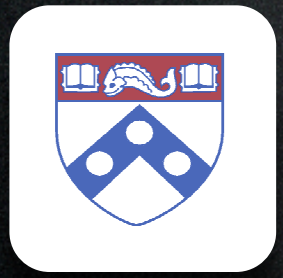
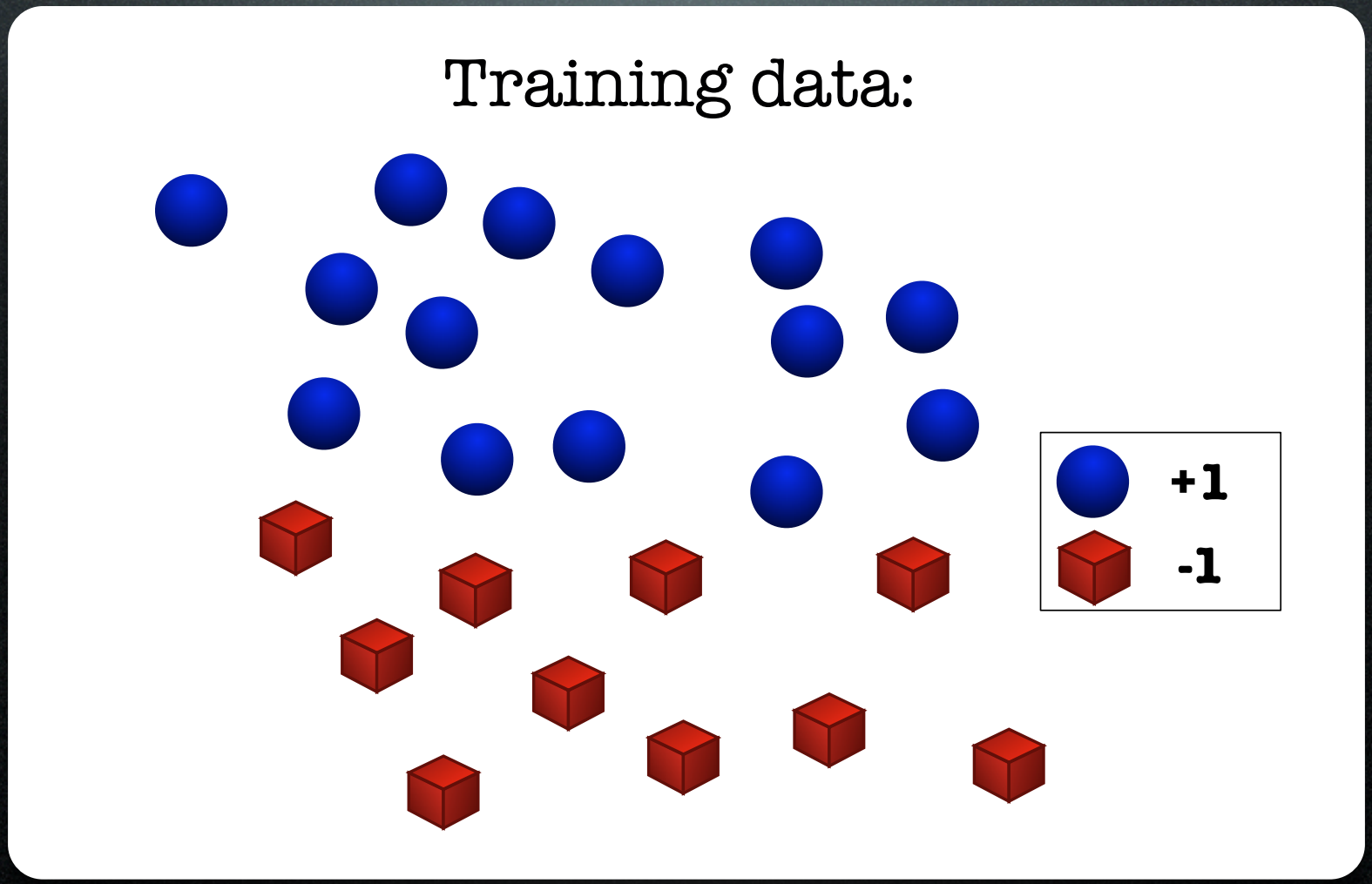
Distance Metric Learning for Large Margin Nearest Neighbor Classification



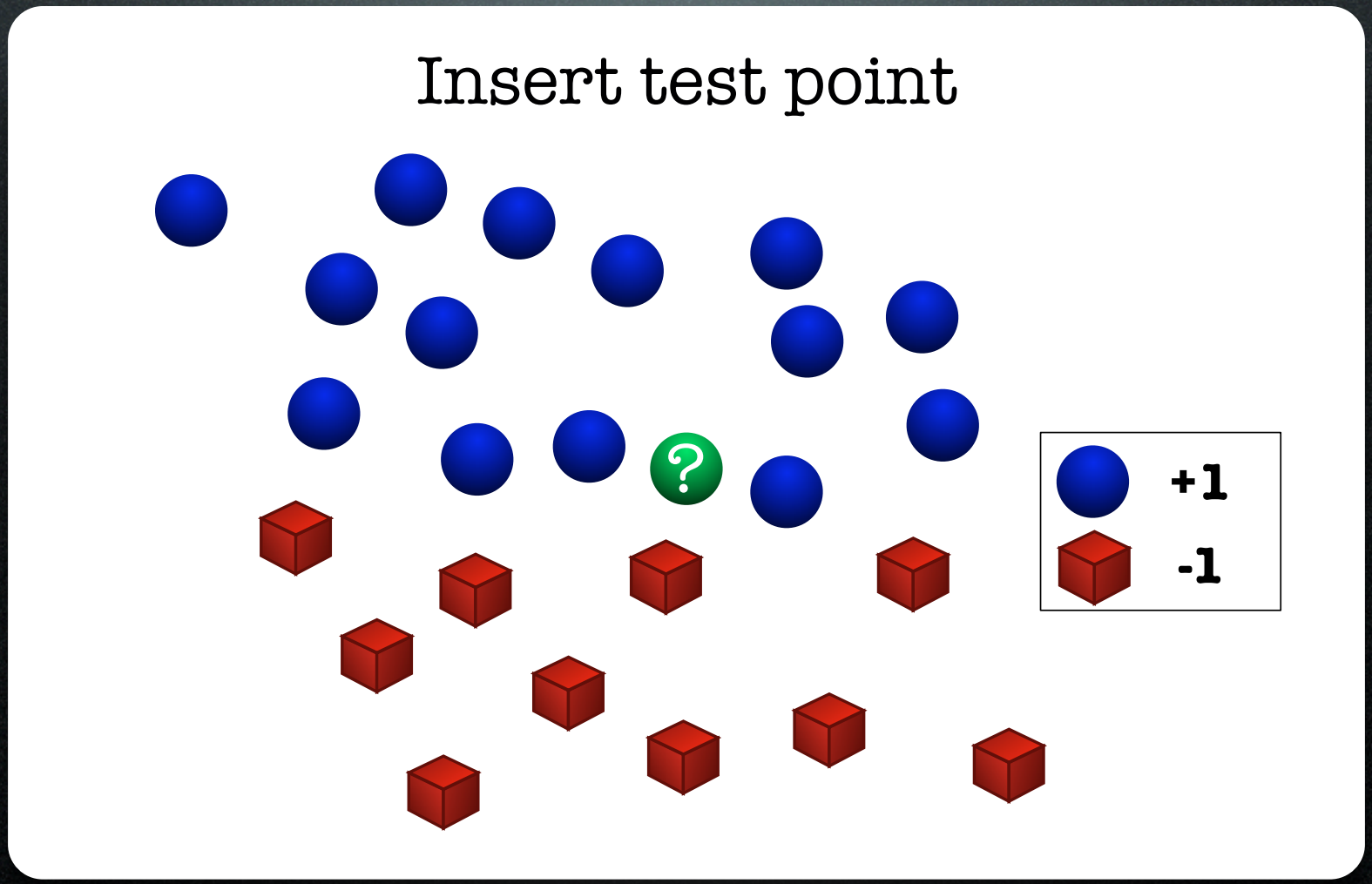
Kilian Q. Weinberger
John C. Blitzer
Lawrence K. Saul

