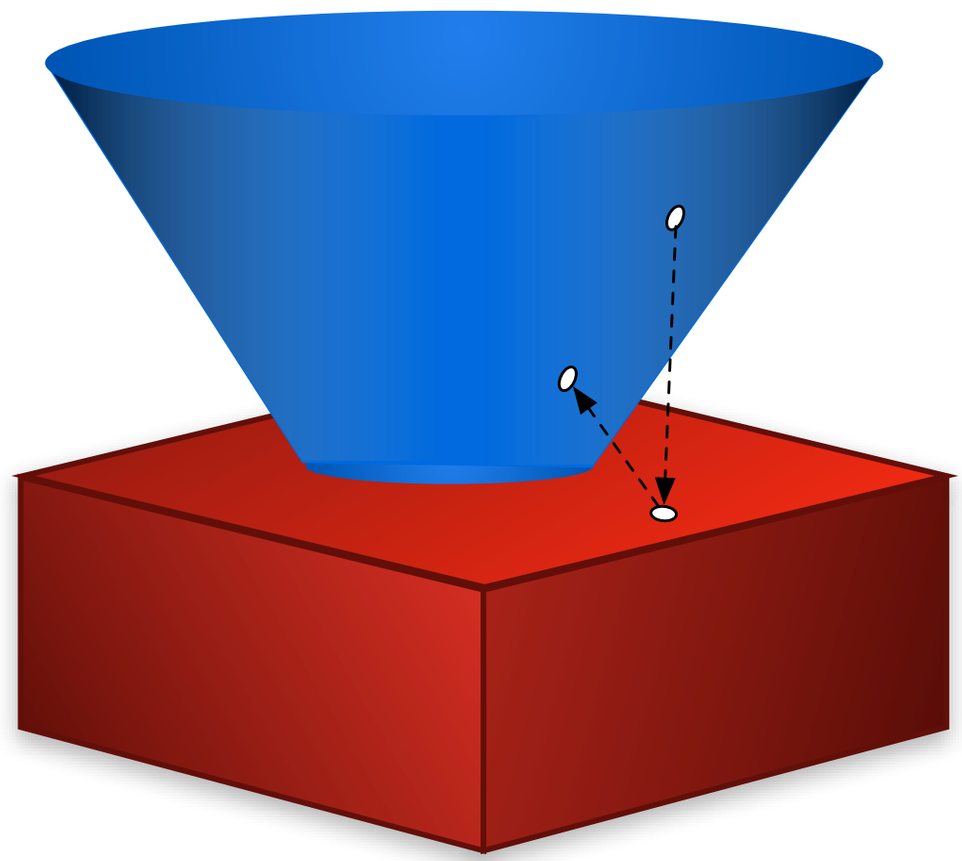
$$\mathbf{M} \succeq 0$$

Both are convex!!

Alternating Projection

Initialize inside PSD cone

Project onto constraint - satisfying hyperplane and back

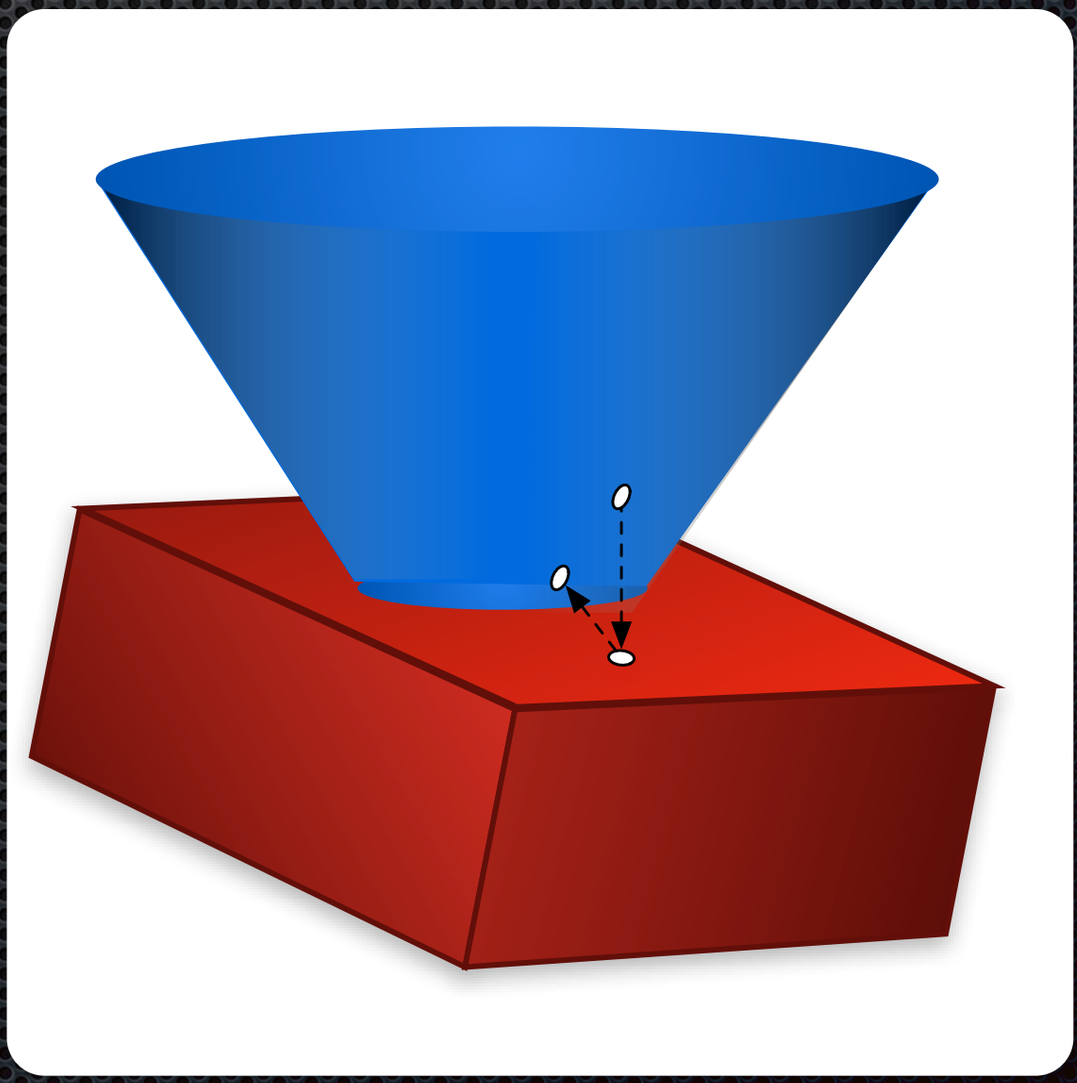


Alternating Projection

Initialize inside PSD cone

Project onto constraint - satisfying hyperplane and back

Repeat with new constraints

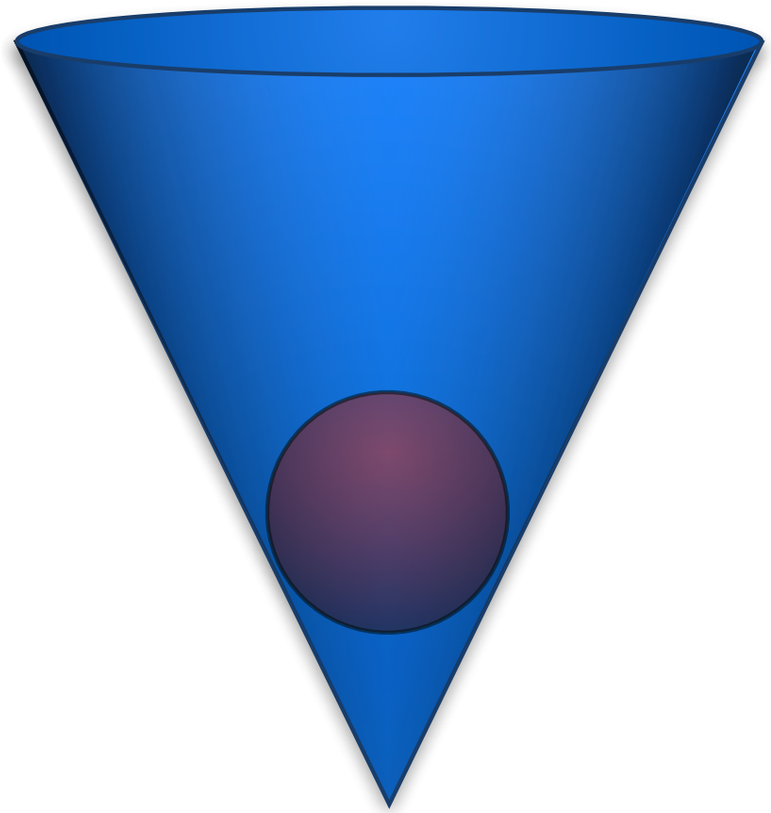


Alternating Projection

Initialize inside PSD cone

Project onto constraint - satisfying hyperplane and back

Repeat with new constraints

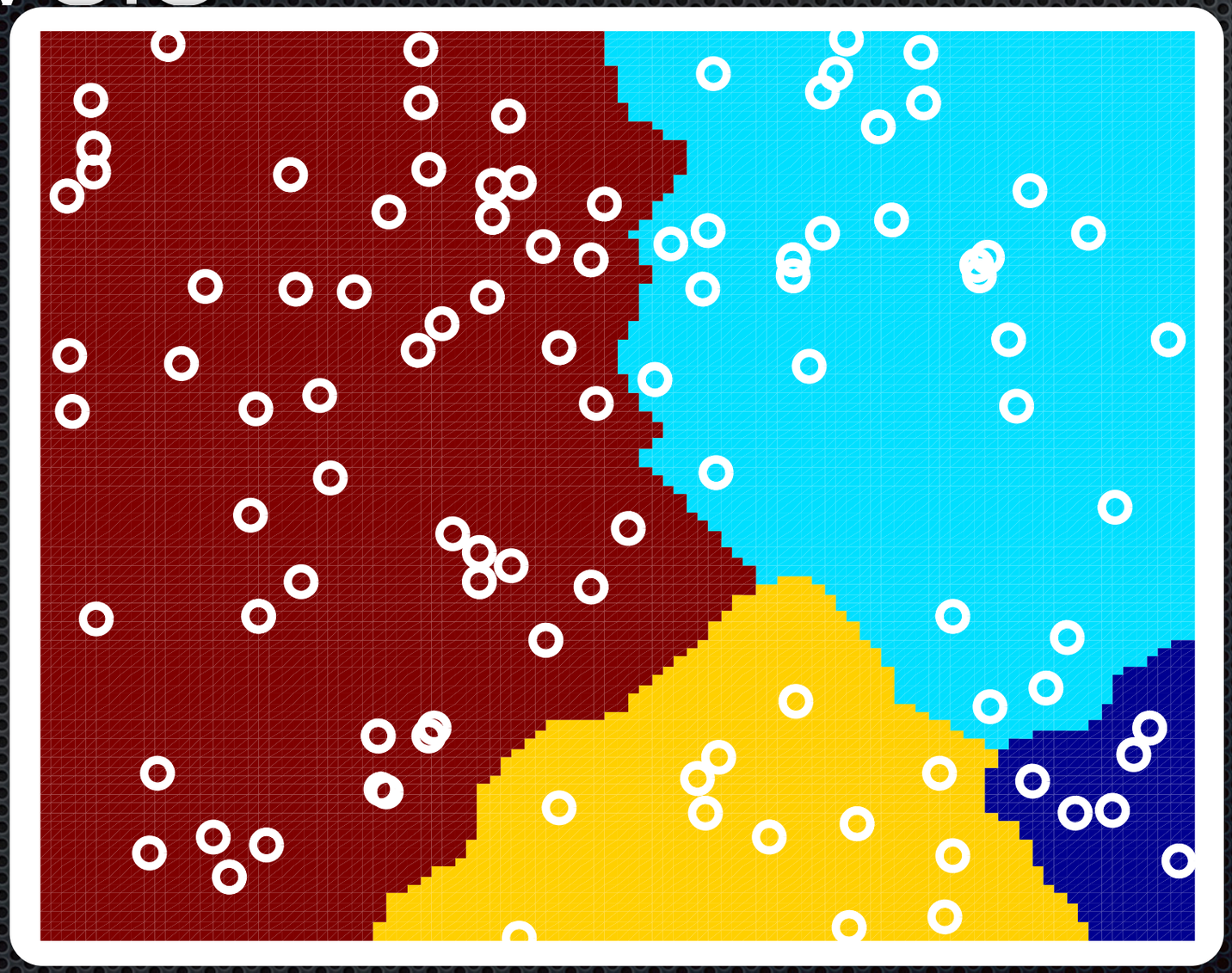


If solution exists, algorithm converges inside intersection.

Summary of POLA

- Learns Mahalanobis metric
- **Online algorithm**
- Can also be kernelized
- **Introduces a margin**
- Algorithm converges if solution exists
- Assumes data to be unimodal

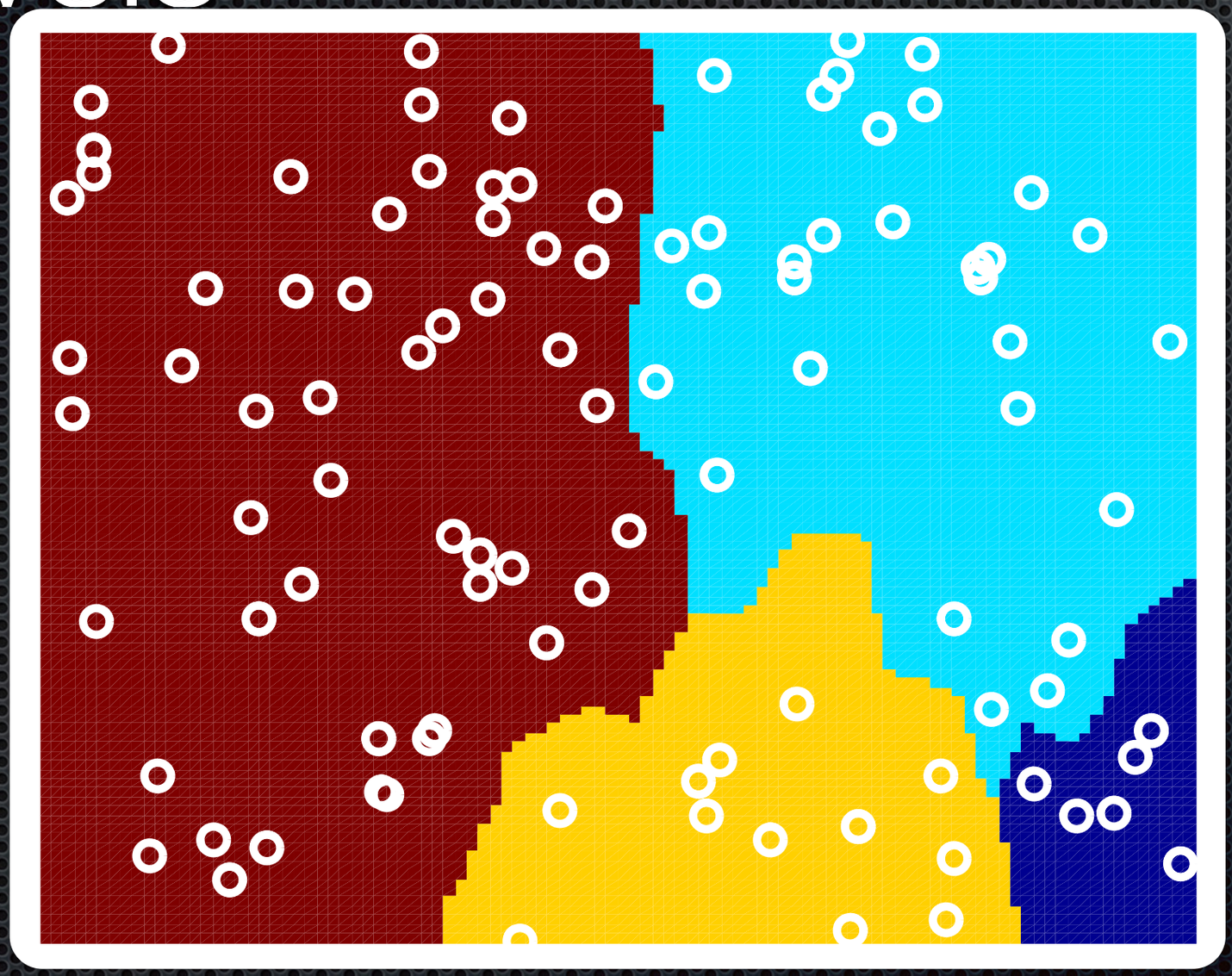
Neighborhood Component Analysis



Distance metric for visualization and kNN

[Goldberger et. al. 2004]

Neighborhood Component Analysis



Distance metric for visualization and kNN

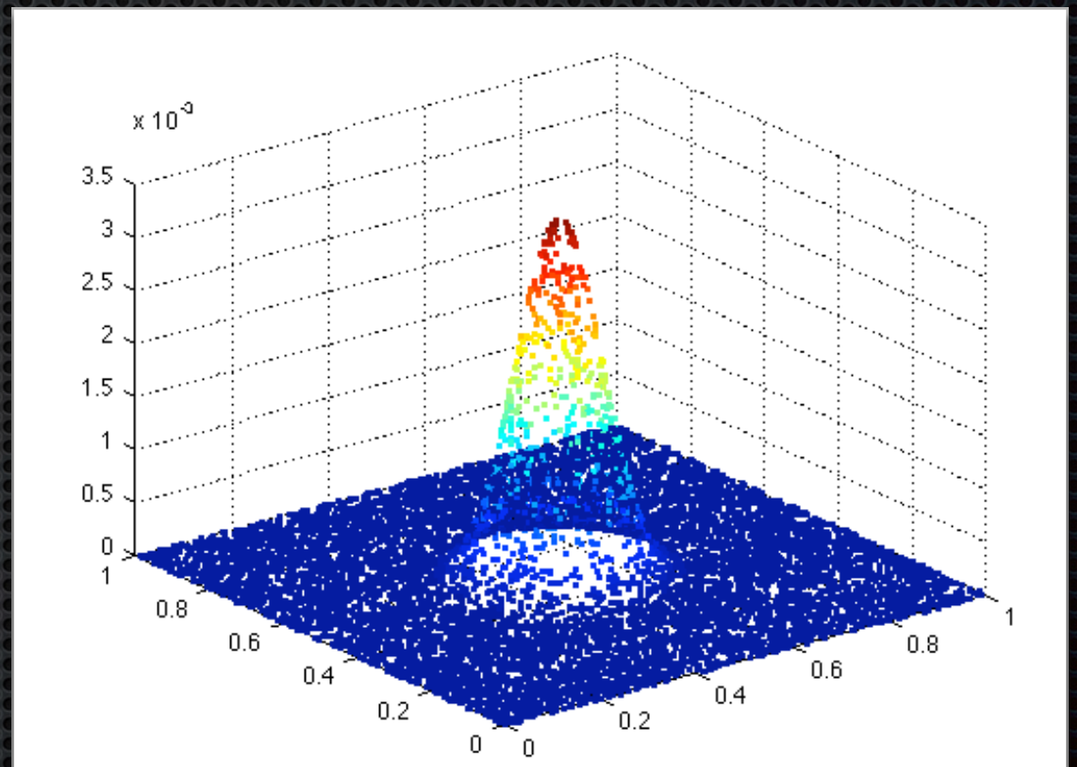
[Goldberger et. al. 2004]

Stochastic Neighborhood

- Pick neighbor randomly

$$p_{ij} = \frac{1}{Z} \exp(-\|\mathbf{A}(\vec{x}_i - \vec{x}_j)\|^2$$

Neighbor distribution



Stochastic Neighborhood

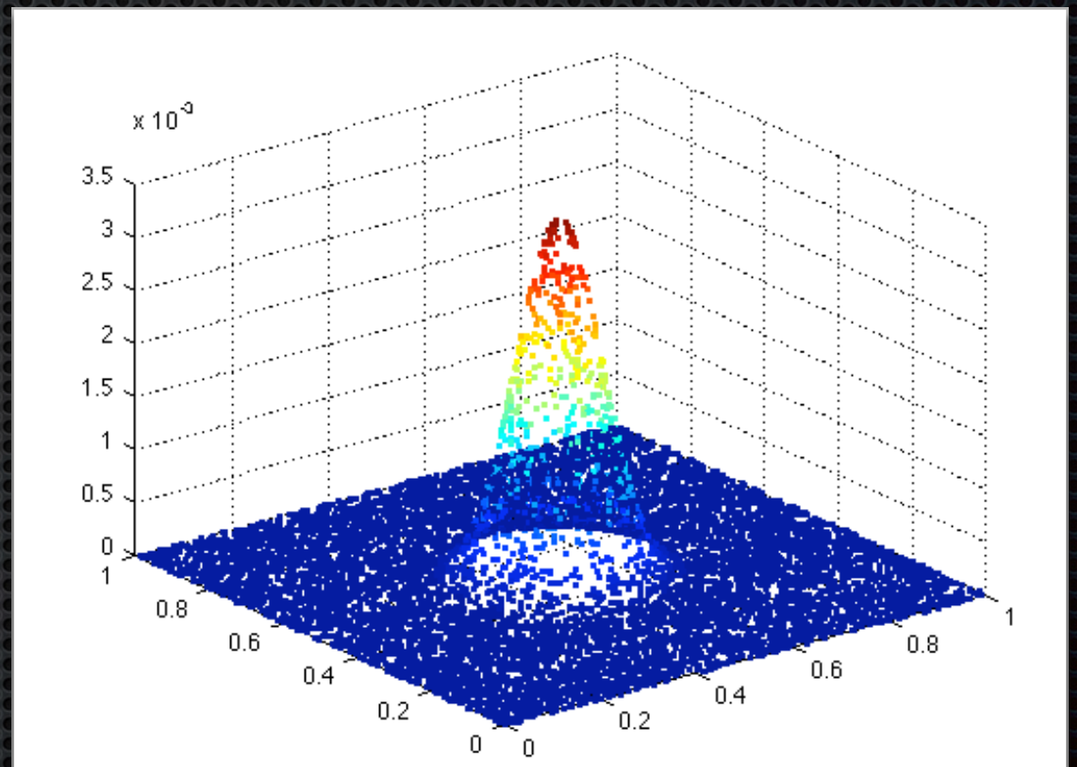
- Pick neighbor randomly

$$p_{ij} = \frac{1}{Z} \exp(-\|\mathbf{A}(\vec{x}_i - \vec{x}_j)\|^2$$

- Maximize expected number of *correct* neighbors

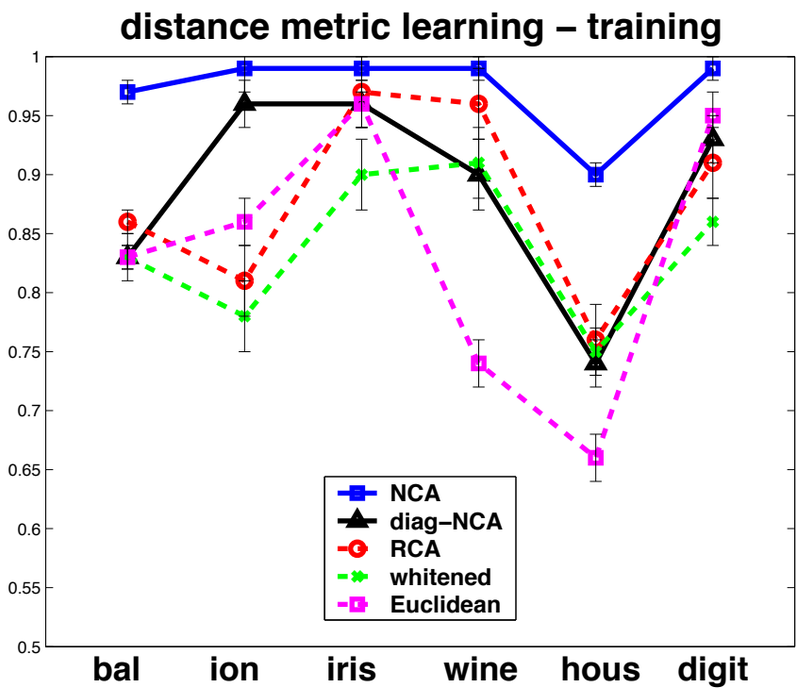
$$\max_A \sum_{(i,j) \in S} p_{ij}$$

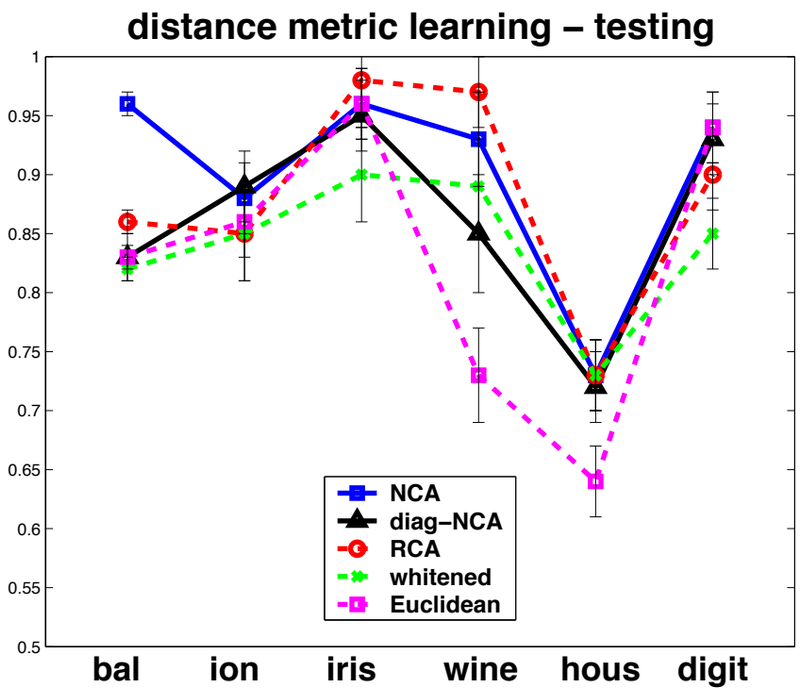
Neighbor distribution

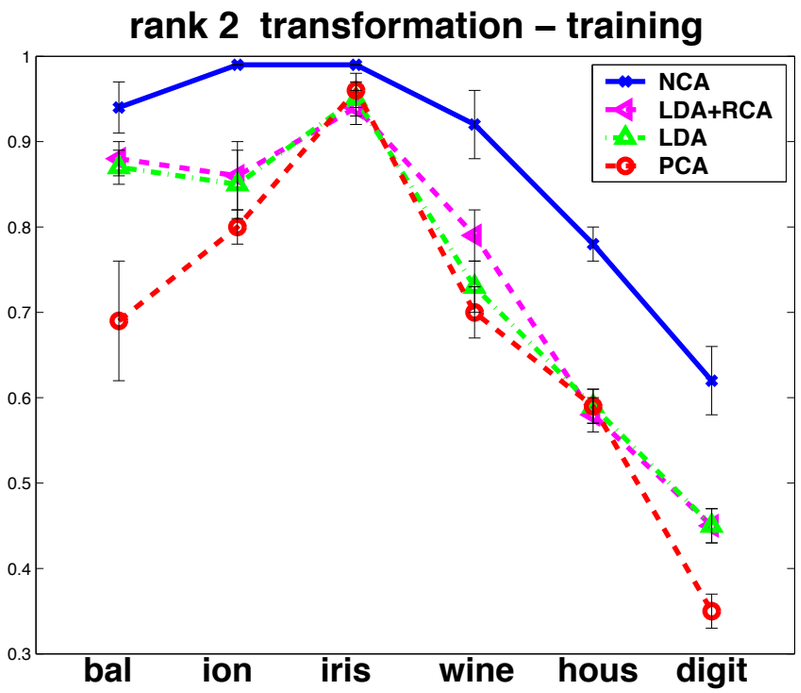


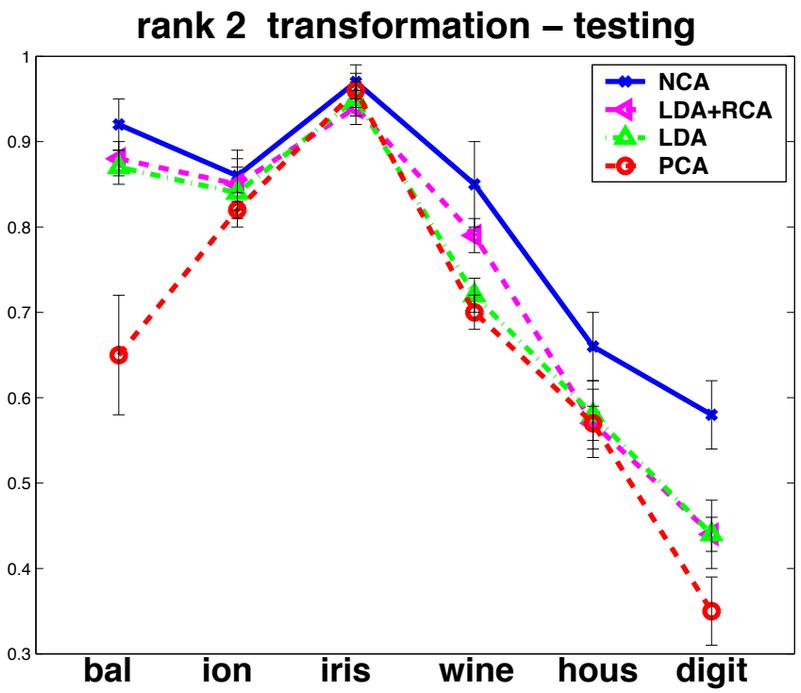
(Unconstraint optimization problem - non-convex)

Results


distance metric learning – training
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
NCA
diag–NCA
RCA
whitened
Euclidean
bal
ion
iris
wine
hous
digit

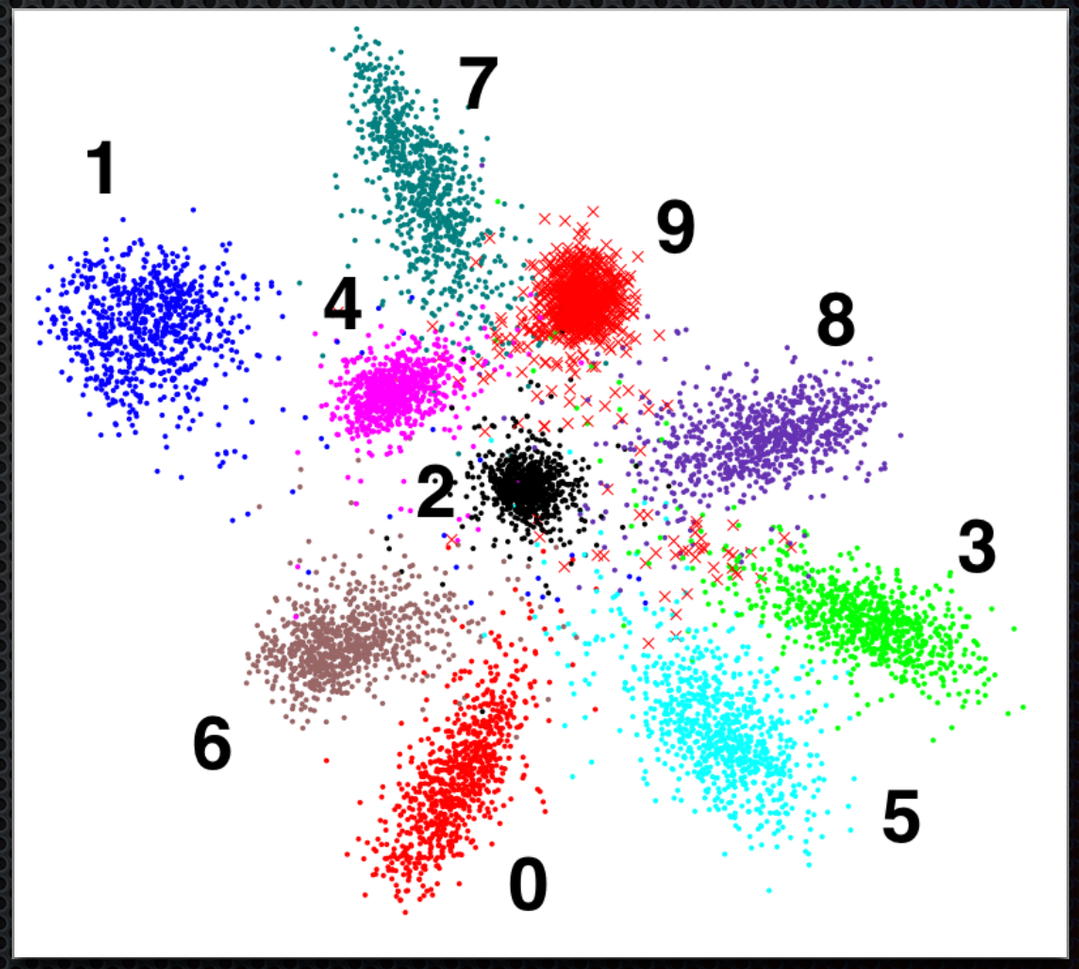

distance metric learning – testing
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
NCA
diag–NCA
RCA
whitened
Euclidean
bal
ion
iris
wine
hous
digit


rank 2 transformation – training
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
NCA
LDA+RCA
LDA
PCA
bal
ion
iris
wine
hous
digit


rank 2 transformation – testing
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
NCA
LDA+RCA
LDA
PCA
bal
ion
iris
wine
hous
digit