University of Pennsylvania
GRASP Laboratory

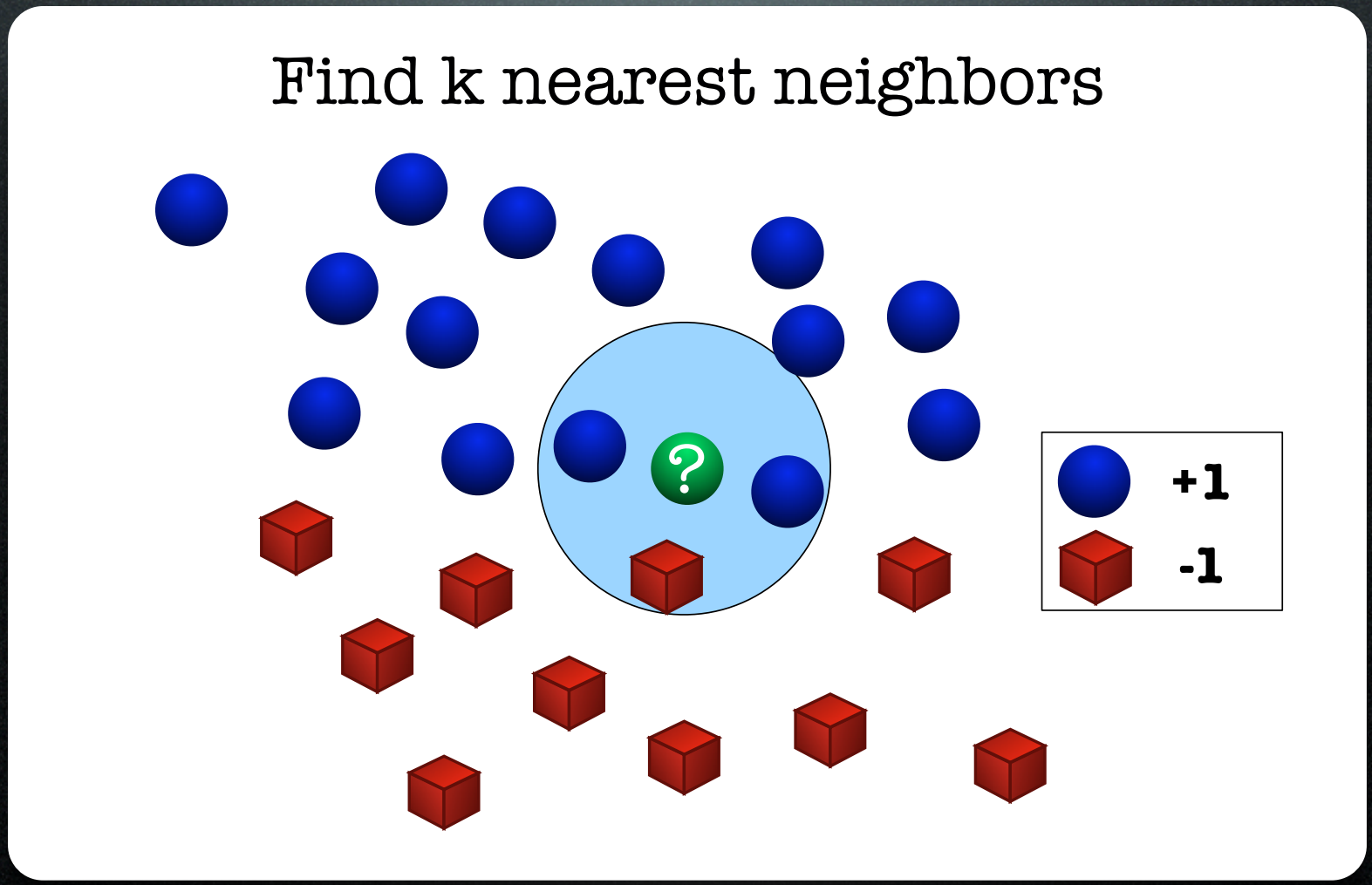
k-nearest neighbor classification



k-nearest neighbor classification

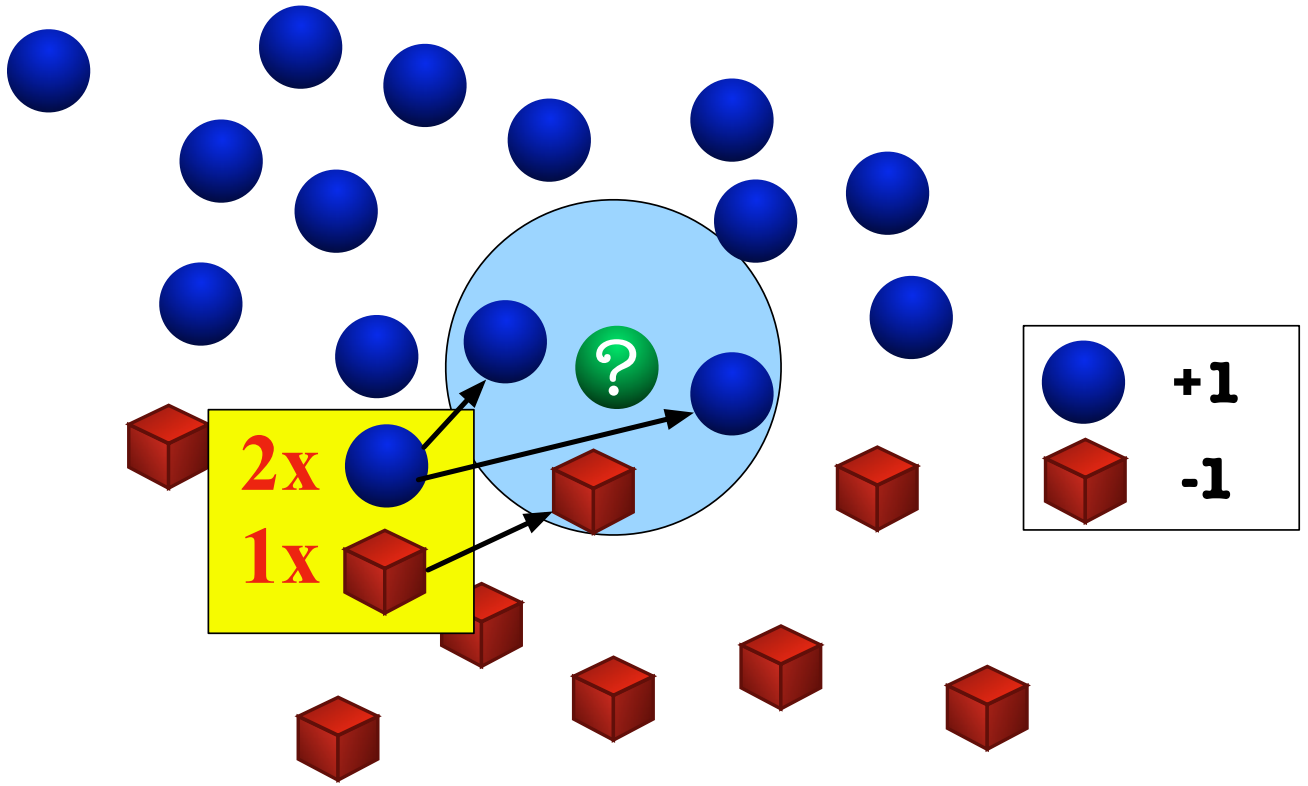


k-nearest neighbor classification



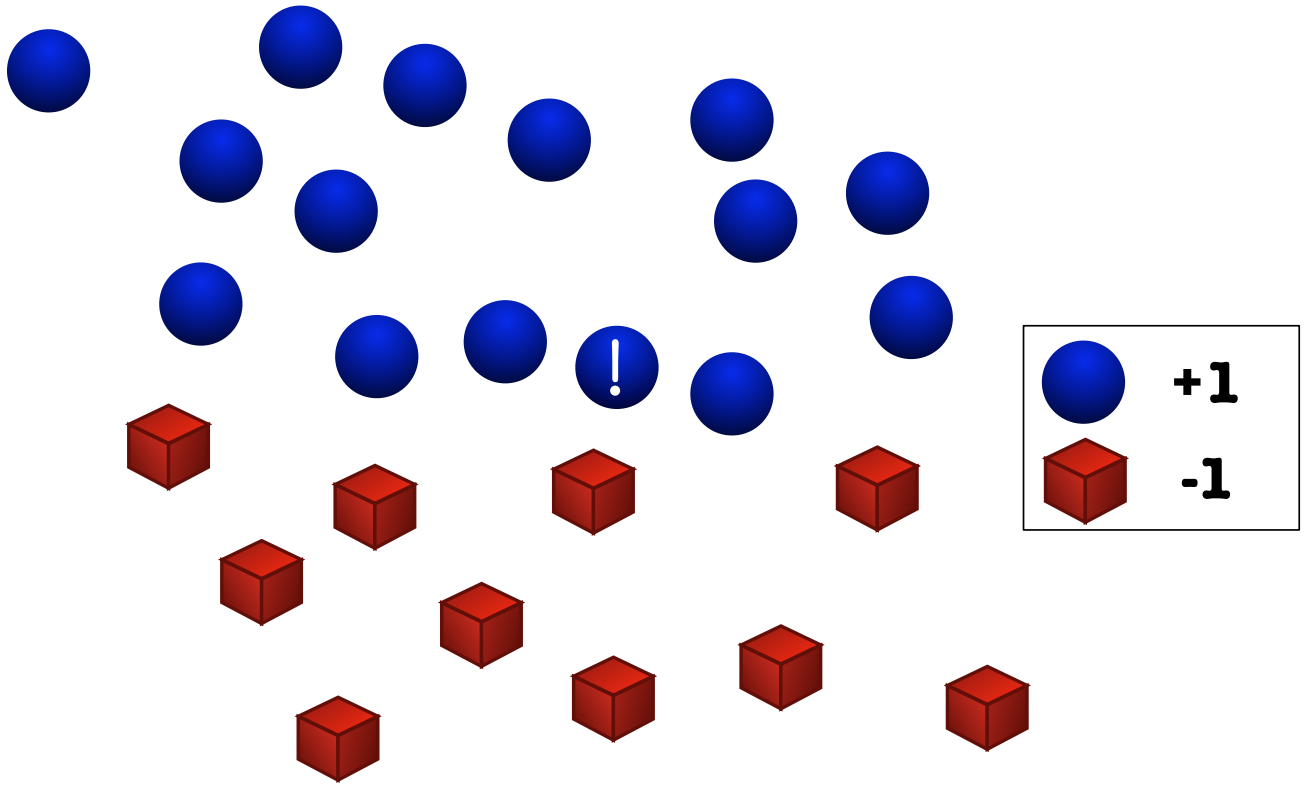
k-nearest neighbor classification

Let neighbors vote



k-nearest neighbor classification

Assign most common label



Strengths of k-NN classification

Strengths of k-NN classification

- straightforward to implement

Strengths of k-NN classification

- straightforward to implement
- nonlinear decision boundaries

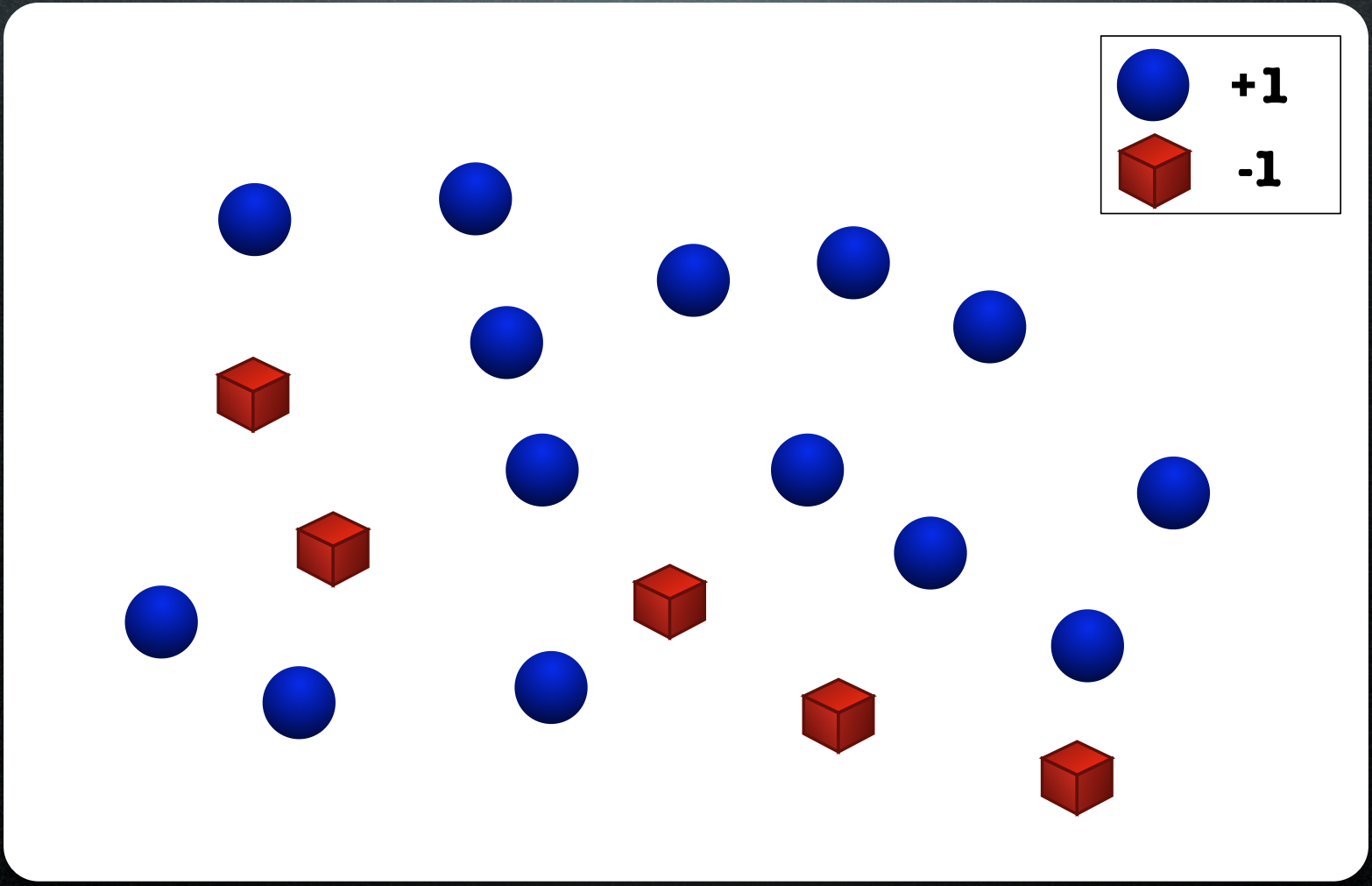
Strengths of k-NN classification

- straightforward to implement
- nonlinear decision boundaries
- complexity independent of # of classes