NCA-Extensions

- [Globerson and Roweis; NIPS 2005]
  - **Convex** loss function
- [Weinberger and Tesauro; AISTATS 2007]
  - Kernel **regression**
- [Salakhutdinov and Hinton; AISTATS 2007]
  - **Non-linear** mapping (DBN)



MNIST handwritten digits embedding - (Courtesy of Salakhutdinov and Hinton)

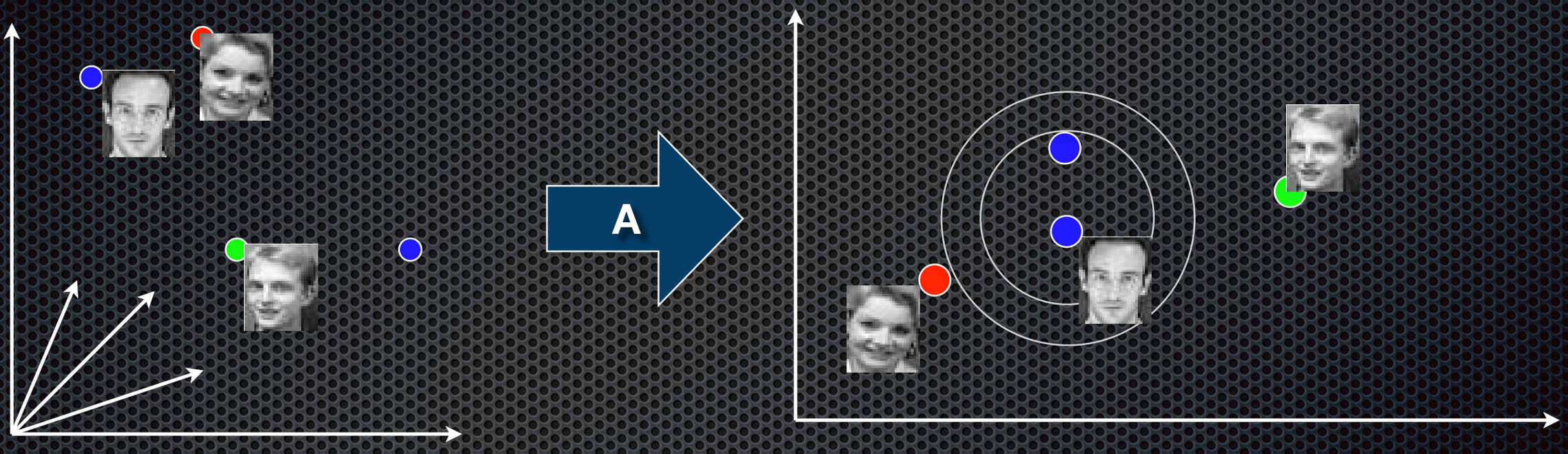
Summary of NCA

- Mahalanobis metric for k-NN
- Also, indirectly, learns parameter k
- Can be used to reduce dimensionality
- NCA optimization is unconstrained
- **NCA can handle multi-modal data** (not MCML)
- NCA objective is not convex
- Normalization becomes expensive for large data

Large Margin Nearest Neighbor

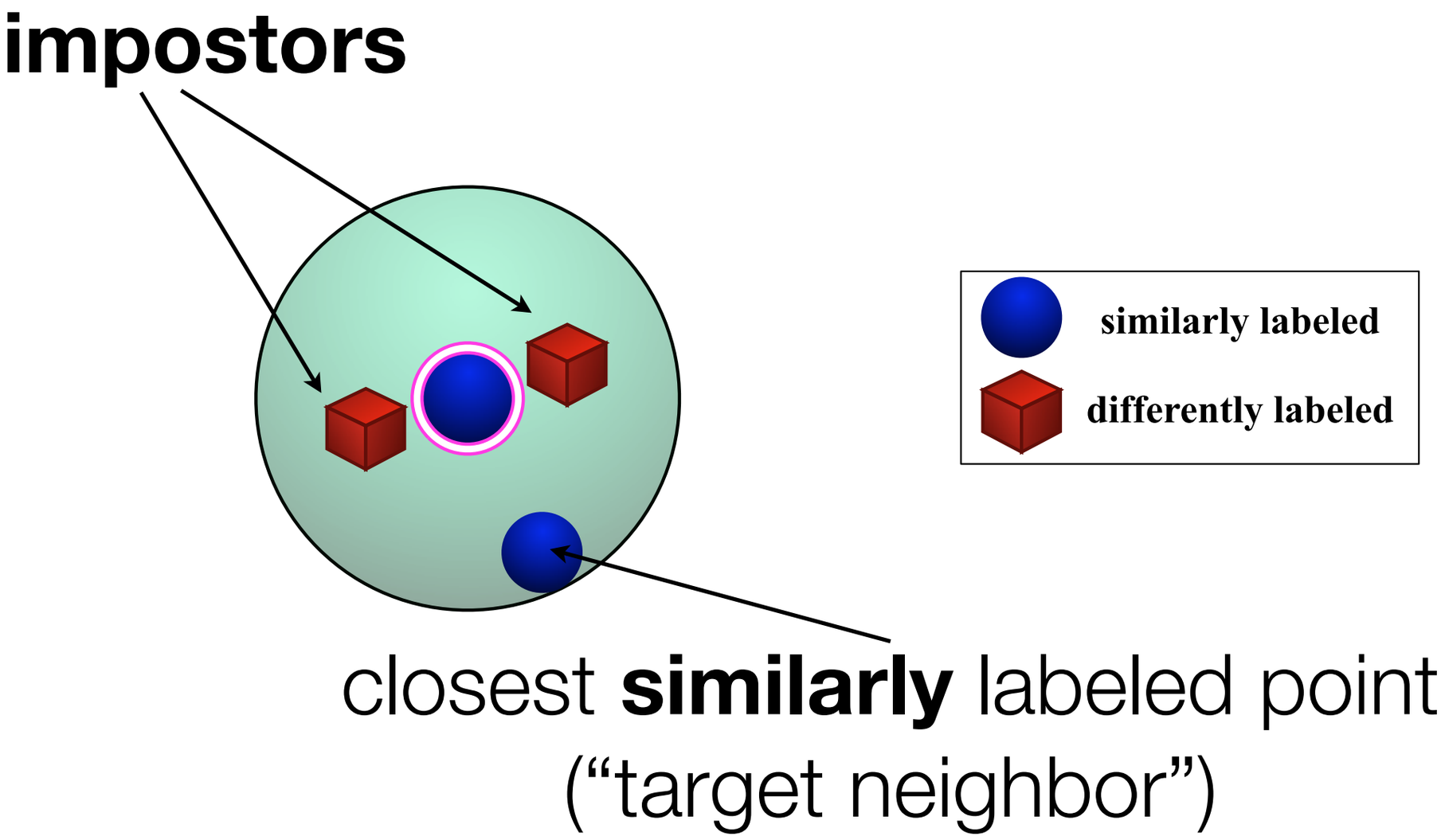
[Weinberger et al.; NIPS 2006]

Large Margin Nearest Neighbor

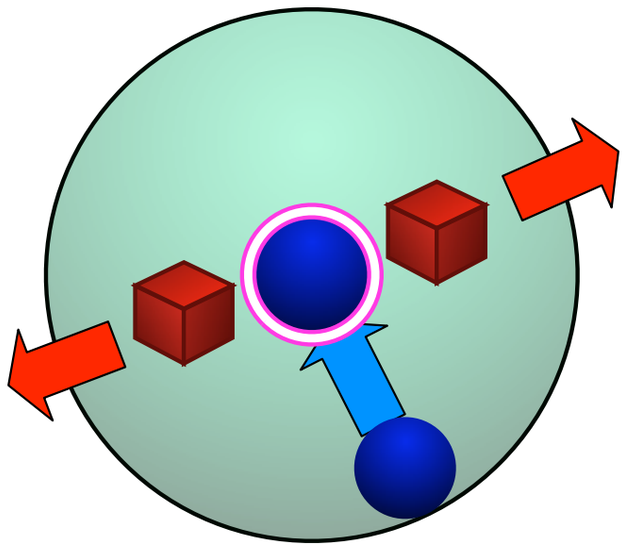


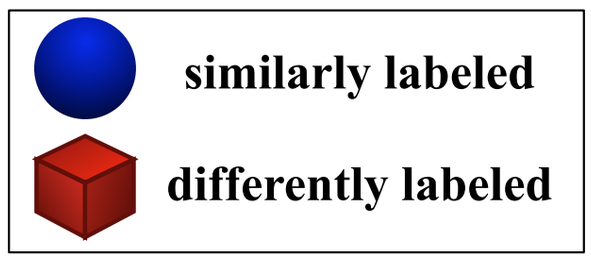
Intuition: Each data point should be closer to the **closest similar** point than to **any different one**.

Intuition


impostors
similarly labeled
differently labeled
closest **similarly** labeled point
("target neighbor")

Intuition

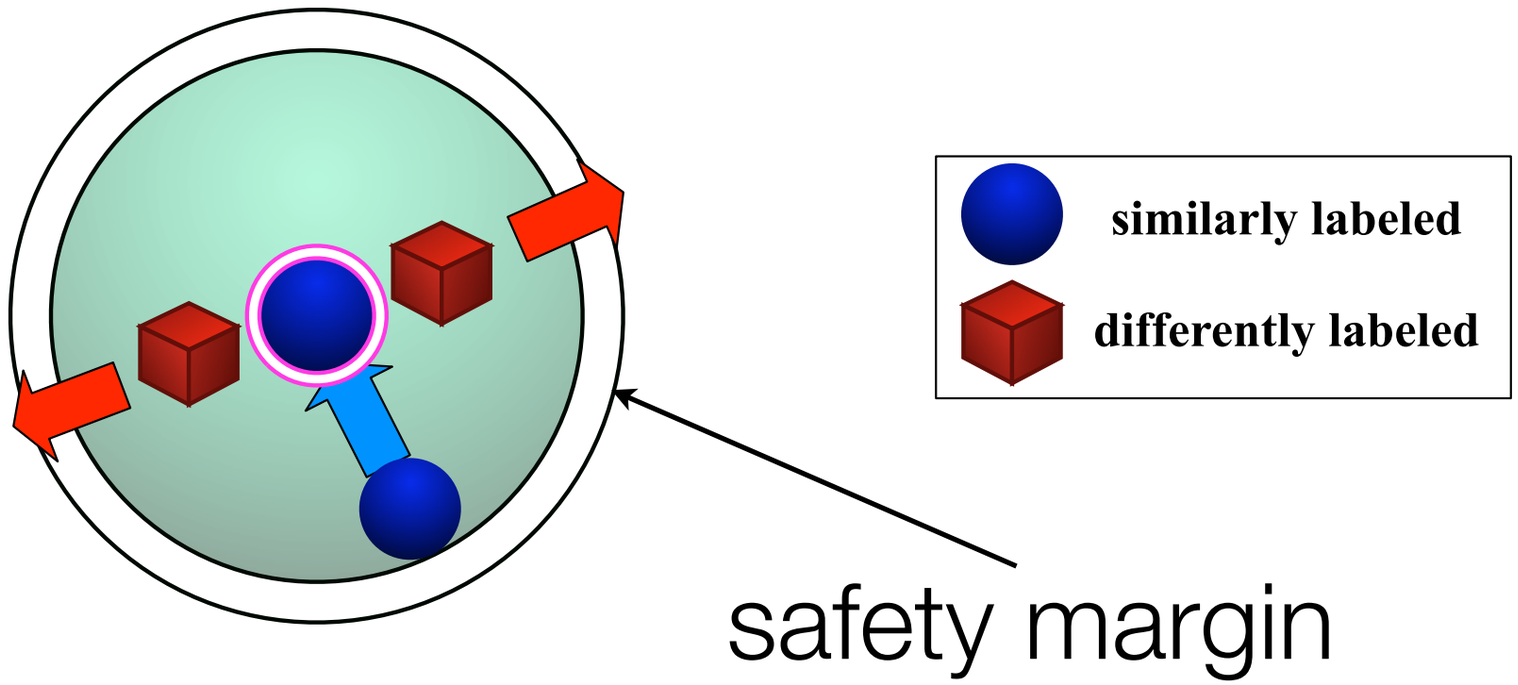




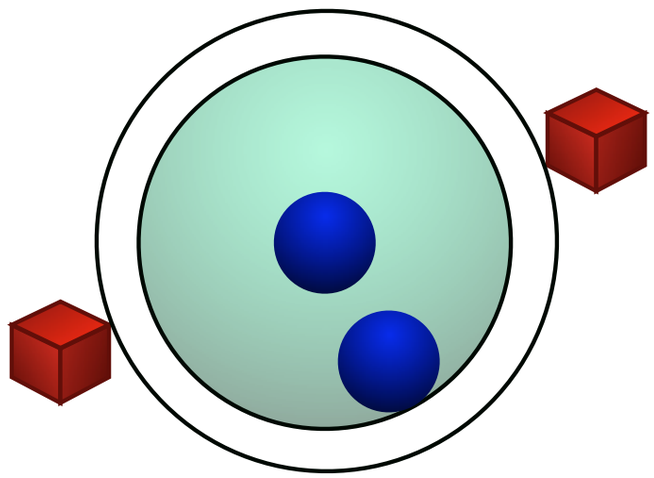
1. pull target neighbors closer

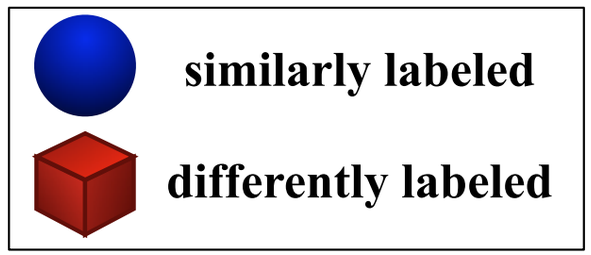
2. push impostors away

Intuition



Clean neighborhood!





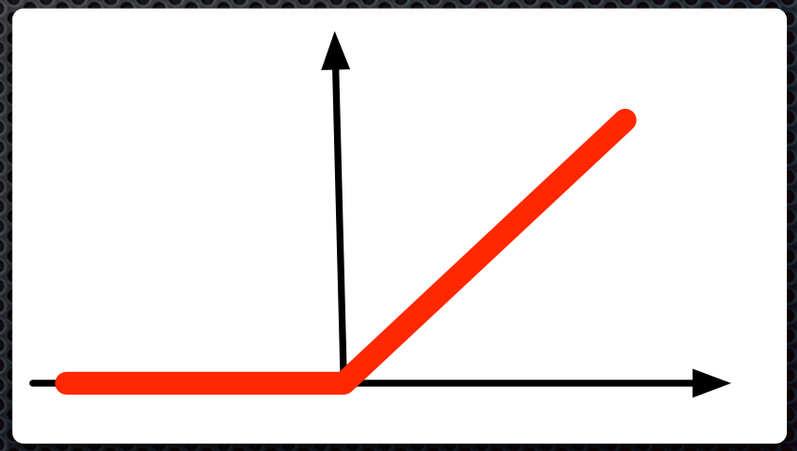
Notation

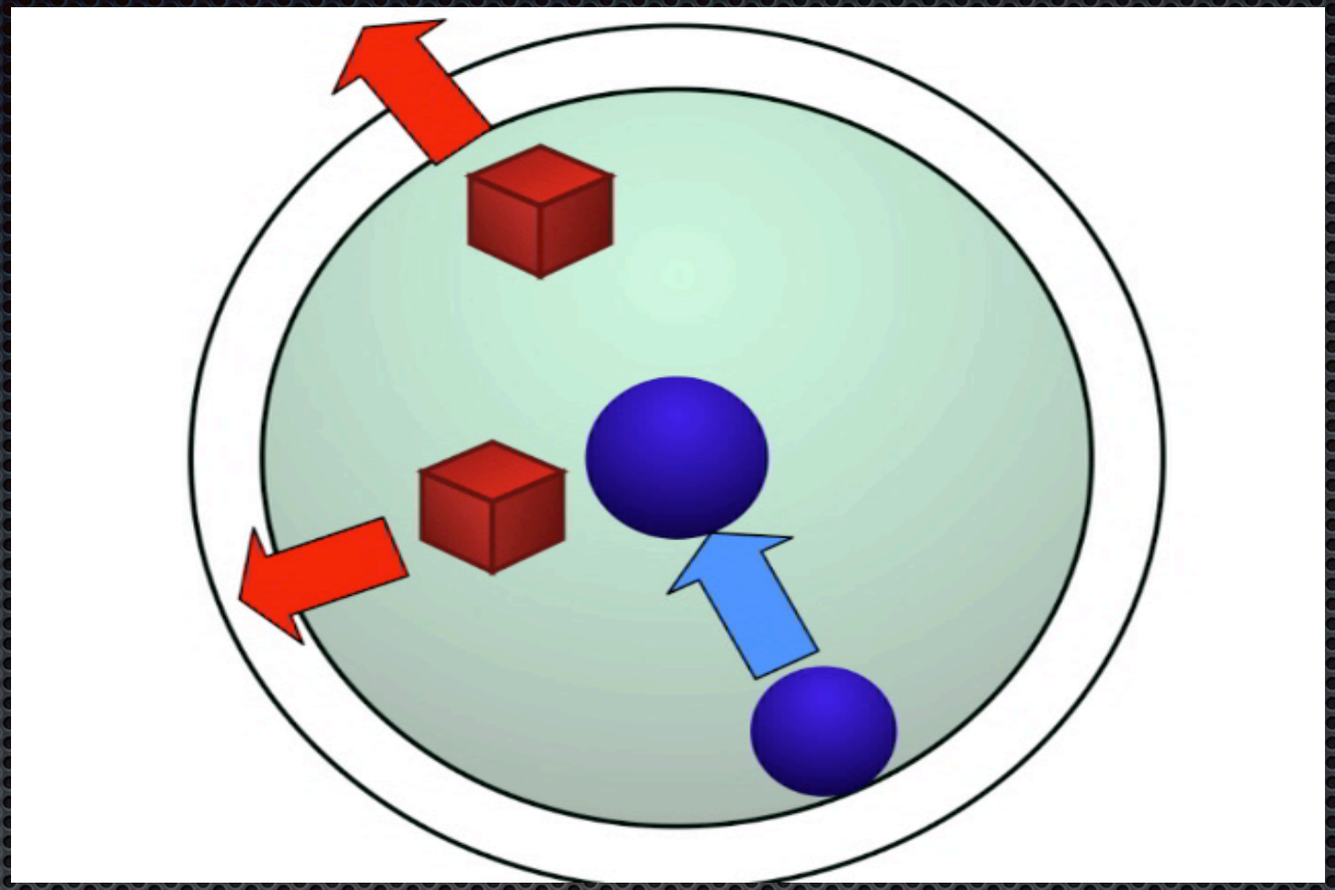
$$D = \{(i,l) \mid \vec{x}_i, \vec{x}_l \text{ are } \textit{dissimilar}\}$$

$$j \rightsquigarrow i : \quad \vec{x}_j \text{ is a } \textit{target neighbor} \text{ of } \vec{x}_i$$

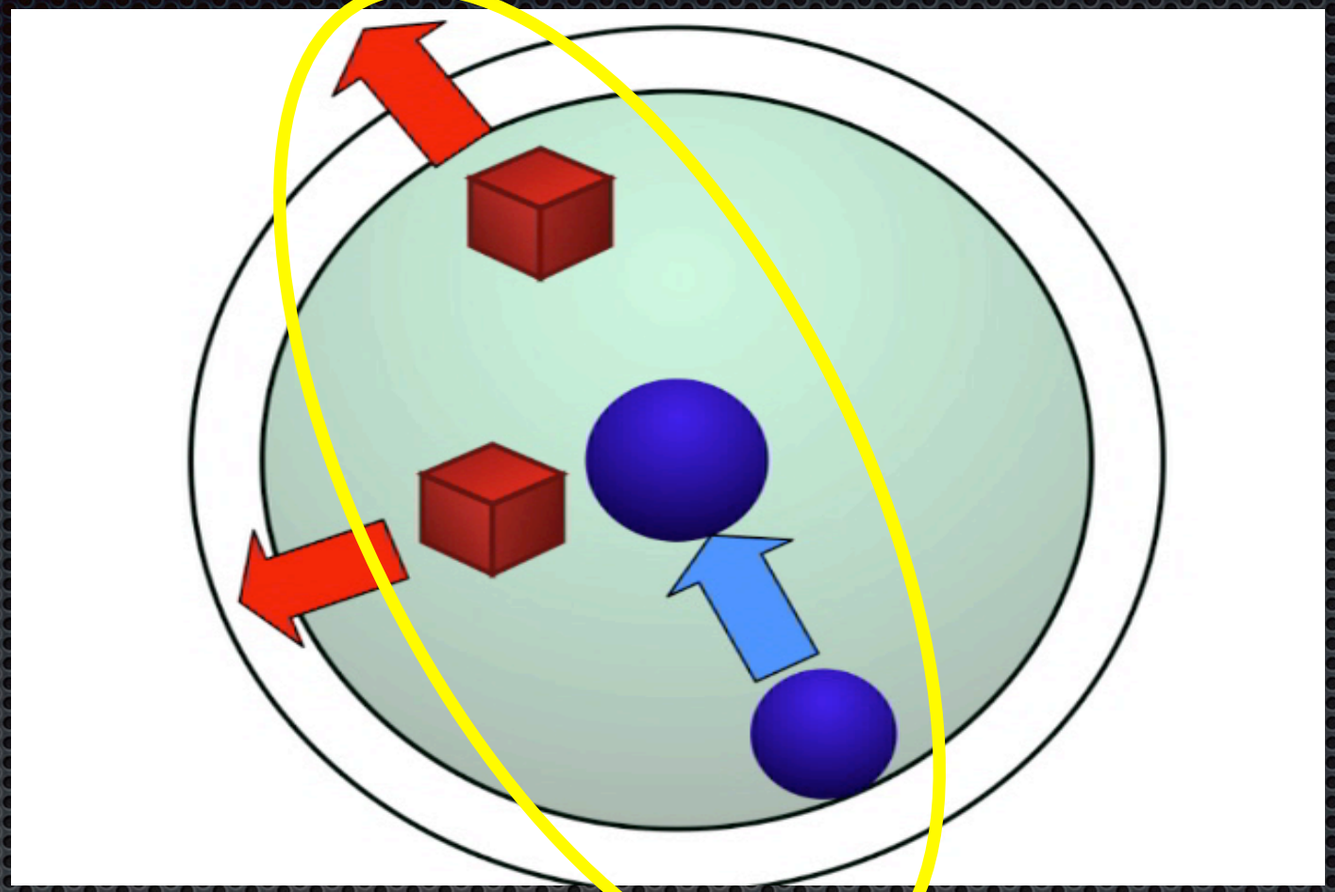
$$[a]_+ = max(a, 0)$$

(hinge-loss)



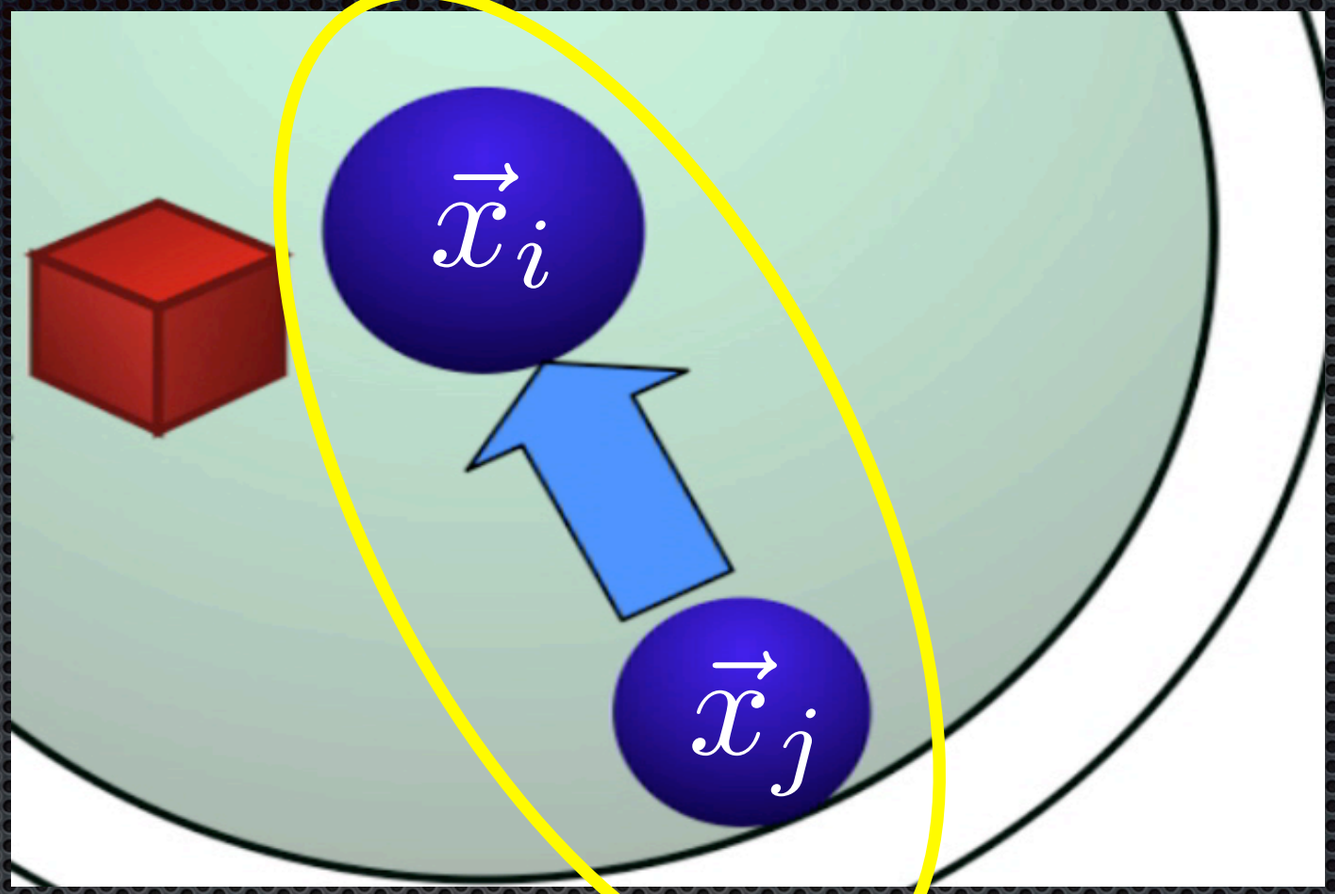


$$\min_{\mathbf{L}} \quad \mathcal{L}(\mathbf{L}) = \mathcal{L}_{pull}(\mathbf{L}) + \mathcal{L}_{push}(\mathbf{L})$$



$$\min_{\mathbf{L}} \quad \mathcal{L}(\mathbf{L}) = \mathcal{L}_{pull}(\mathbf{L}) + \mathcal{L}_{push}(\mathbf{L})$$

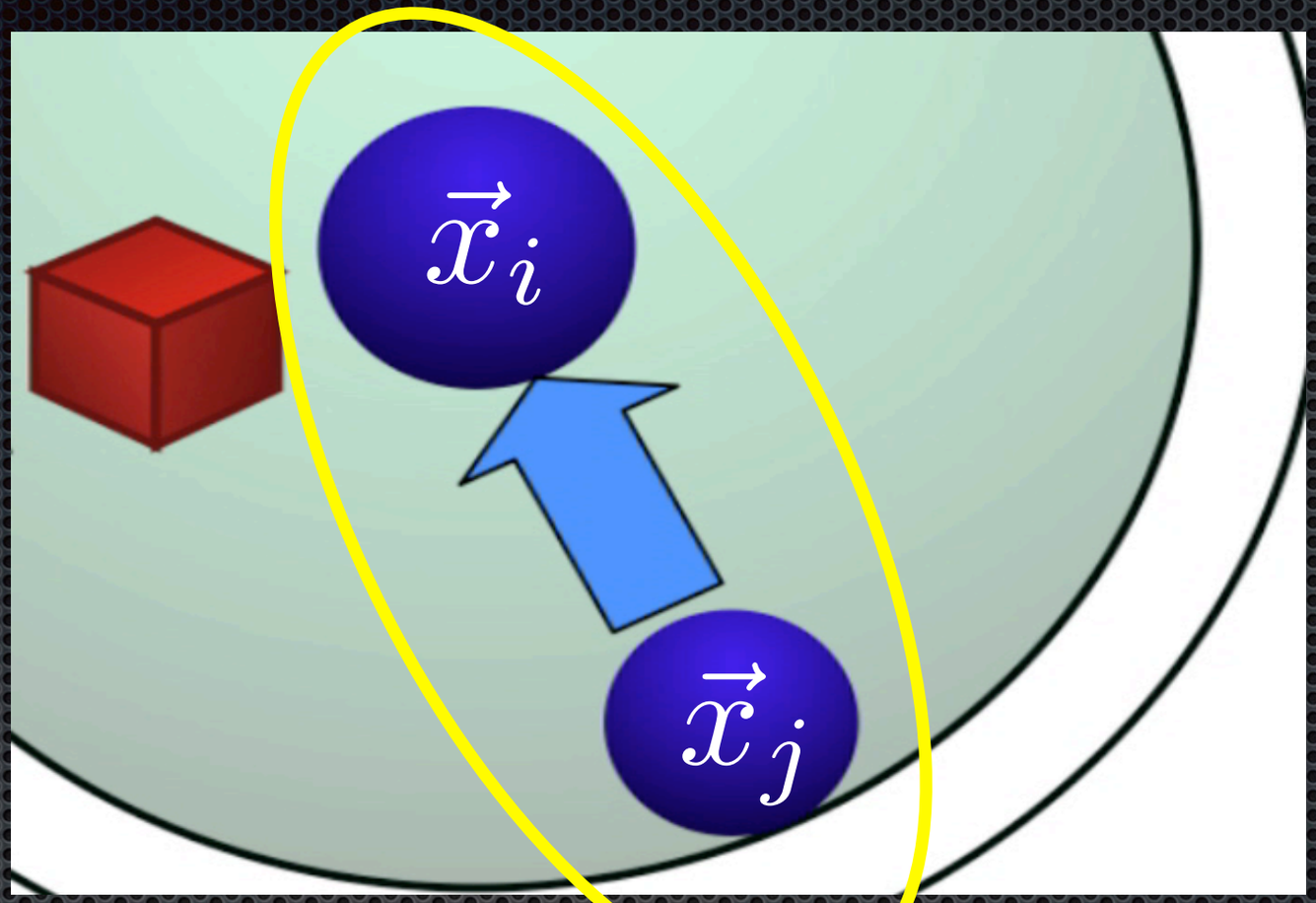
$$\mathcal{L}_{pull}(\mathbf{L}) = \sum_{j \rightsquigarrow i} \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|_2^2$$



$$\min_{\mathbf{L}} \quad \mathcal{L}(\mathbf{L}) = \mathcal{L}_{pull}(\mathbf{L}) + \mathcal{L}_{push}(\mathbf{L})$$

$$\mathcal{L}_{pull}(\mathbf{L}) = \sum_{j \rightsquigarrow i} \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|_2^2$$