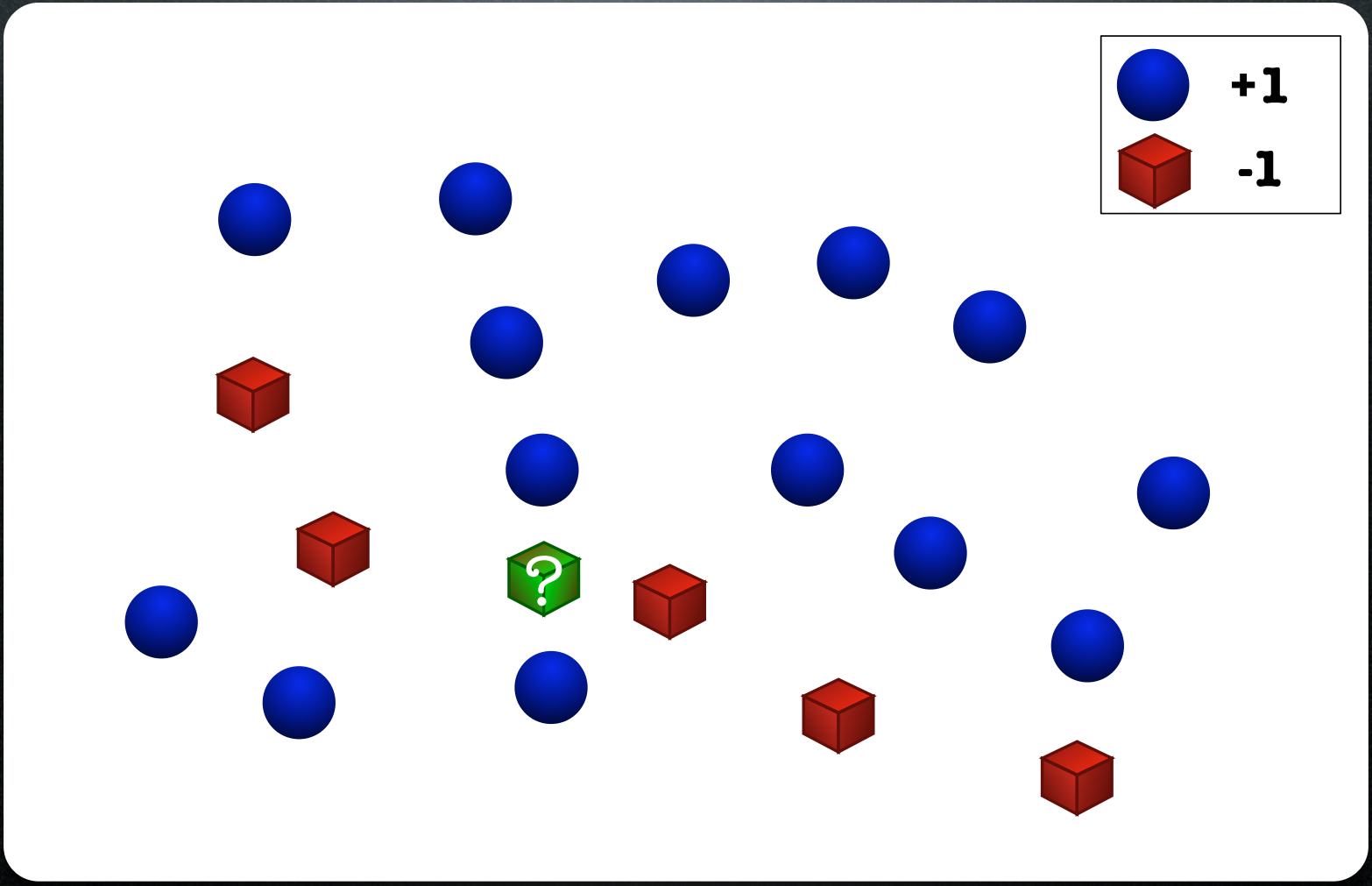
Strengths of k-NN classification

- straightforward to implement
- nonlinear decision boundaries
- complexity independent of # of classes
- asymptotic guarantees