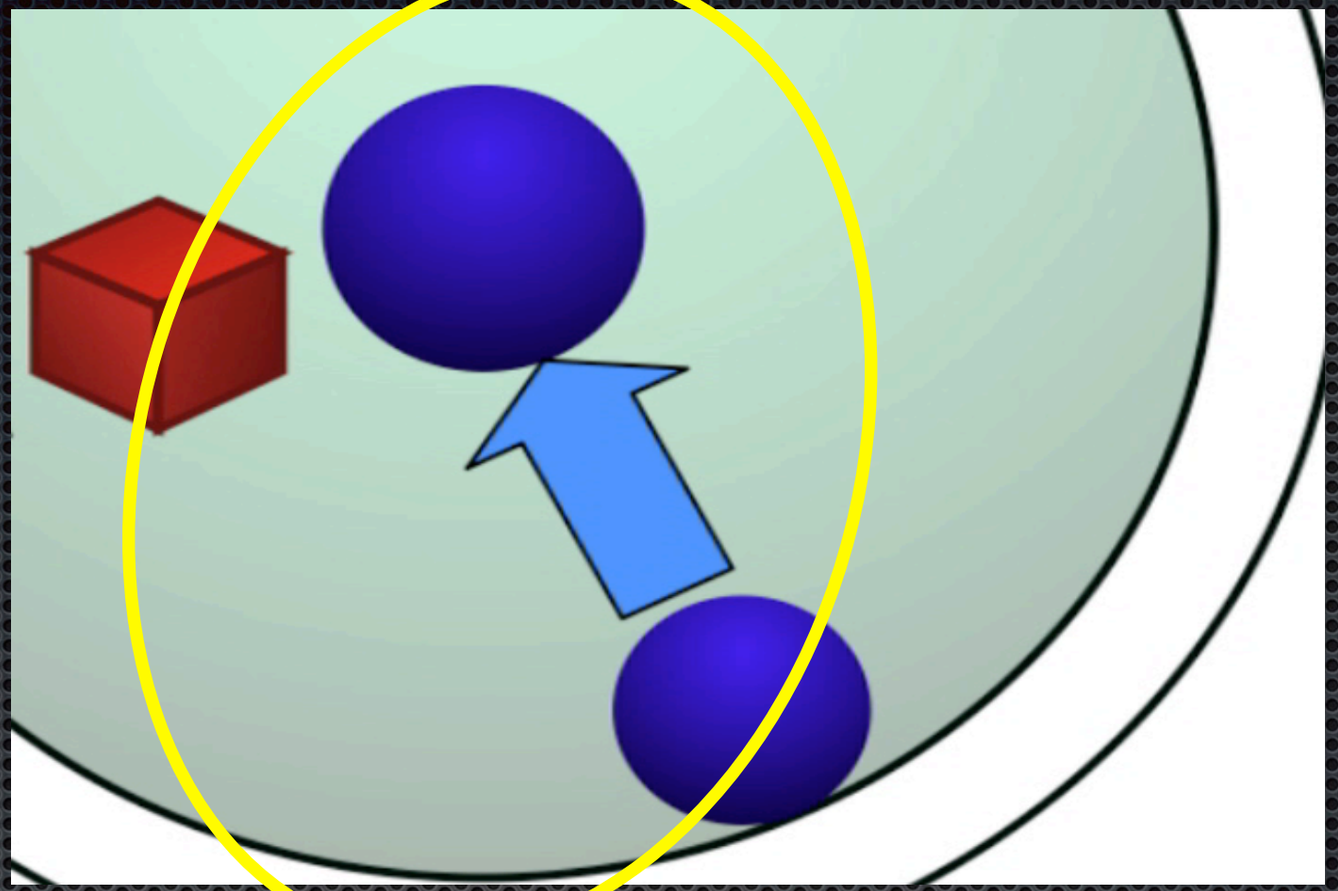
Target neighbors



$$\min_{\mathbf{L}} \quad \mathcal{L}(\mathbf{L}) = \mathcal{L}_{pull}(\mathbf{L}) + \mathcal{L}_{push}(\mathbf{L})$$

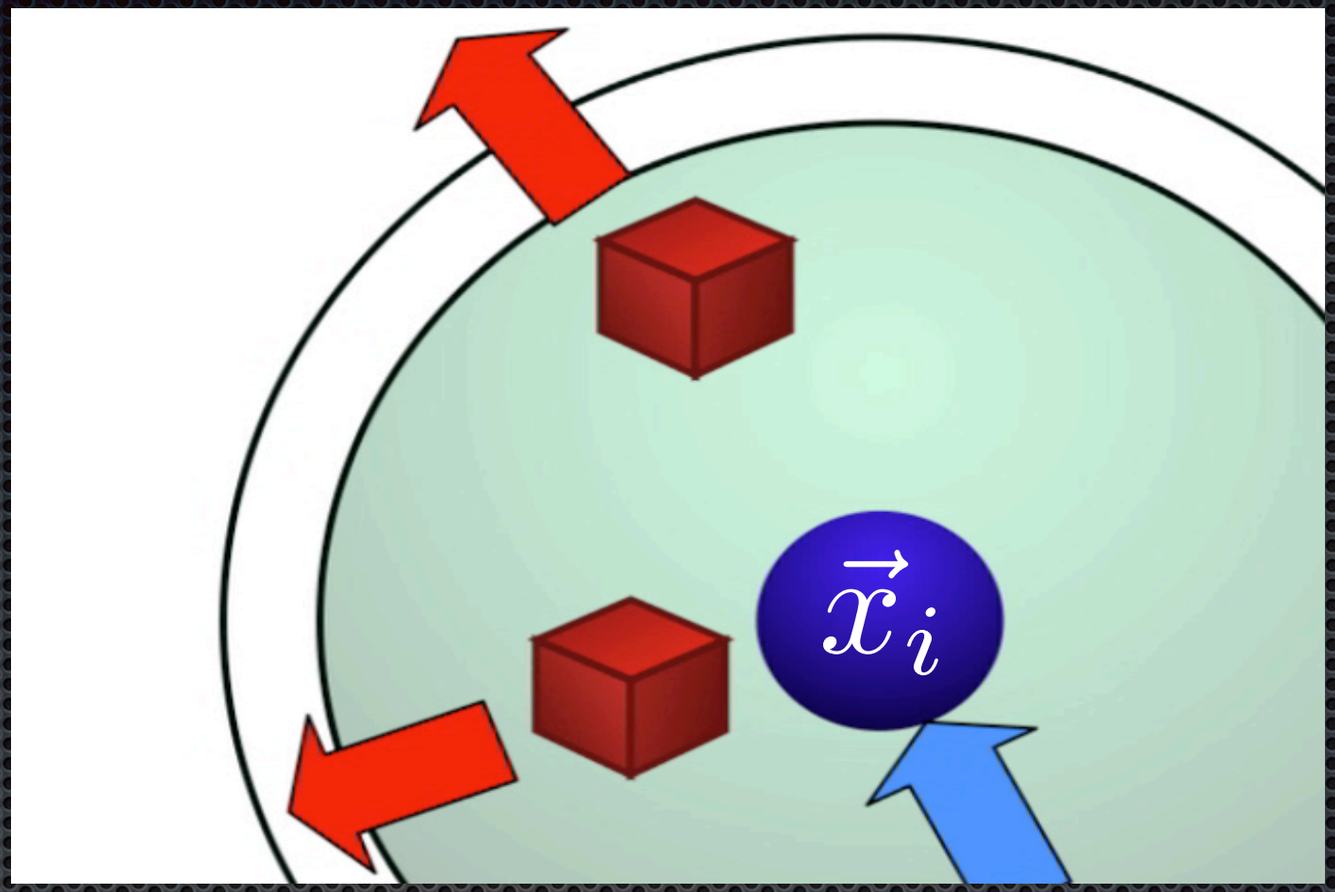
$$\mathcal{L}_{pull}(\mathbf{L}) = \sum_{j \rightsquigarrow i} \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|_2^2$$

Distance to target neighbor



$$\min_{\mathbf{L}} \quad \mathcal{L}(\mathbf{L}) = \mathcal{L}_{pull}(\mathbf{L}) + \mathcal{L}_{push}(\mathbf{L})$$

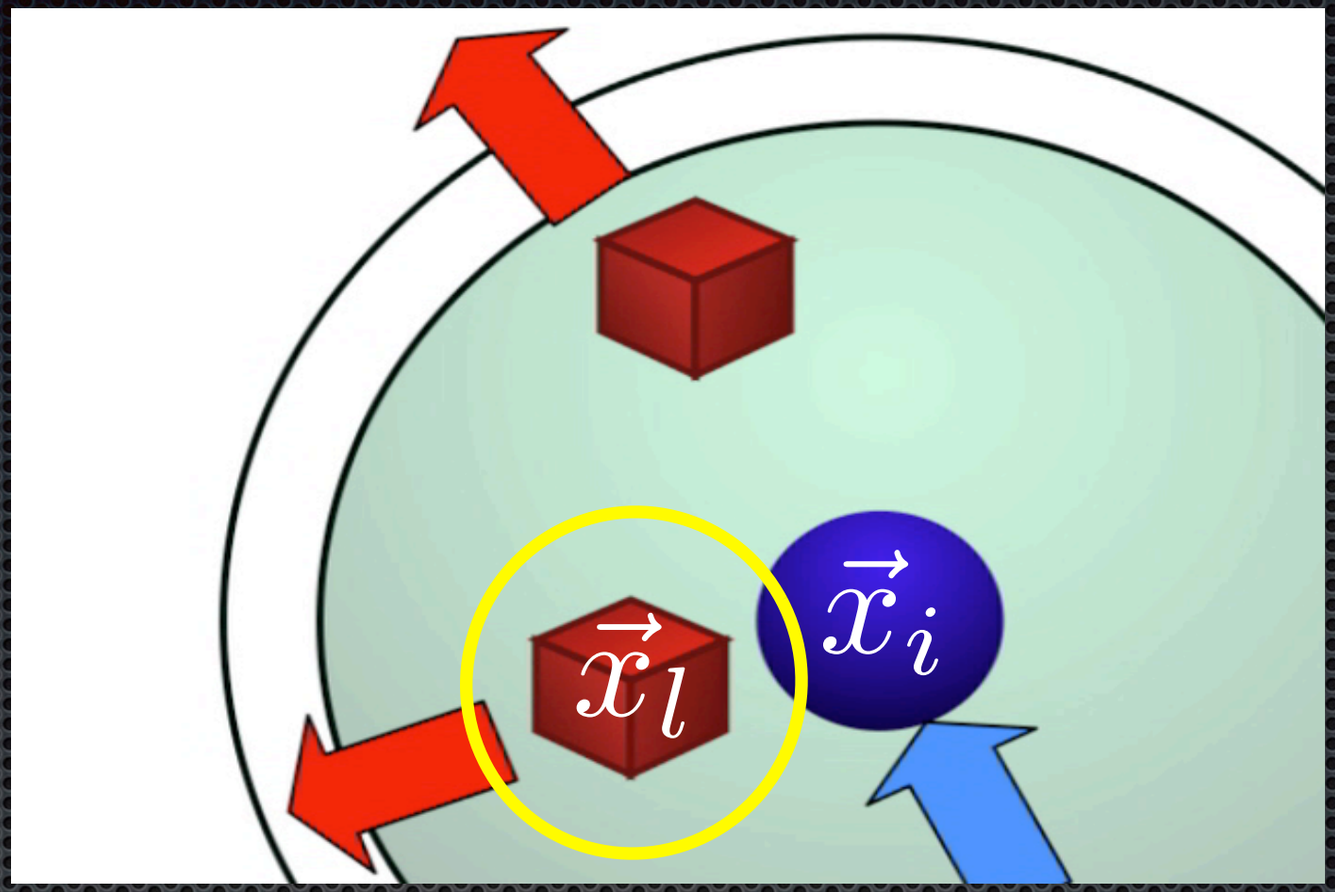
$$\mathcal{L}_{push}(\mathbf{L}) = \sum_{j \rightsquigarrow i} \sum_{(i,l) \in D} [1 + \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|_2^2 - \|\mathbf{L}(\vec{x}_i - \vec{x}_l)\|_2^2]_+$$



$$\min_{\mathbf{L}} \quad \mathcal{L}(\mathbf{L}) = \mathcal{L}_{pull}(\mathbf{L}) + \mathcal{L}_{push}(\mathbf{L})$$

$$\mathcal{L}_{push}(\mathbf{L}) = \sum_{j \rightsquigarrow i} \sum_{(i,l) \in D} [1 + \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|_2^2 - \|\mathbf{L}(\vec{x}_i - \vec{x}_l)\|_2^2]_+$$

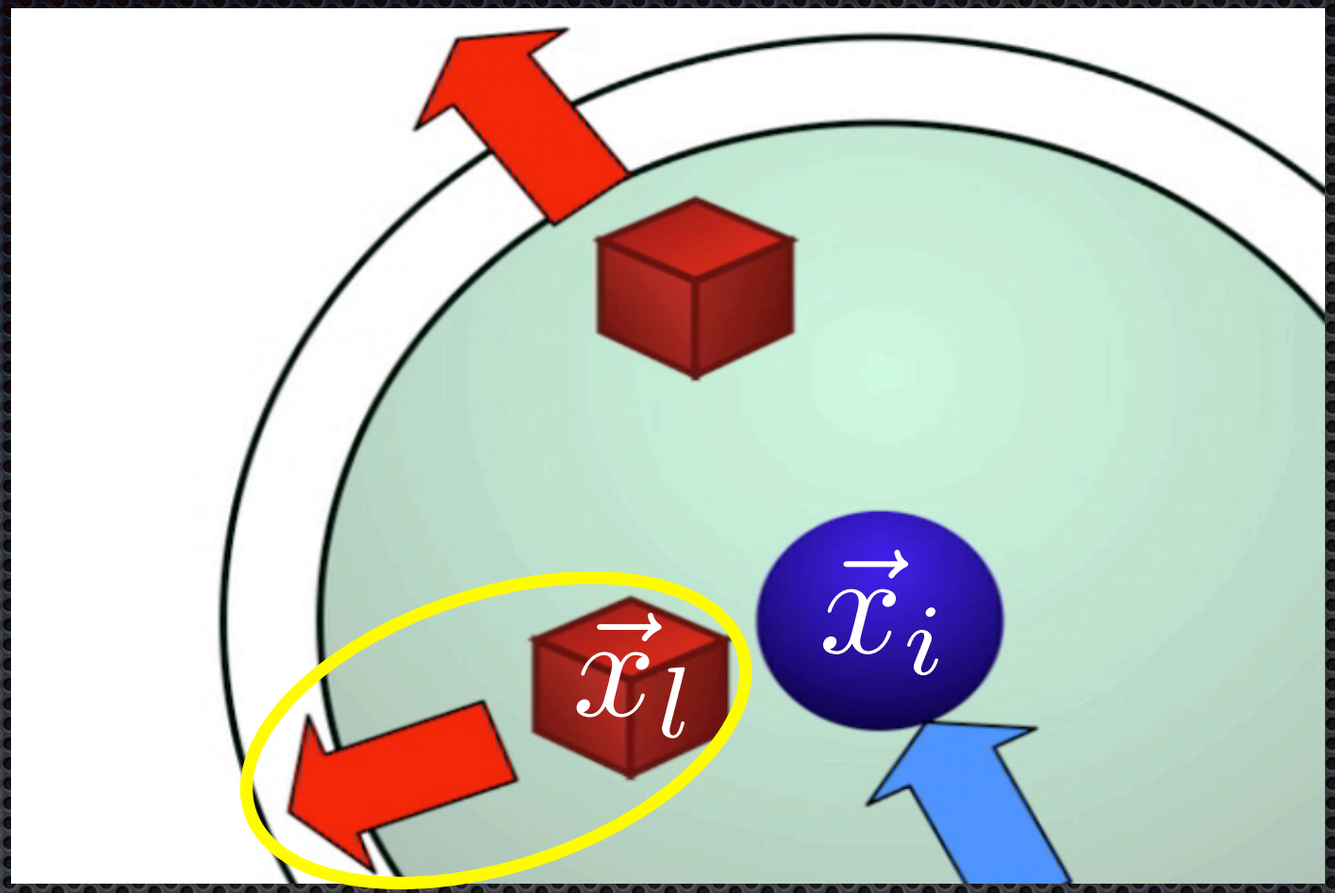
target neighbor



$$\min_{\mathbf{L}} \quad \mathcal{L}(\mathbf{L}) = \mathcal{L}_{pull}(\mathbf{L}) + \mathcal{L}_{push}(\mathbf{L})$$

$$\mathcal{L}_{push}(\mathbf{L}) = \sum_{j \rightsquigarrow i} \sum_{(i,l) \in D} [1 + \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|_2^2 - \|\mathbf{L}(\vec{x}_i - \vec{x}_l)\|_2^2]_+$$

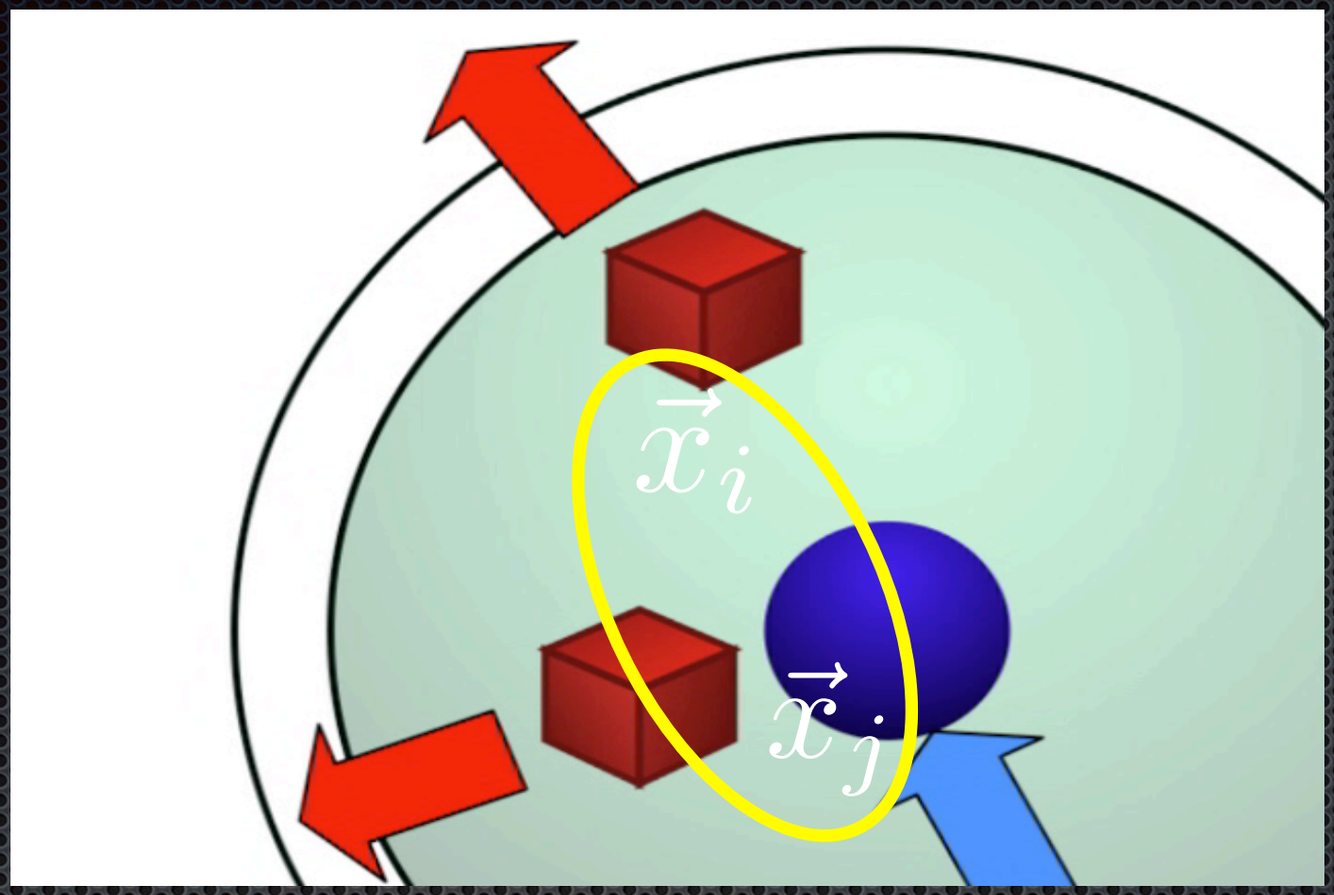
impostor



$$\min_{\mathbf{L}} \quad \mathcal{L}(\mathbf{L}) = \mathcal{L}_{pull}(\mathbf{L}) + \mathcal{L}_{push}(\mathbf{L})$$

$$\mathcal{L}_{push}(\mathbf{L}) = \sum_{j \rightsquigarrow i} \sum_{(i,l) \in D} [1 + \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|_2^2 - \|\mathbf{L}(\vec{x}_i - \vec{x}_l)\|_2^2]_+$$

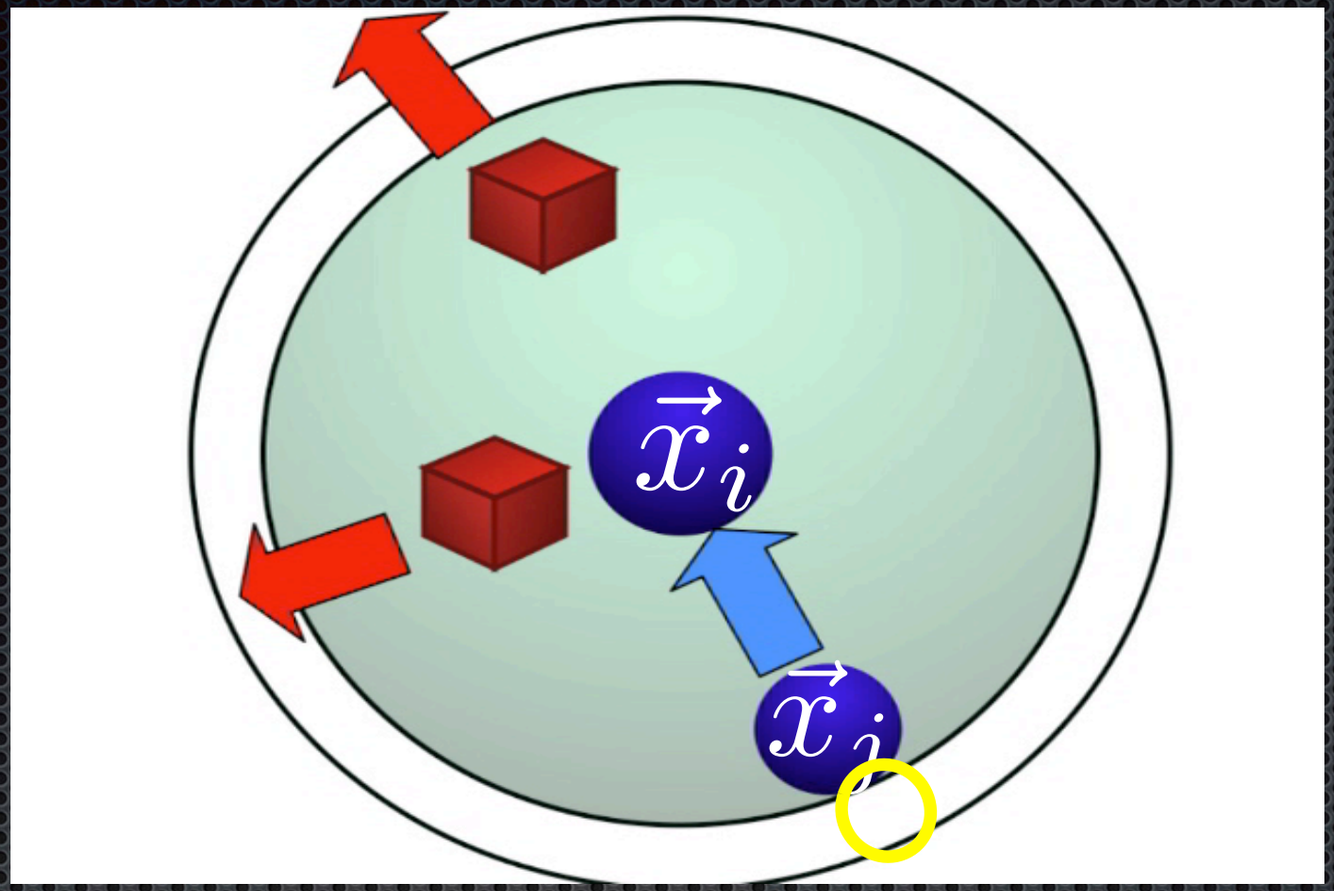
hinge loss



$$\min_{\mathbf{L}} \quad \mathcal{L}(\mathbf{L}) = \mathcal{L}_{pull}(\mathbf{L}) + \mathcal{L}_{push}(\mathbf{L})$$

$$\mathcal{L}_{push}(\mathbf{L}) = \sum_{j \rightsquigarrow i} \sum_{(i,l) \in D} [1 + \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|_2^2 - \|\mathbf{L}(\vec{x}_i - \vec{x}_l)\|_2^2]_+$$

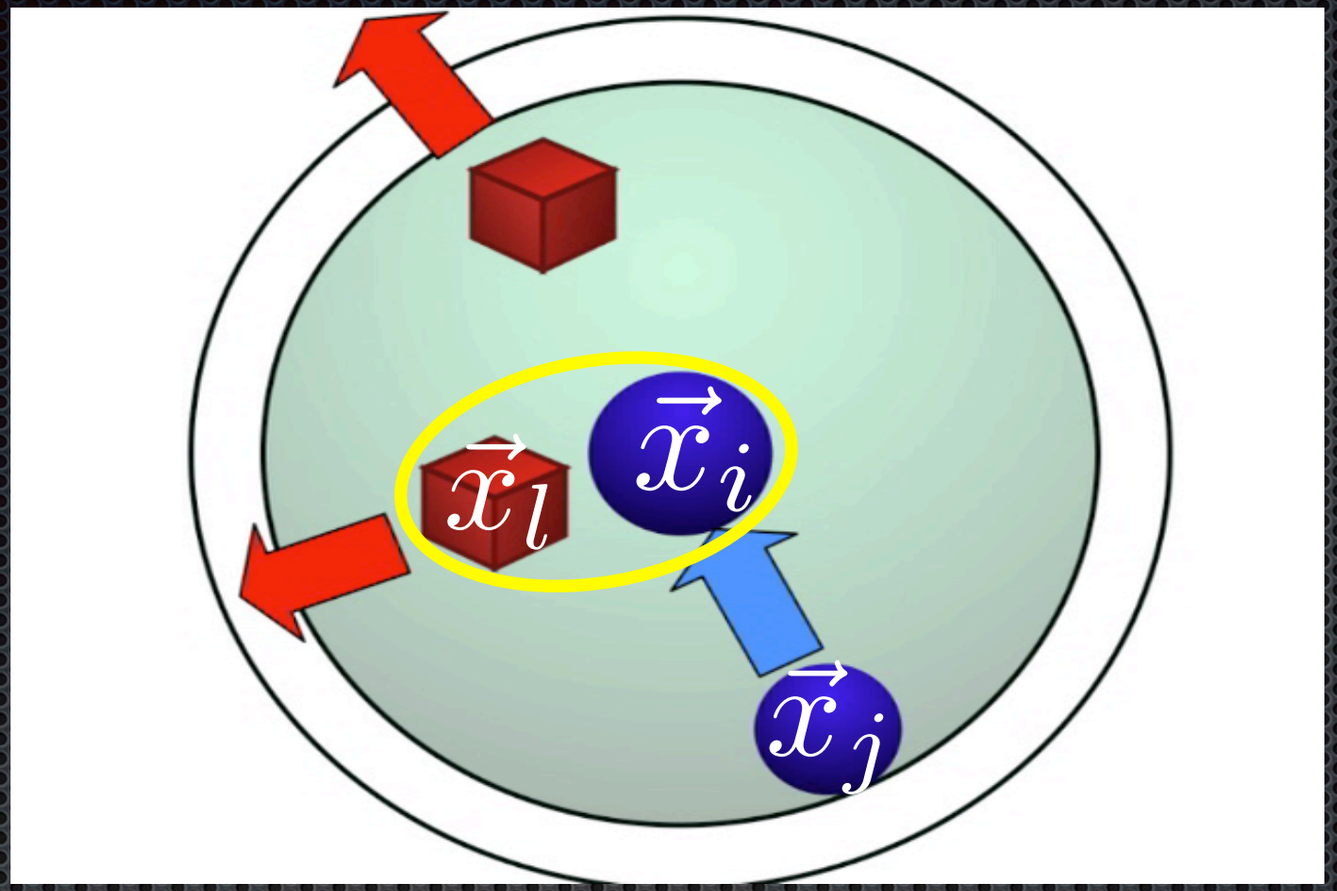
neighborhood radius



$$\min_{\mathbf{L}} \quad \mathcal{L}(\mathbf{L}) = \mathcal{L}_{pull}(\mathbf{L}) + \mathcal{L}_{push}(\mathbf{L})$$

$$\mathcal{L}_{push}(\mathbf{L}) = \sum_{j \rightsquigarrow i} \sum_{(i,l) \in D} [1 + \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|_2^2 - \|\mathbf{L}(\vec{x}_i - \vec{x}_l)\|_2^2]_+$$

safety margin

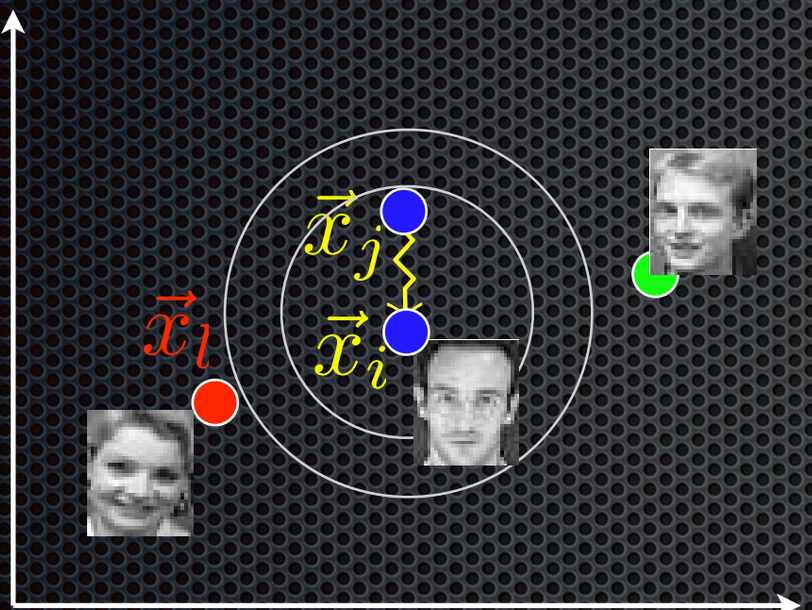


$$\min_{\mathbf{L}} \quad \mathcal{L}(\mathbf{L}) = \mathcal{L}_{pull}(\mathbf{L}) + \mathcal{L}_{push}(\mathbf{L})$$

$$\mathcal{L}_{push}(\mathbf{L}) = \sum_{j \rightsquigarrow i} \sum_{(i,l) \in D} [1 + \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|_2^2 - \|\mathbf{L}(\vec{x}_i - \vec{x}_l)\|_2^2]_+$$

distance to impostor

Large margin nearest neighbor (LMNN)



Semi-definite program

$$\min_{\mathbf{M}} \sum_{i,j \rightsquigarrow i} \|\vec{x}_i - \vec{x}_j\|^2_{\mathbf{M}} + \mu \sum_{i,j,l} \xi_{ijl}$$

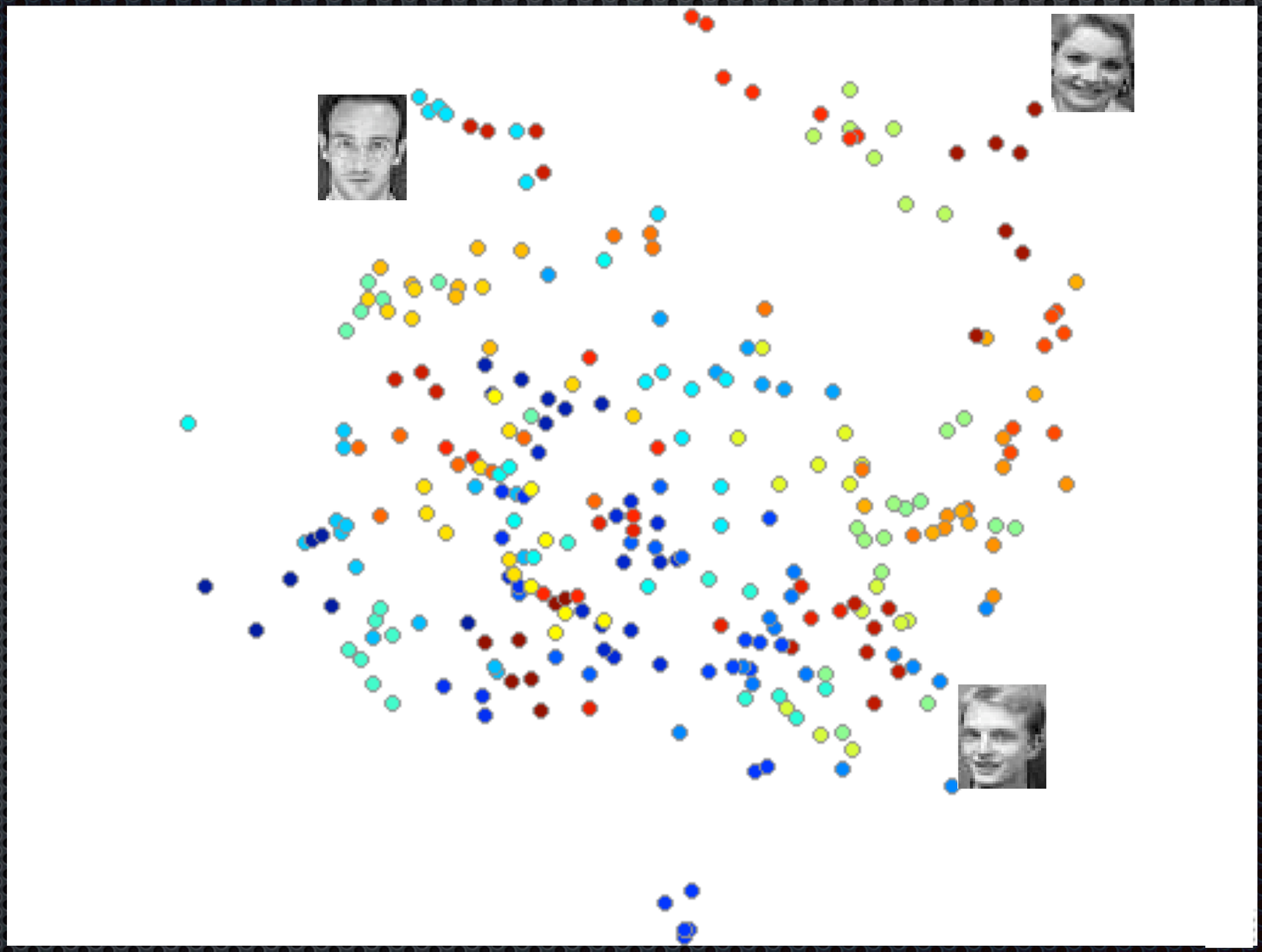
$$\text{subject to:} \quad \forall j \rightsquigarrow i, (l,i) \in D$$

$$\|\vec{x}_i - \vec{x}_j\|^2_{\mathbf{M}} + 1 \leq \|\vec{x}_i - \vec{x}_l\|^2_{\mathbf{M}} + \xi_{ijl}$$