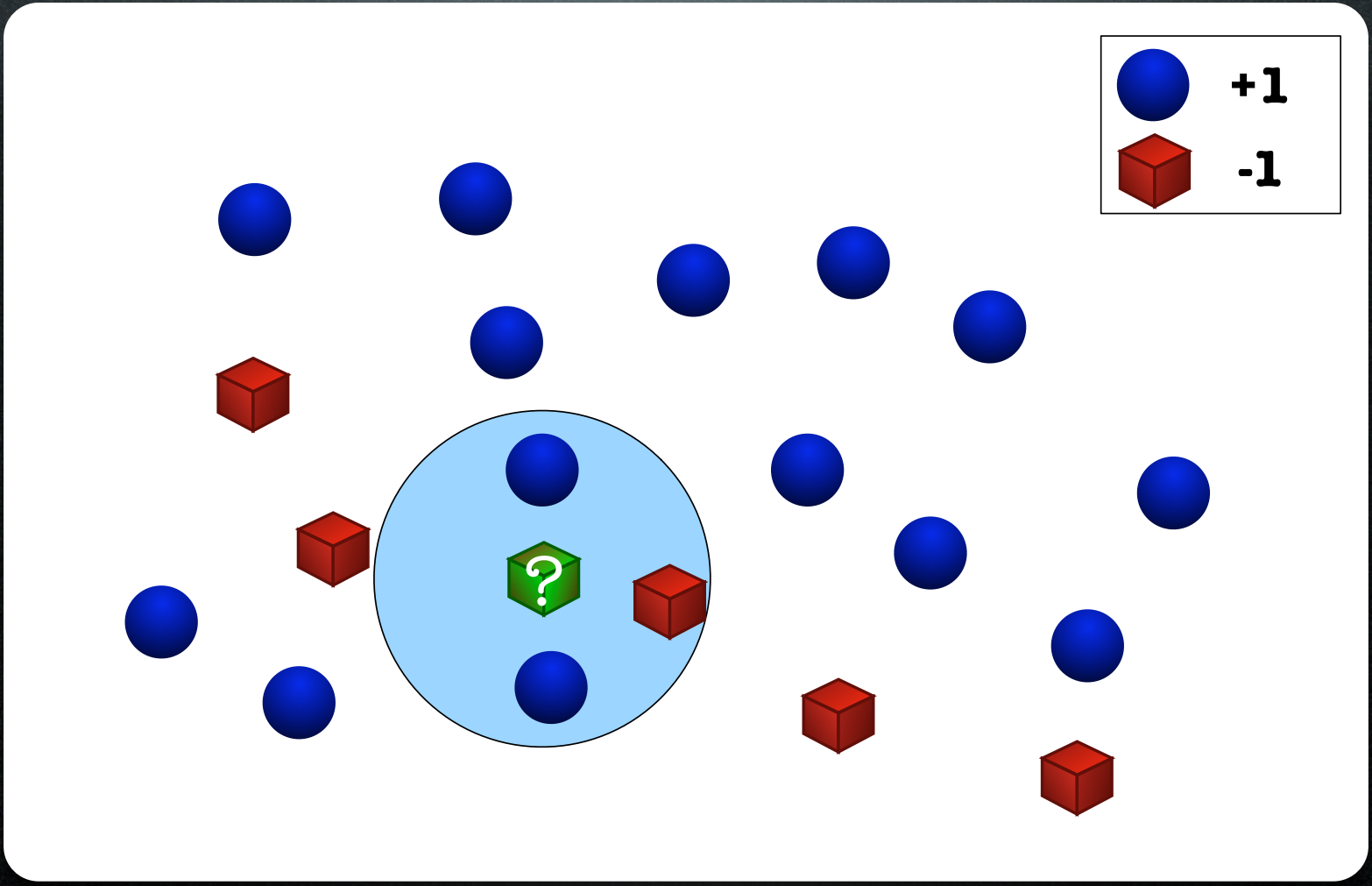
Problem with k-NN



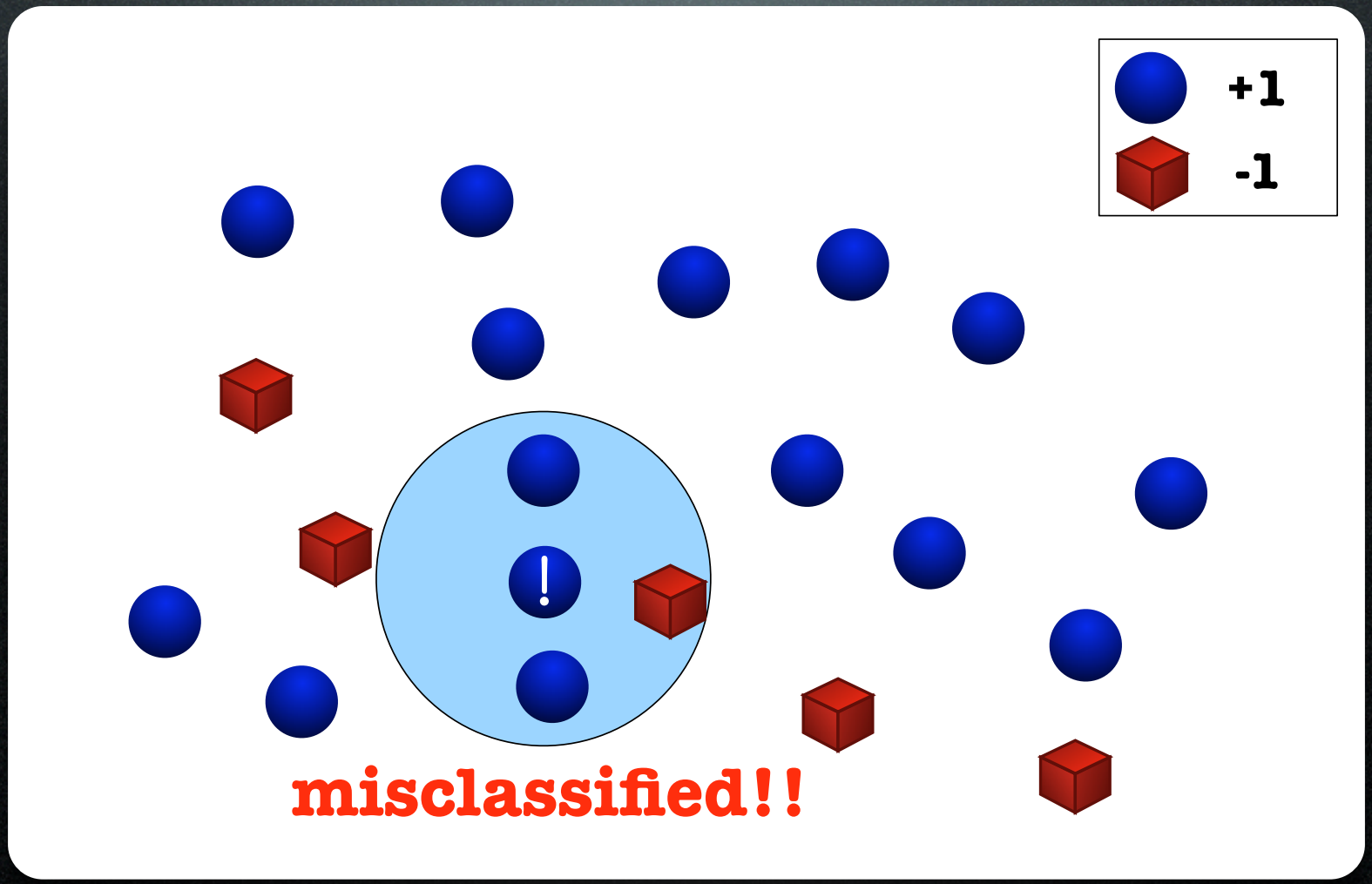
Problem with k-NN



Problem with k-NN



Problem with k-NN

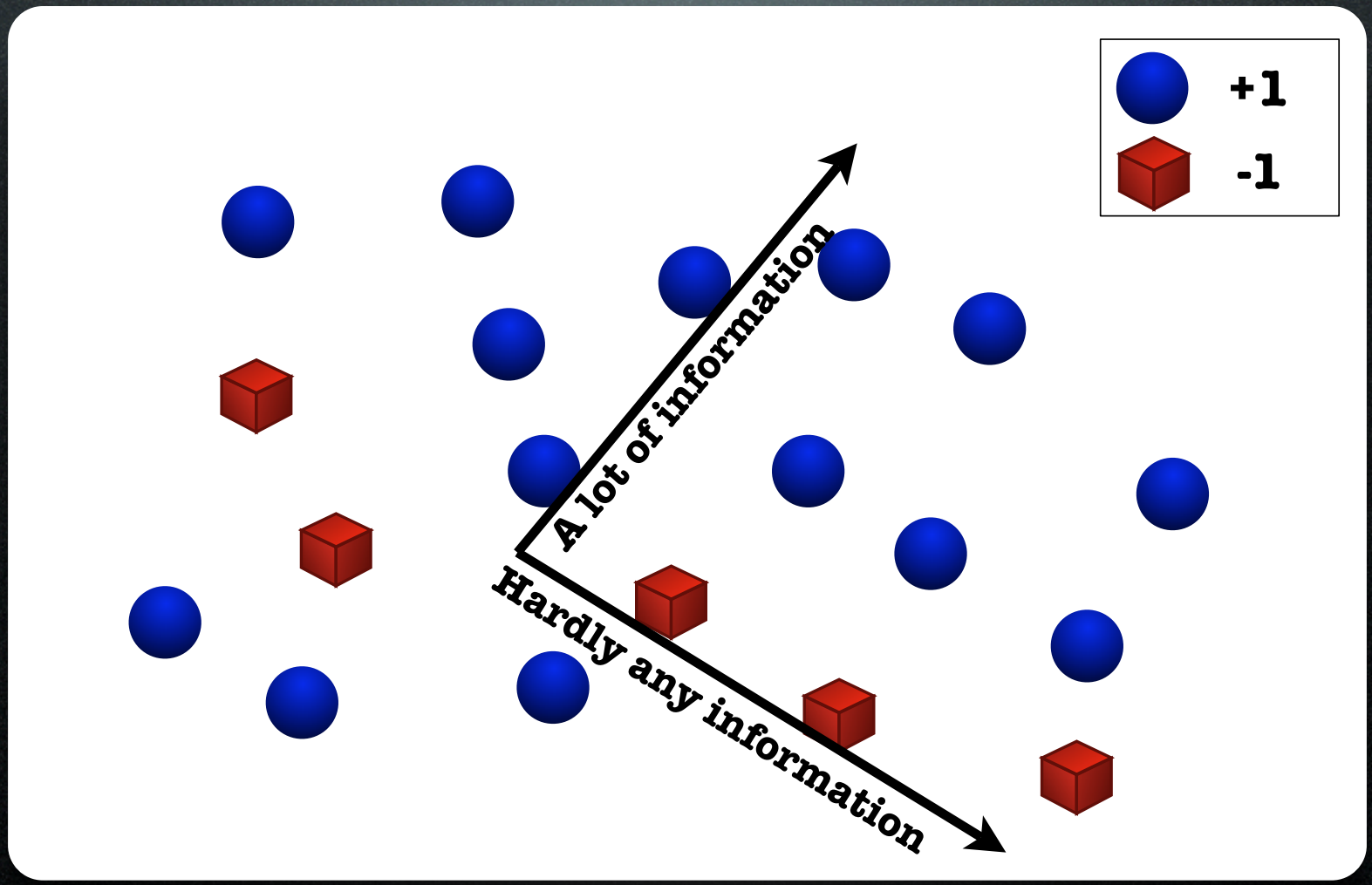


Problem with k-NN

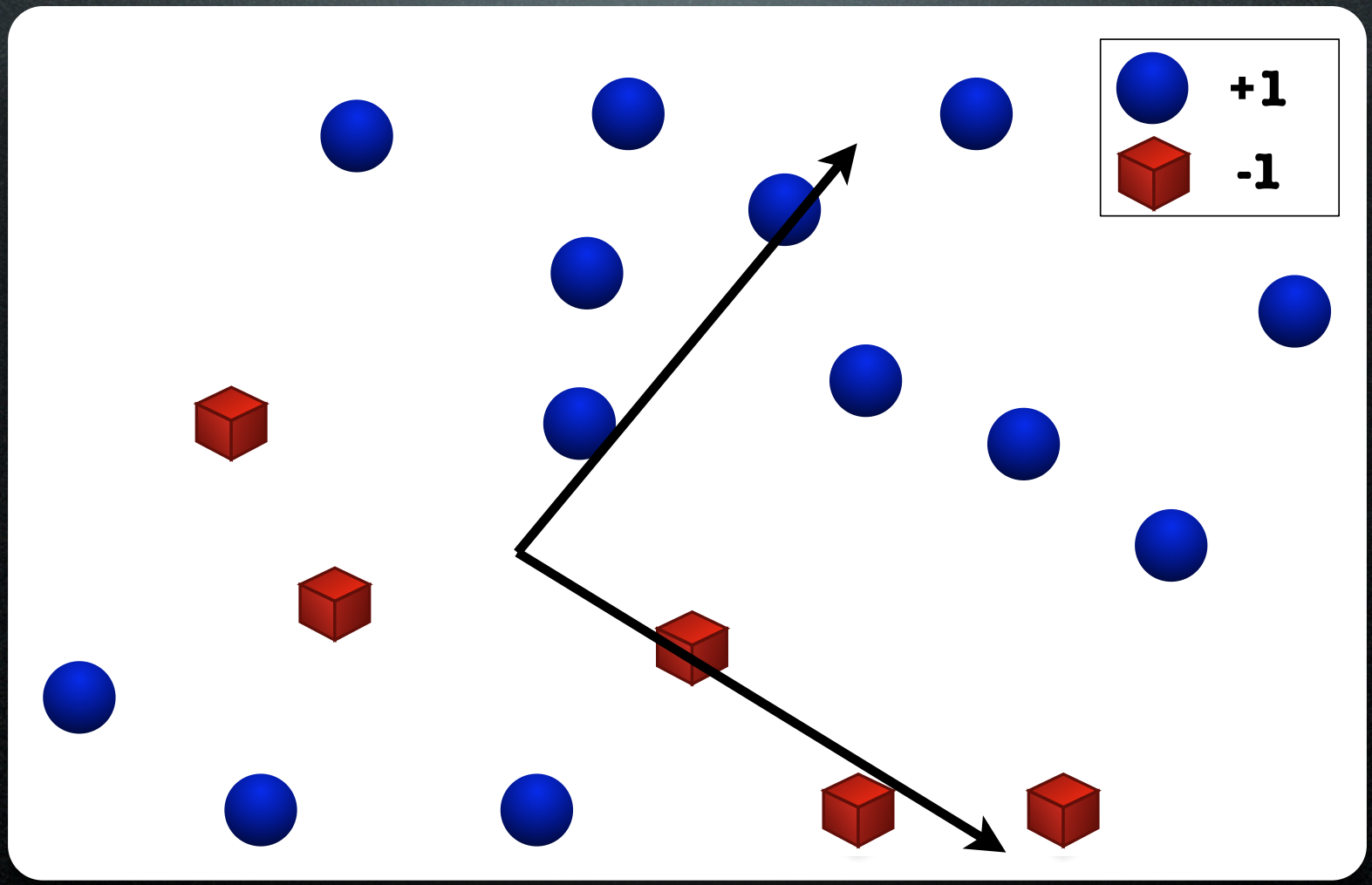
The problem:

Euclidean distances

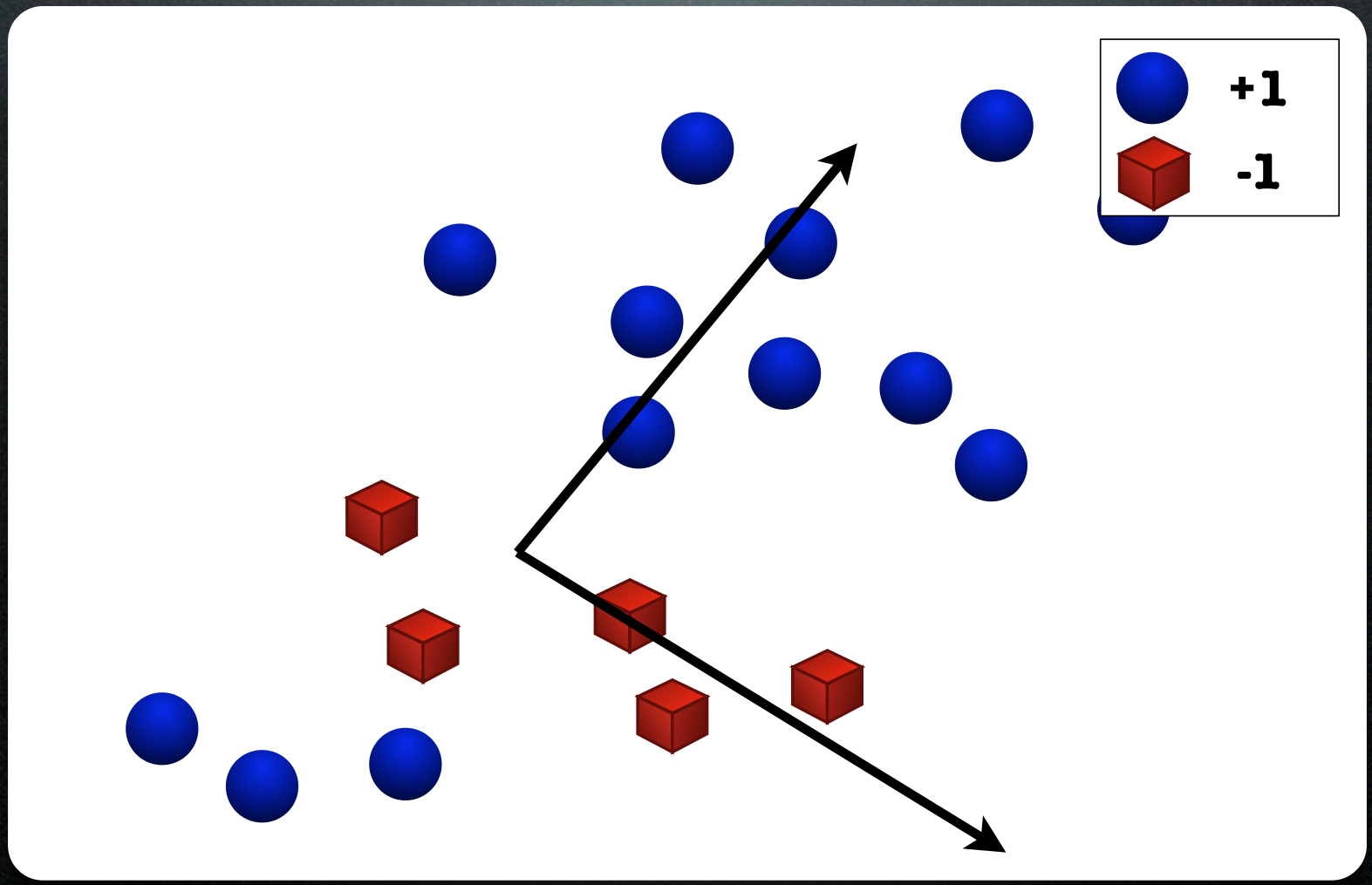
Problem with k-NN



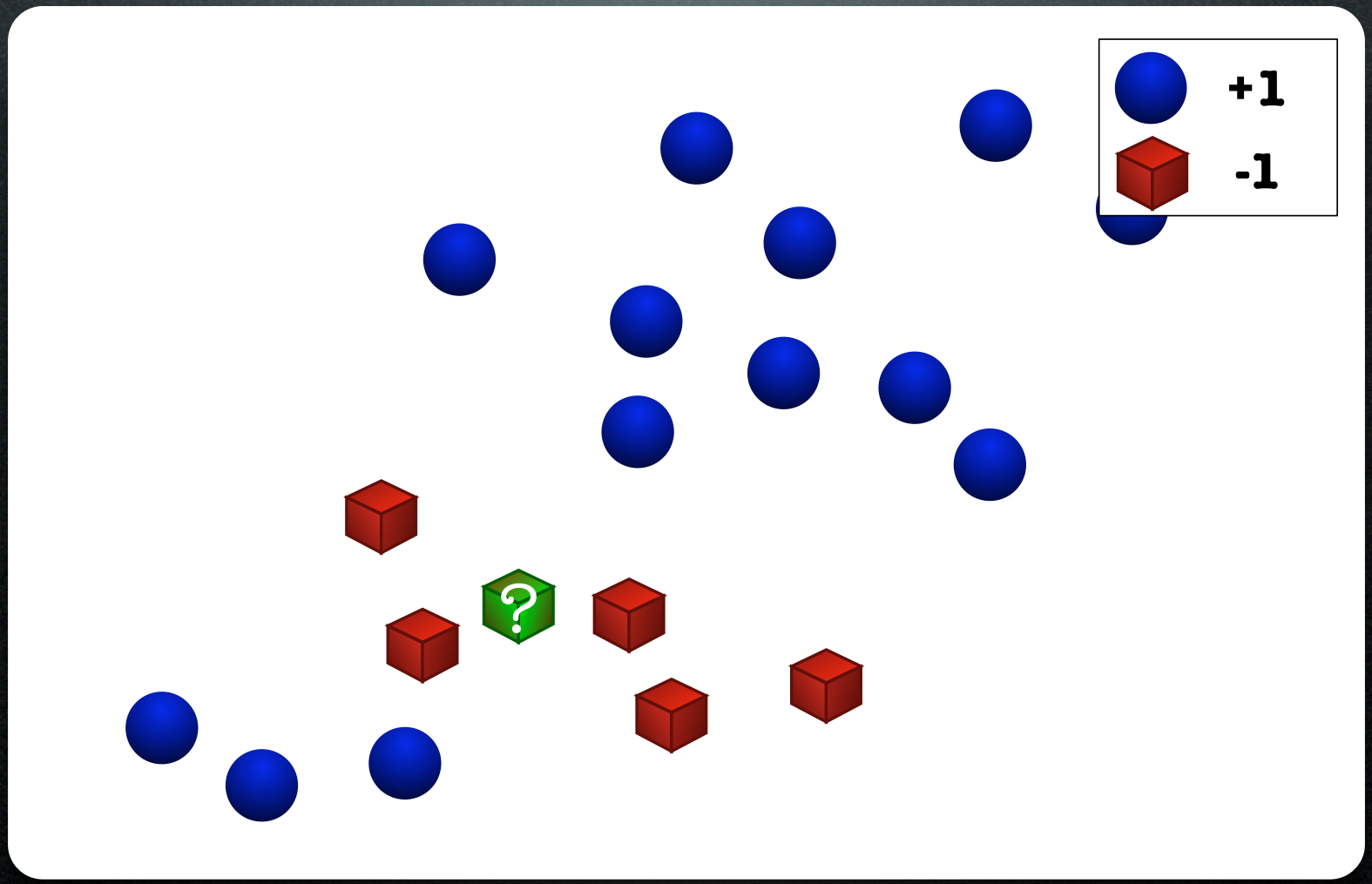
Amplify informative directions



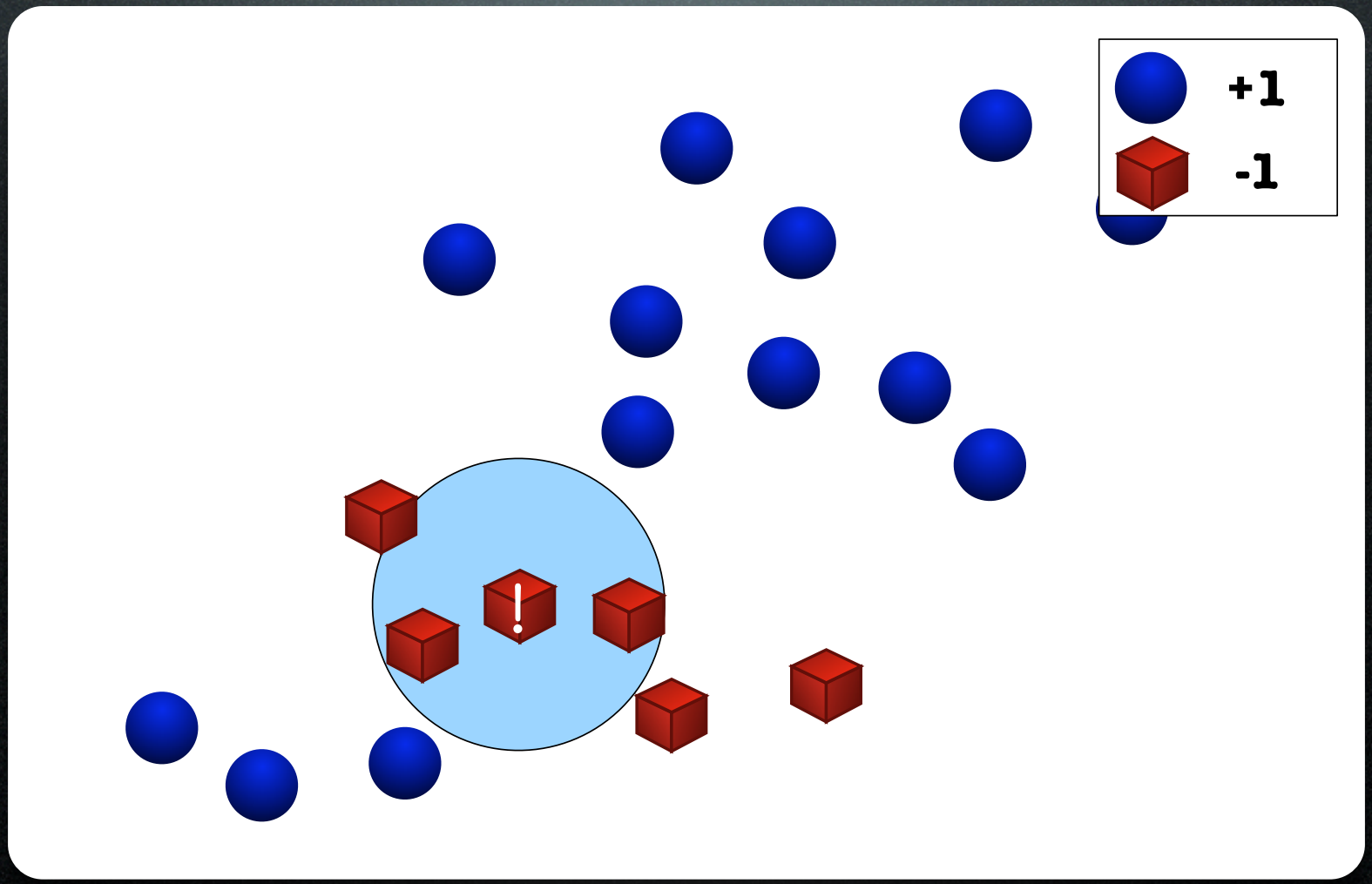
Dampen non-informative directions



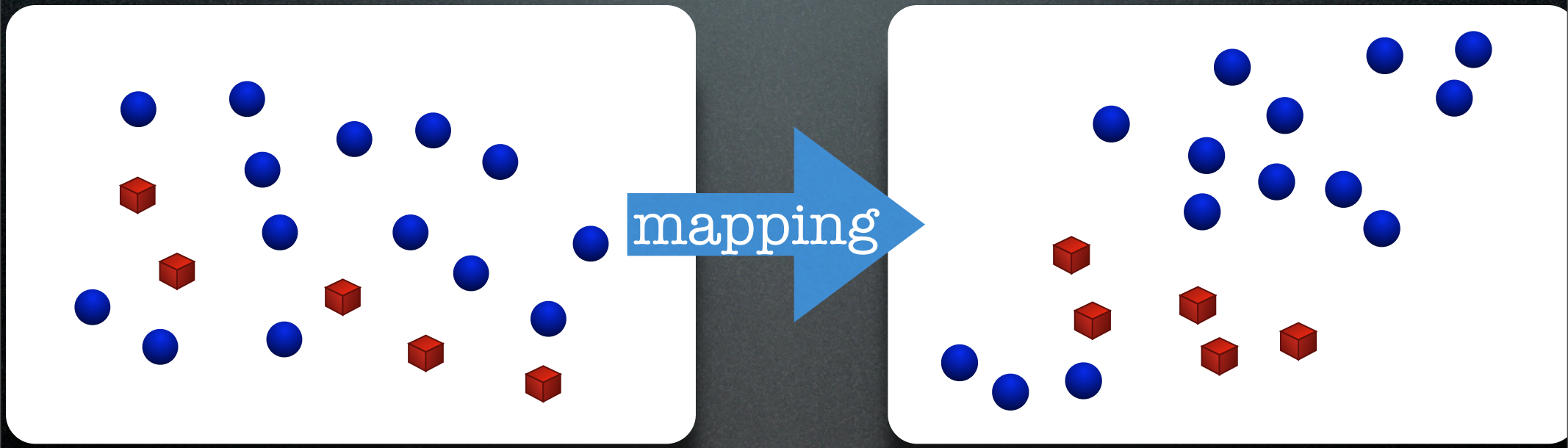
Finally!!



Finally!!



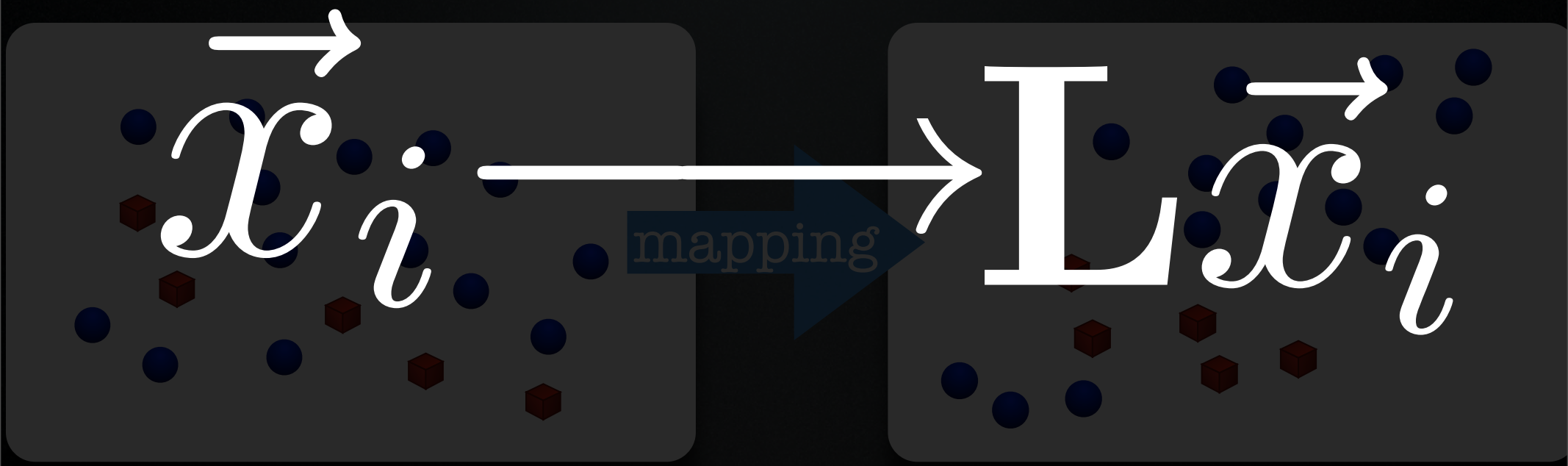
Our goal: Learn the transformation



Input space

Feature space
(meaningful distances)