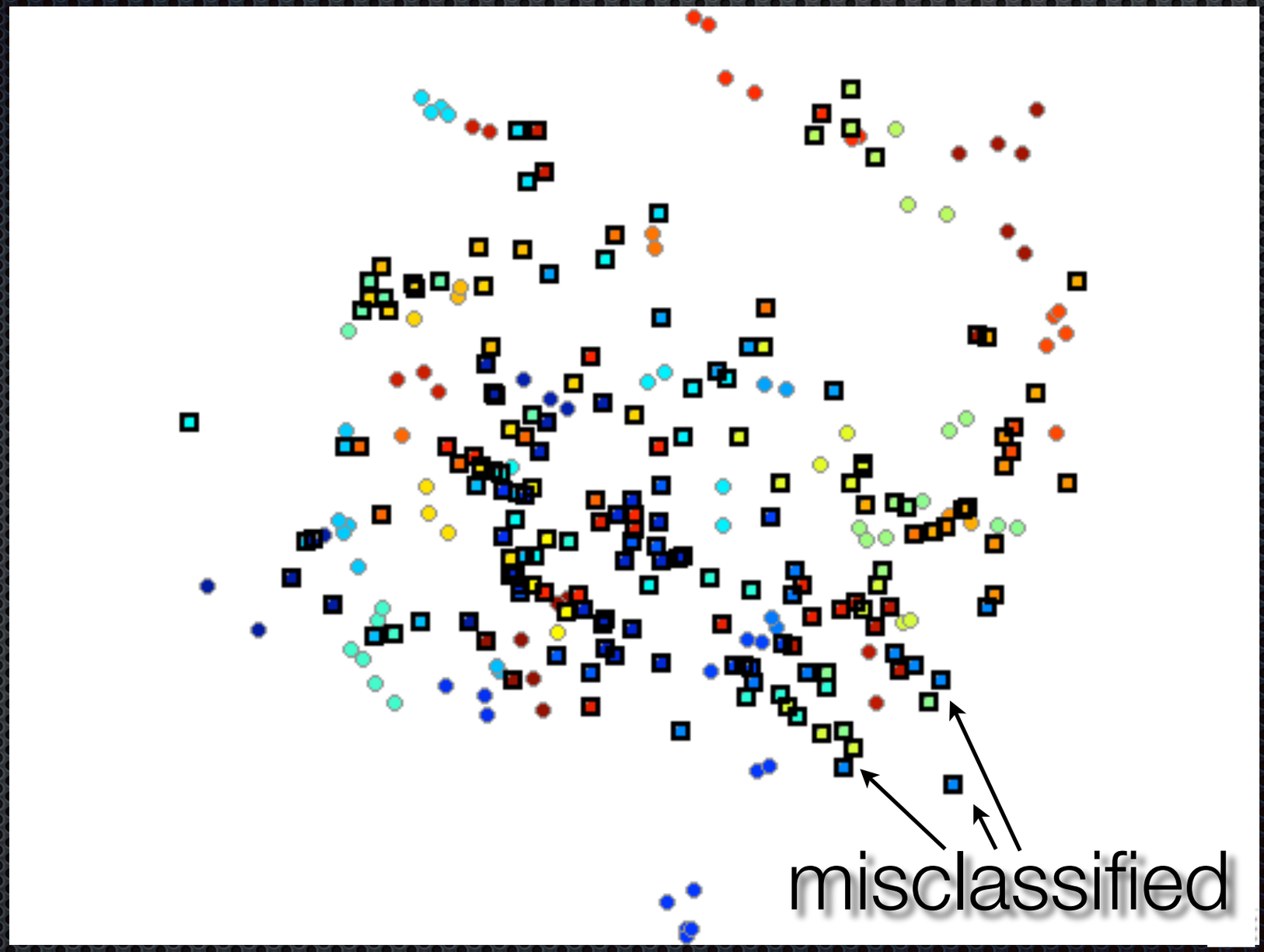
$$\mathbf{M} \succeq 0, \ \xi_{ijl} \geq 0$$

Special-purpose solver [Weinberger 2009]
MNIST: 3.2 Billion constraints in 20 minutes

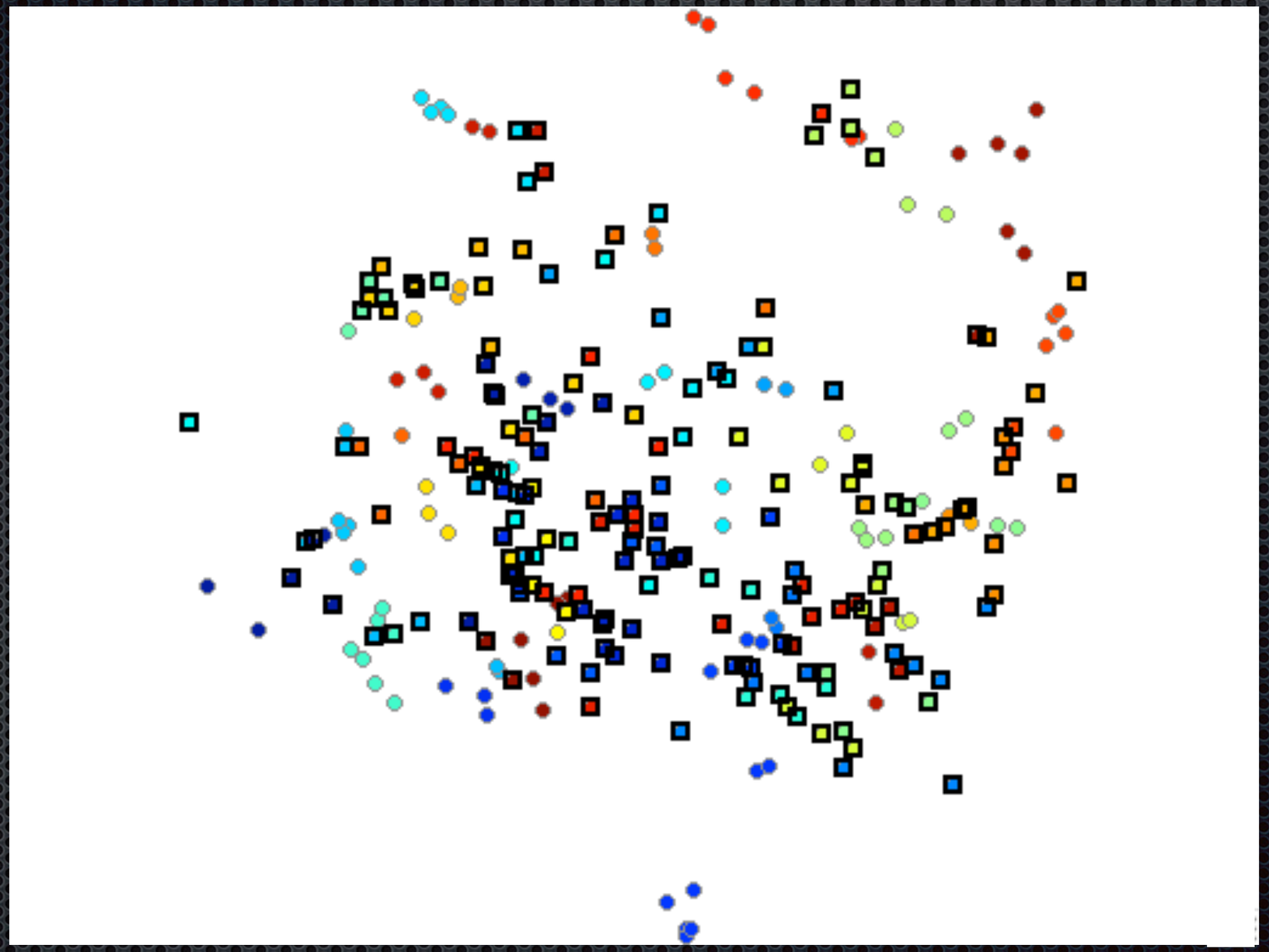
Faces data set



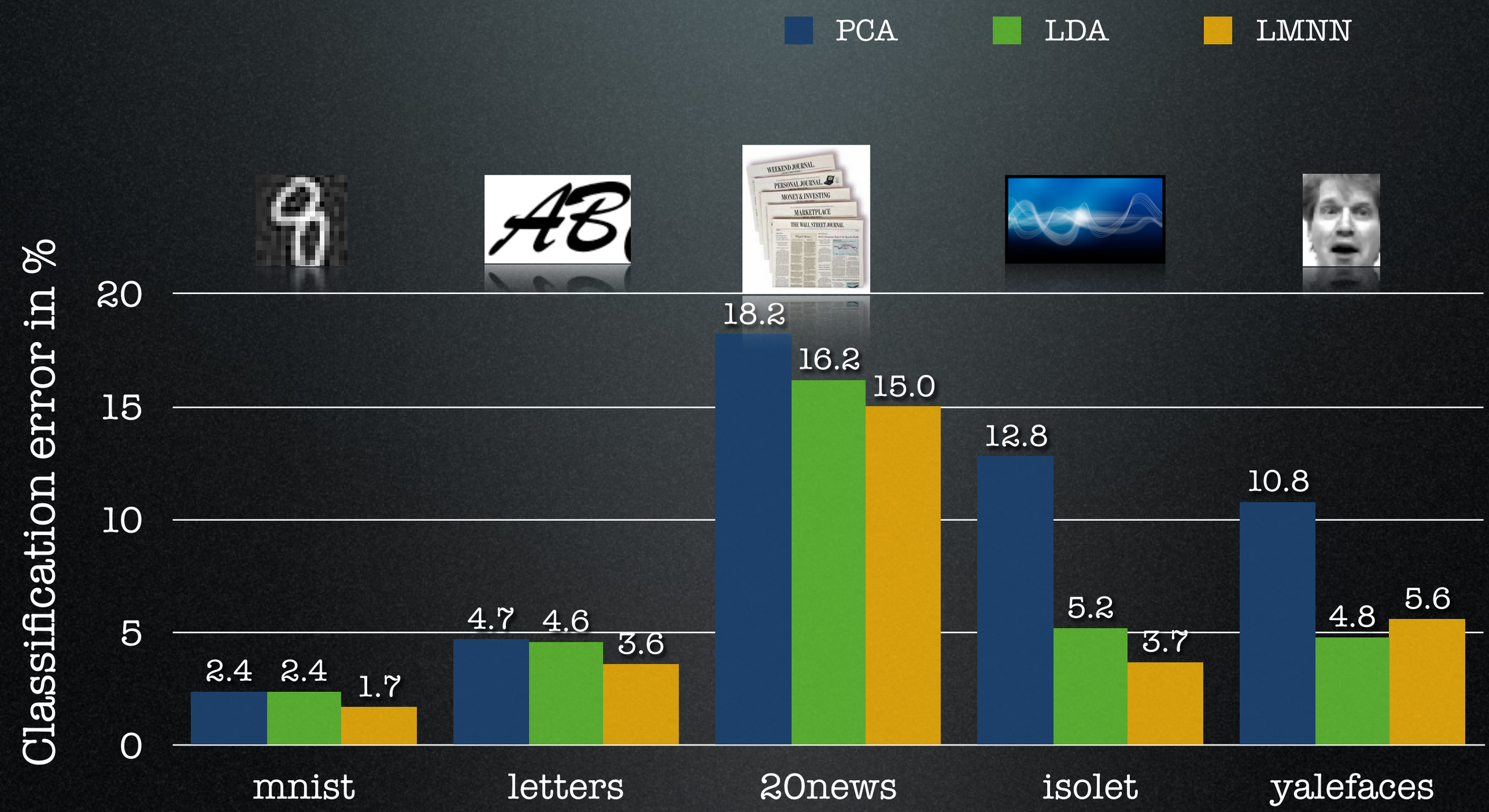
Faces data set



Faces data set

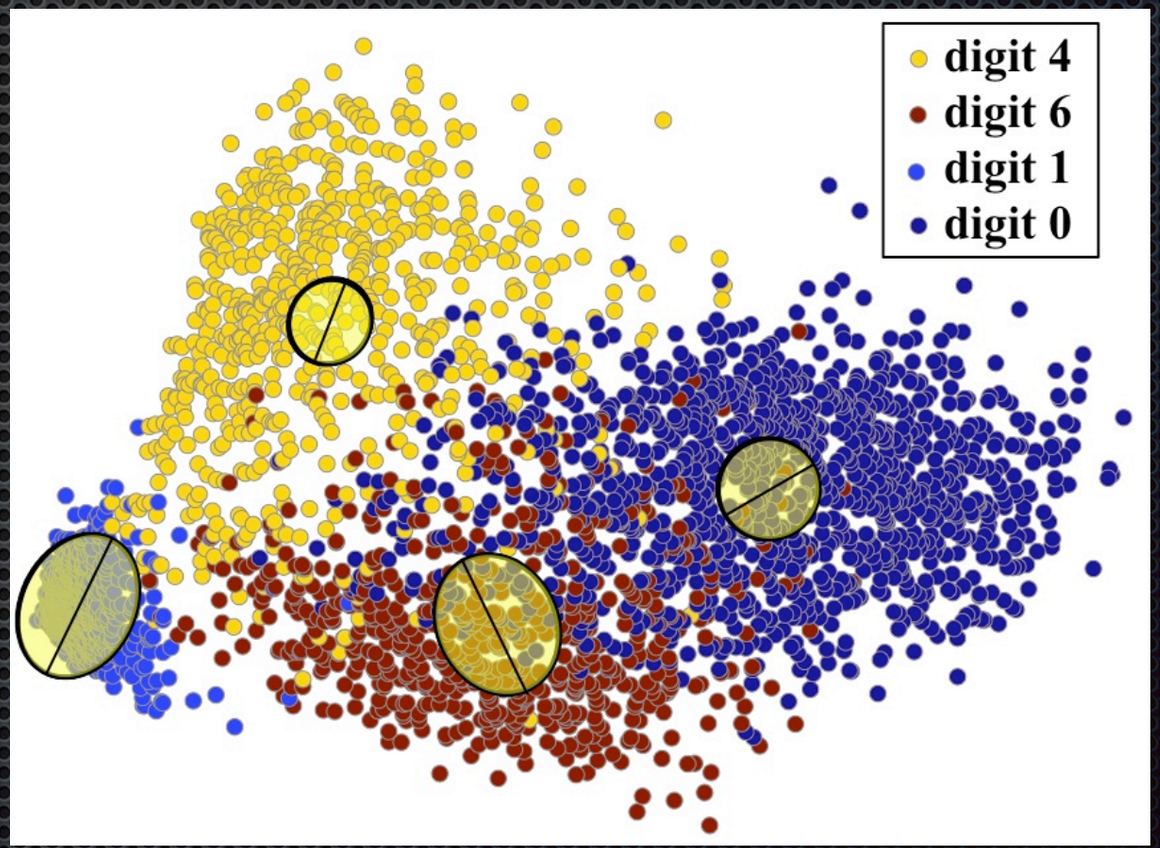


RESULTS


PCA
LDA
LMNN
Classification error in %
20
15
10
5
0
18.2
16.2
15.0
12.8
10.8
5.6
4.8
5.2
4.7
4.6
3.6
3.7
2.4
2.4
1.7
mnist
letters
20news
isolet
yalefaces

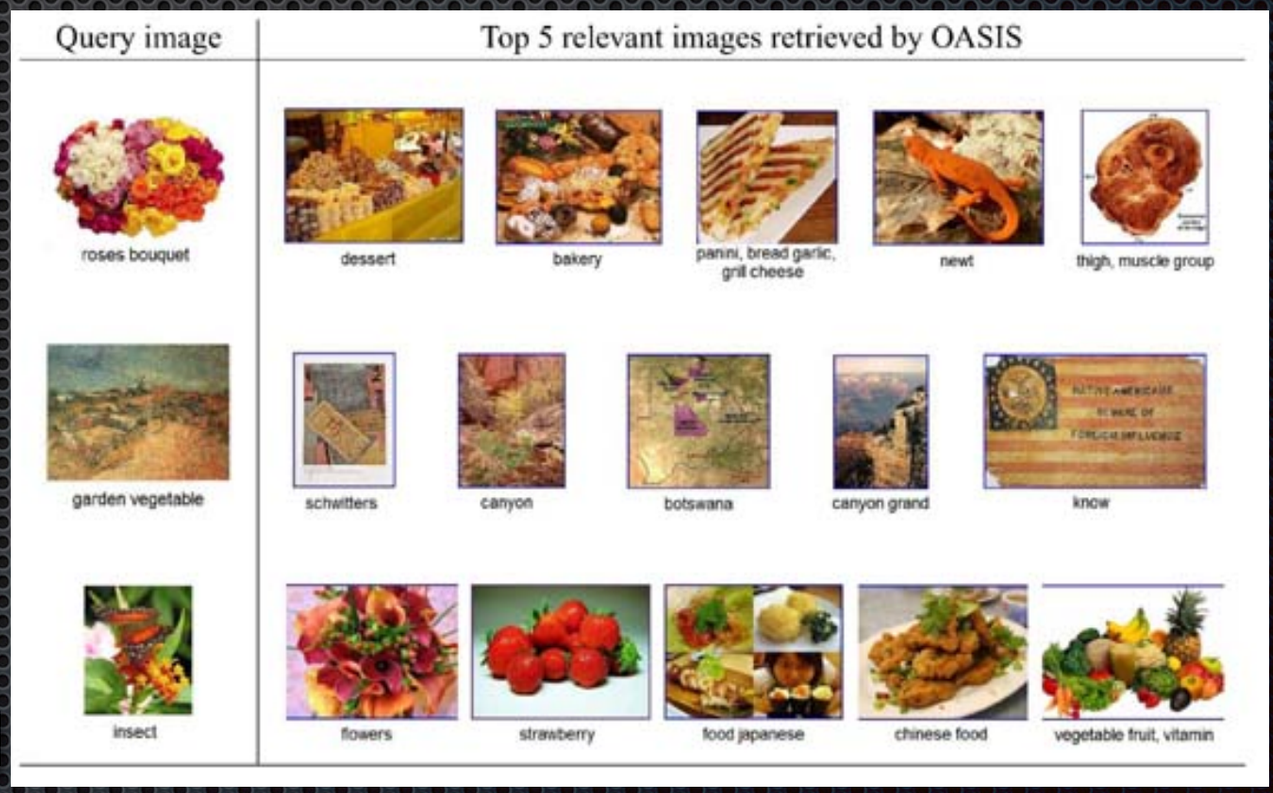
Extensions ...

- Torresani and Lee, [NIPS 2006]
  - **Kernelization** (significantly slower but more powerful)
  - **Linear dimensionality reduction** (potentially better classification results, but breaks convexity)
- Weinberger and Saul, [ICML 2008]
  - Many **locally linear metrics** instead of a single global metric.
  - **Fast nearest neighbor retrieval** with ball-tree data structures.
- Kumar et al., [ICCV 2007]
  - i-LMNN metric **invariant** to **polynomial** input **transformations**



Extensions … (cnt'd)

- Shen et al., [NIPS 2008]
  - Optimize through **boosting**
  - Weak learners: rank 1 PSD matrices
- Chechik et al., [NIPS 2009]
  - **Online version**.
  - Scales to **gigantic** data sets.
  - Relaxation of SDP constraint
- Parameswaran and Weinberger [NIPS 2010]
  - **Multi-class** adaptation
  - **Transfer learning**

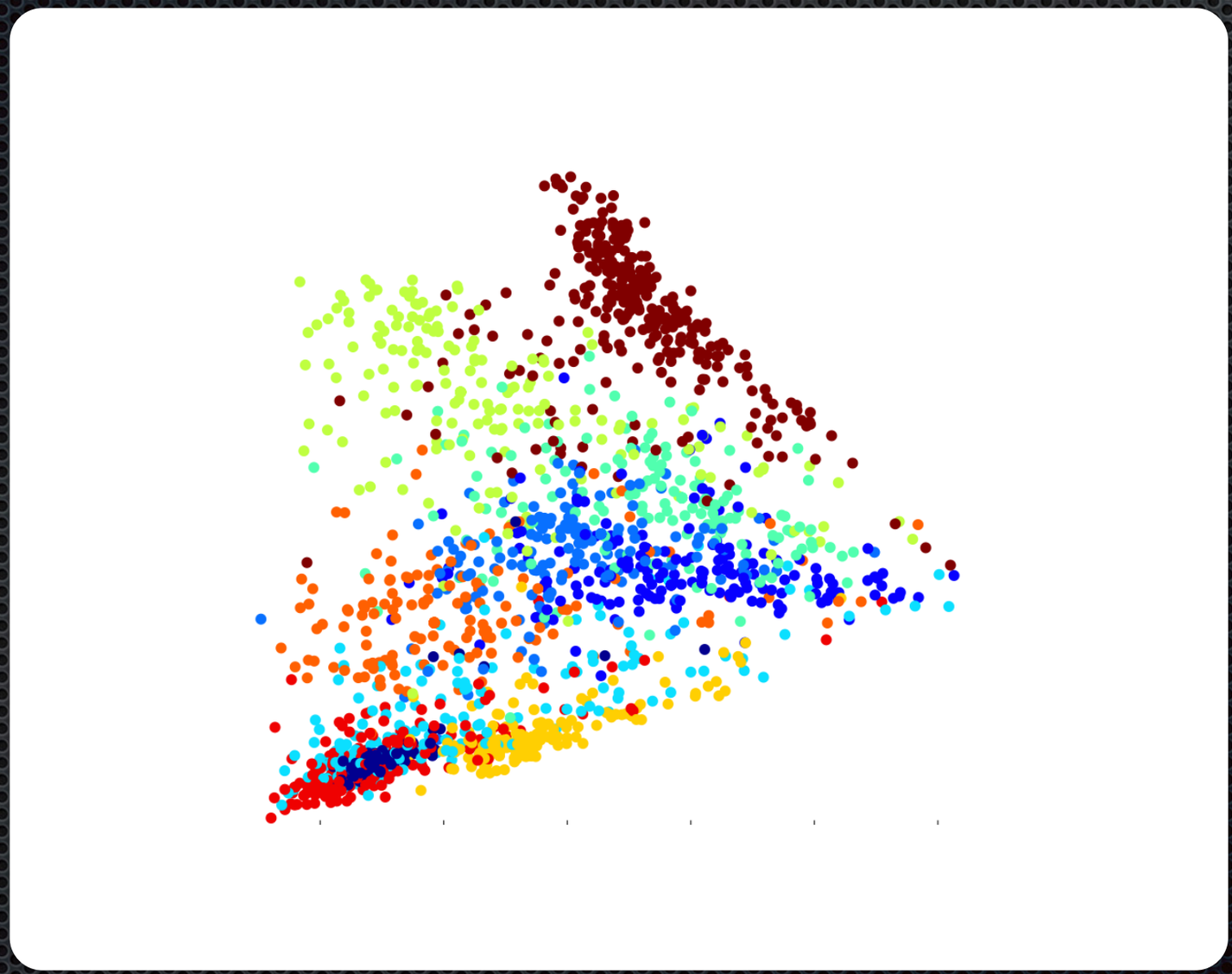


[Courtesy of Chechik et al.]

Summary of LMNN

- Learns Mahalanobis metric
- **Local margins**
- Optimization problem is convex
- Algorithm is guaranteed to converge
- Scales well -- generally best for large data
- Requires “target neighbors”

Information Theoretic Metric Learning (ITML)



[Davis et al. ICML 2007]

ITML Optimization

Define some "initial" Mahalanobis distance $\mathbf{M}_0$
(e.g. Euclidean $\mathbf{M}_0 = \mathbf{I}$)

Convex optimization:

$$\min_{\mathbf{M}} \operatorname{tr}(\mathbf{M}\mathbf{M}_0^{-1}) - \log\det(\mathbf{M}\mathbf{M}_0^{-1})$$

$$(\vec{x}_i - \vec{x}_j)^\top \mathbf{M}(\vec{x}_i - \vec{x}_j) \leq s,\ \forall (i,j) \in S$$

$$(\vec{x}_i - \vec{x}_\ell)^\top \mathbf{M}(\vec{x}_i - \vec{x}_\ell) \geq d,\ \forall (i,\ell) \in D$$

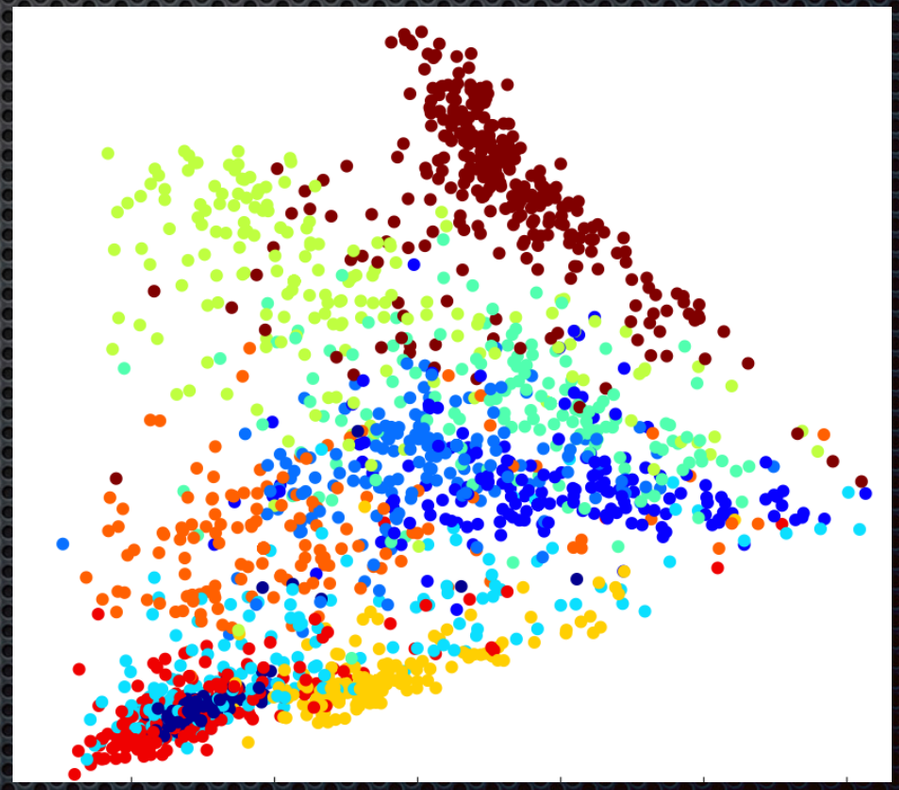
Stein's Loss

$$\min_{\mathbf{M}} \operatorname{tr}(\mathbf{M}\mathbf{M}_0^{-1}) - \log\det(\mathbf{M}\mathbf{M}_0^{-1})$$

- Keep metric **close to initial guess** (e.g. Euclidean)
- **Automatically preserves positive semidefiniteness**
- Allows cheap rank-1 updates

Extensions ...

- P. Jain, B. Kulis, I. S. Dhillon, K. Grauman., [NIPS 2009]
  - **Online** formulation
  - **Local Sensitive Hashing** for fast similarity search
- Z. Lu, P. Jain, I. S. Dhillon. [ICML 2009]
  - **Geometry aware** metric learning.
  - Data set **Visualization**.
- P. Jain, B. Kulis, I. S. Dhillon. [NIPS 2010]
  - Link with **kernel learning**.
  - Generalization of **regularizers**



[Zhengdong et al.]

Summary of ITML

- Mahalanobis metric for k-NN
- Performs well in practice
- Simple updates
- Scales well (online algorithm)
- **Convex without PSD constraint**
- Assumes unimodal data

A few more things ...

Many more Algorithms...

- Semi-supervised M. learning (EM update) **[Bilenko et al.; ICML'04]**
- Metric Learning for SVM: SVML **[Zu et al., TR'11]**
- Siamese neural networks **[Chopra et al., CVPR'05]**
- SVM-style feature reweighing **[Schultz & Joachims, NIPS'02]**
  - Generalization **[Kwok and Tsang; ICML'03]**
- Discr. Adaptive Nearest Neighbors **[Hastie & Tibshirani; PRMI'96]**
- Generative local metric learning for n.n. **[Noh et al.; NIPS'10]**
- ... many more

About High Dimensionality...

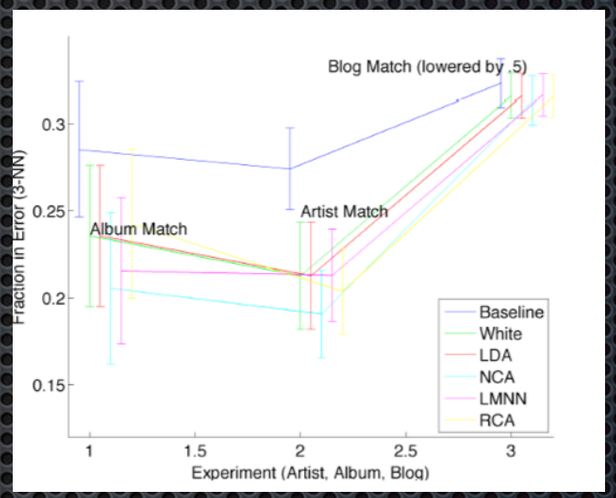
- Mahalanobis matrix is dxd
- BUT:
  - All algorithms from this talk can be kernelized
  - Kernelization simple: Everything in terms of dot-products
  - Close relationship between **kernel**- and **metric** learning

[Chatpatanasiri et al.; ArXiV 2008]

Applications

Face Verification
[Chopra et al. 2005]

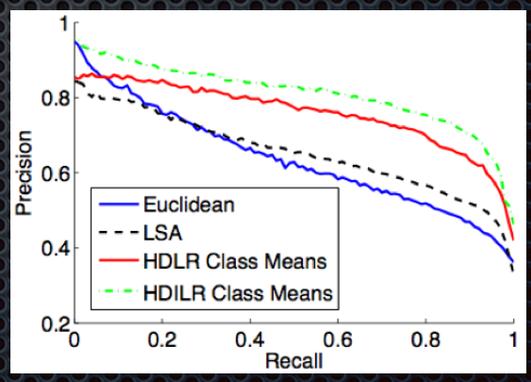
Object Recognition
[Fromme et al. 2007]



Music Similarity
[Slaney et al. 2008]

Pedestrian detection
[Dikmen et al. 2010]

Pose estimation
[Jain et al. 2010]



Text retrieval
[Davis et al. 2008]

When use what?

- **Best accuracy**: LMNN, ITML
- **Online**: LMNN*, ITML*
- **Large-Scale**: LMNN, ITML
- **Small data**: NCA,MCML
- **Clustering**: MMC

Outlook

- Loose ends:
  - Non-linear metrics (so far neural nets and boosting)
  - Algorithms beyond nearest neighbors
  - Unified theory (noteworthy start [Jain et al. 2011])
  - Weaker supervision

Summary

- Metric Learning is **fun**!
- Linear transformation: Mahalanobis metric
  - Many algorithms, different strengths weaknesses
  - Very diverse applications
- Think twice before you use the Euclidean metric.
- Still a lot of open problems ...