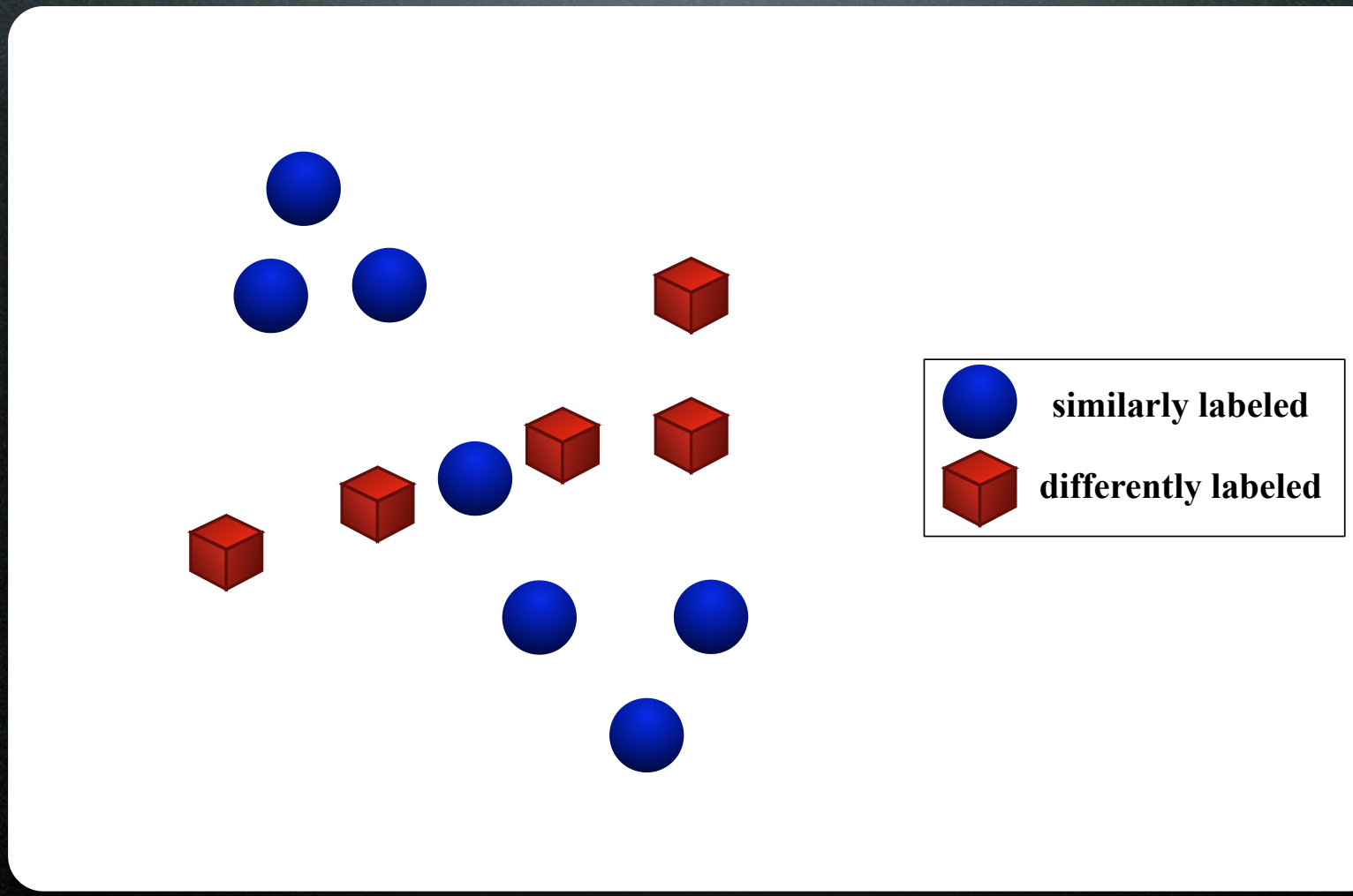
Our goal: Learn the transformation



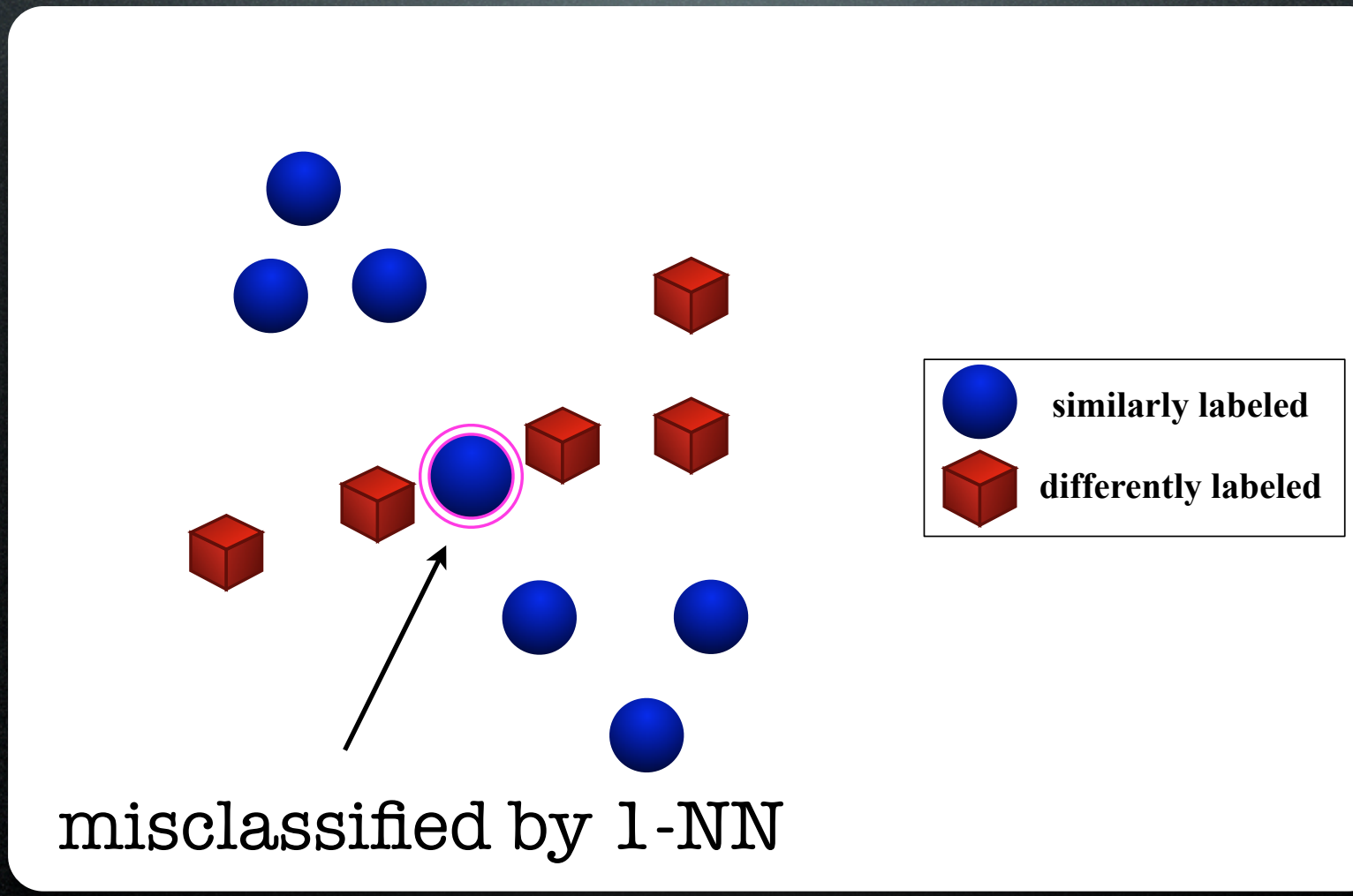
Input space

Feature space
(meaningful distances)

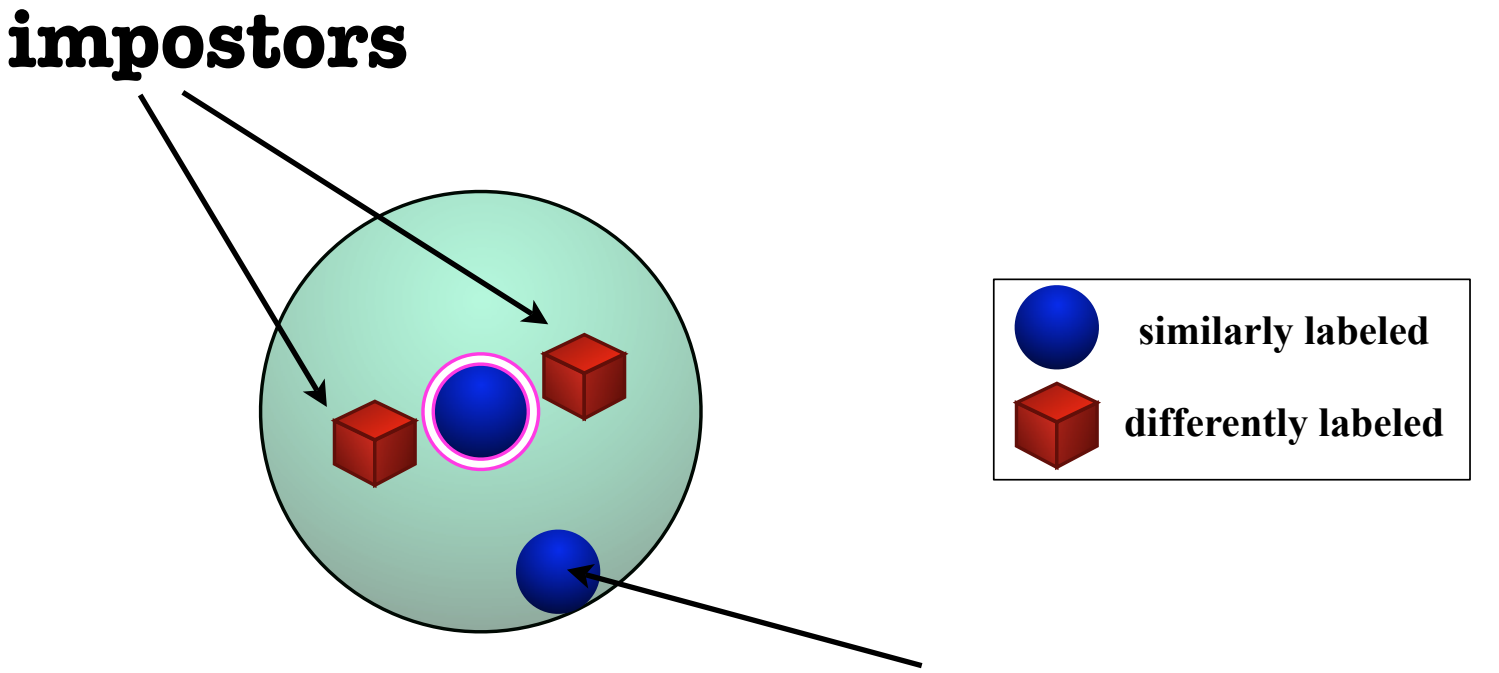
Intuition



Intuition

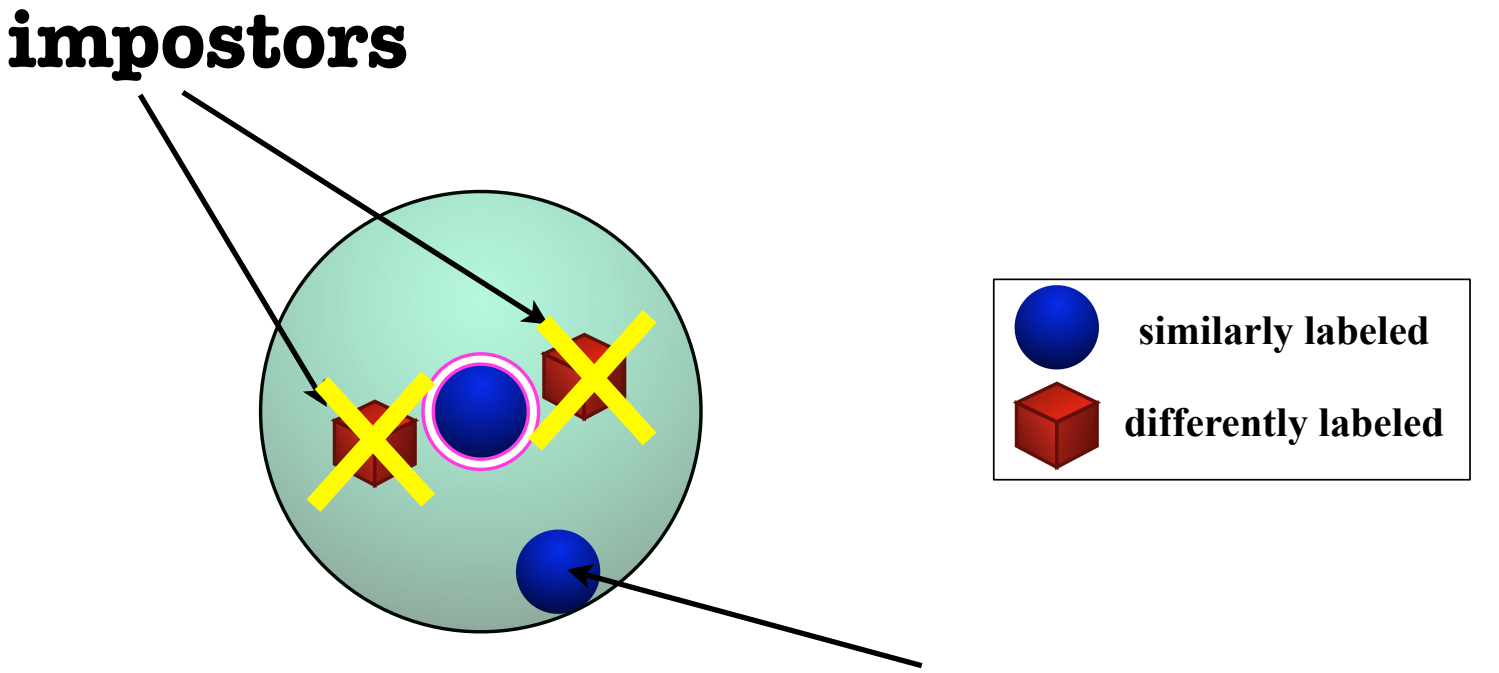

similarly labeled
differently labeled
misclassified by 1-NN

Intuition



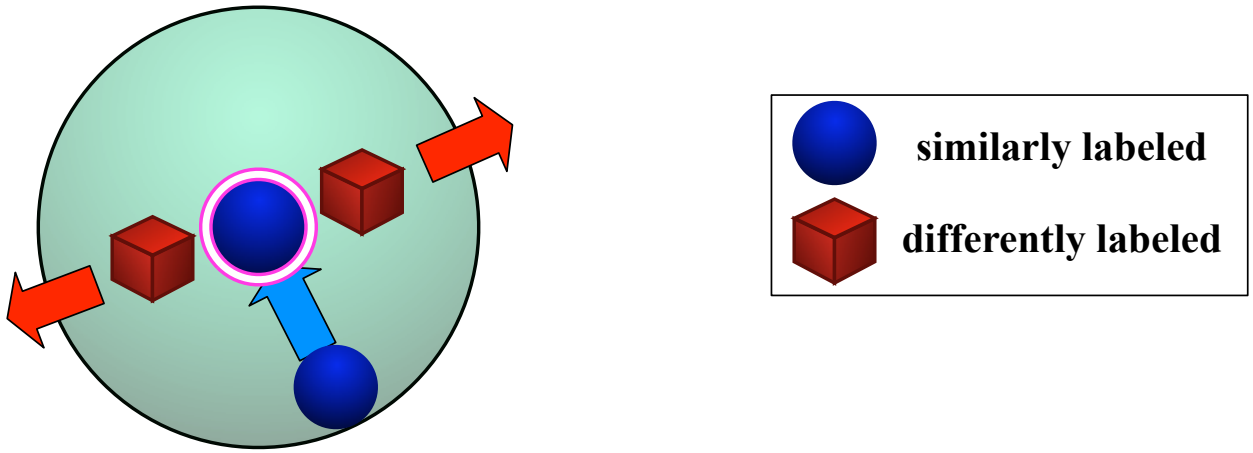
closest **similarly** labeled point
("target neighbor")

Intuition



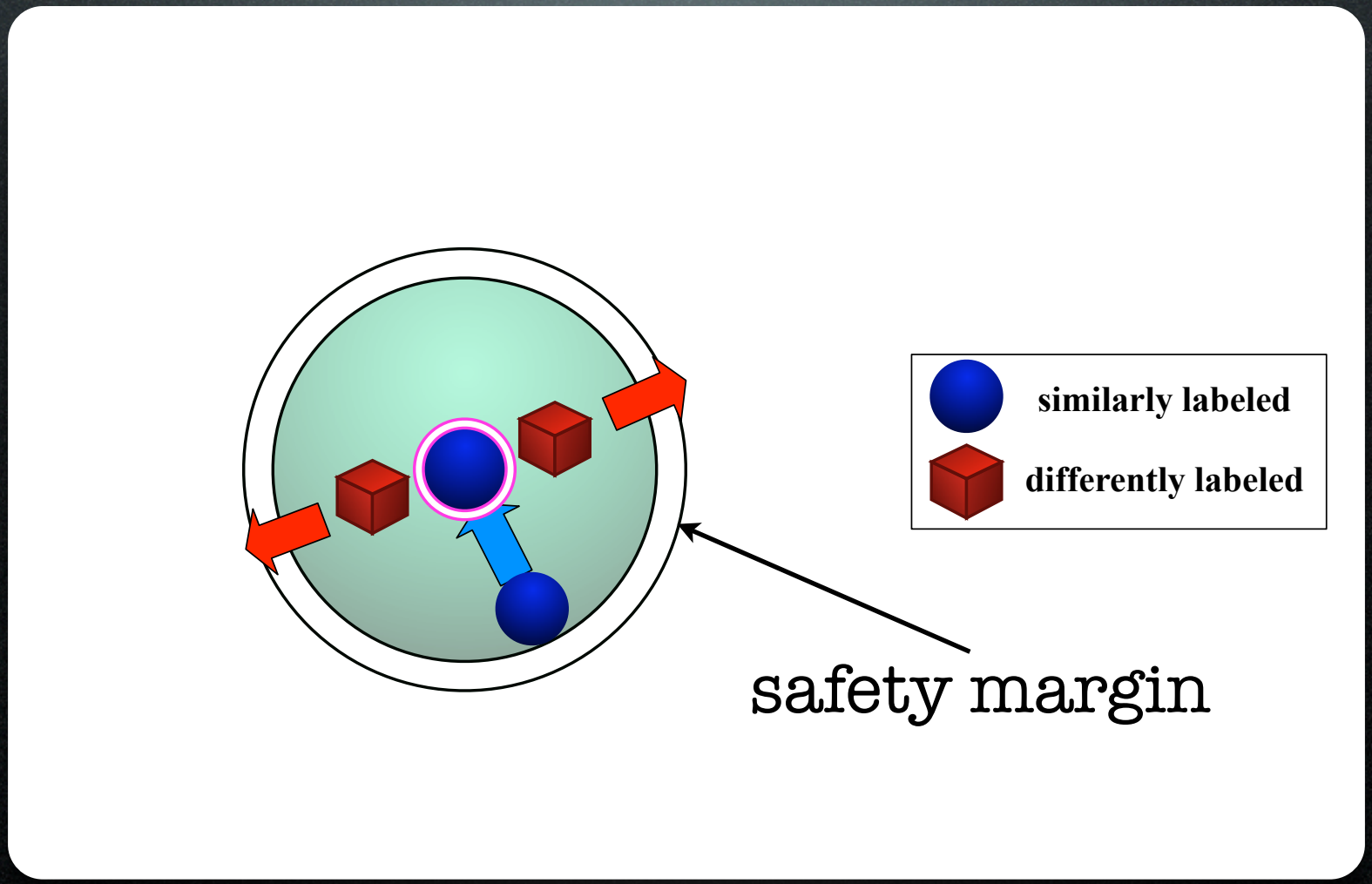
closest **similarly** labeled point
("target neighbor")

Intuition

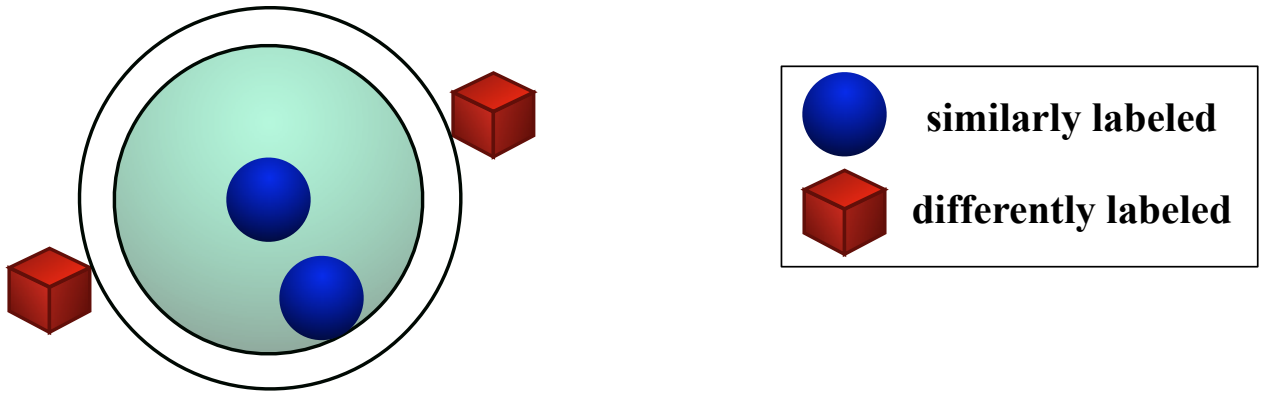


1. pull target neighbors closer
2. push impostors away

Intuition



Clean neighborhood!



Notation

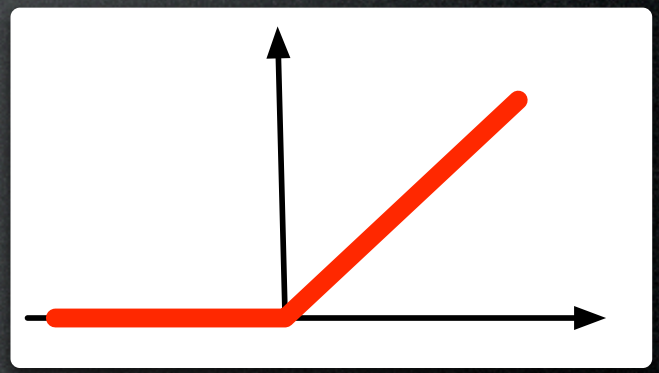
$\vec{x}_i$ input vectors

y_{ij} $\begin{cases} 1 \text{ if } \vec{x}_i \text{ and } \vec{x}_j\text{'s labels match} \\ 0 \text{ otherwise} \end{cases}$

η_{ij} $\begin{cases} 1 \text{ if } \vec{x}_j \text{ is ``target neighbor'' of } \vec{x}_i \\ 0 \text{ otherwise} \end{cases}$

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

(hinge loss)



Notation

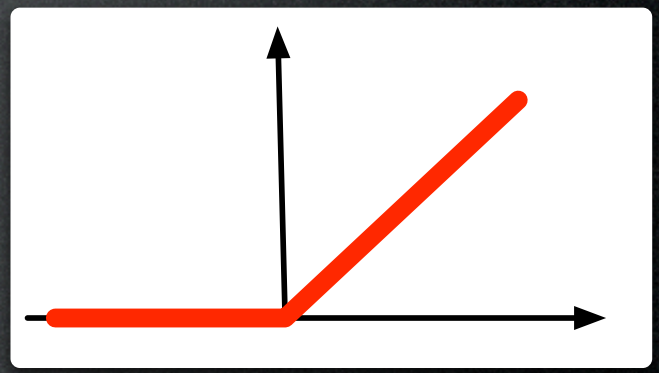
$\vec{x}_i$ input vectors

$$y_{ij} \begin{cases} 1 \text{ if } \vec{x}_i \text{ and } \vec{x}_j\text{'s labels match} \\ 0 \text{ otherwise} \end{cases}$$

$$\eta_{ij} \begin{cases} 1 \text{ if } \vec{x}_j \text{ is ``target neighbor'' of } \vec{x}_i \\ 0 \text{ otherwise} \end{cases}$$

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

(hinge loss)



Notation

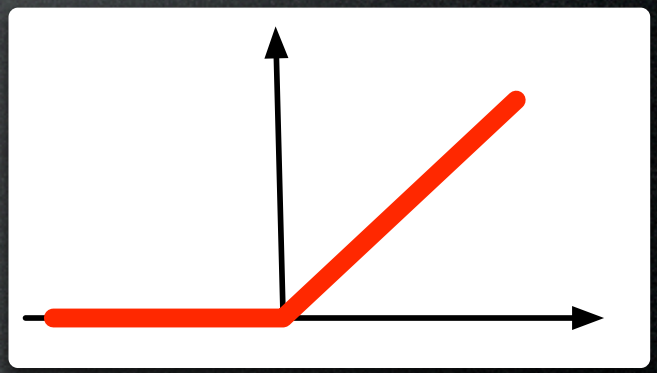
$\vec{x}_i$ input vectors

$$y_{ij} \begin{cases} 1 \text{ if } \vec{x}_i \text{ and } \vec{x}_j\text{'s labels match} \\ 0 \text{ otherwise} \end{cases}$$

$$\eta_{ij} \begin{cases} 1 \text{ if } \vec{x}_j \text{ is ``target neighbor'' of } \vec{x}_i \\ 0 \text{ otherwise} \end{cases}$$

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

(hinge loss)



Notation

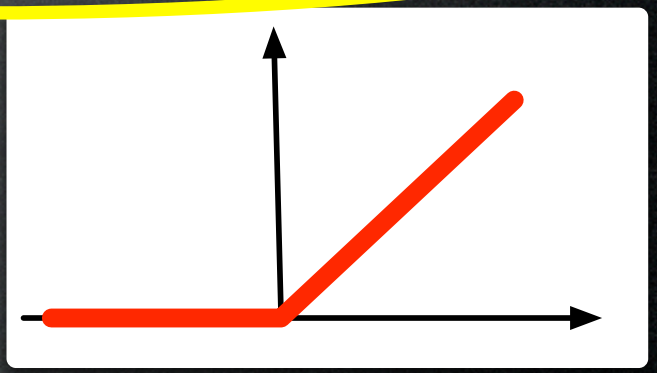
$\vec{x}_i$ input vectors

$$y_{ij} \begin{cases} 1 \text{ if } \vec{x}_i \text{ and } \vec{x}_j\text{'s labels match} \\ 0 \text{ otherwise} \end{cases}$$

$$\eta_{ij} \begin{cases} 1 \text{ if } \vec{x}_j \text{ is ``target neighbor'' of } \vec{x}_i \\ 0 \text{ otherwise} \end{cases}$$

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

(hinge loss)



Notation

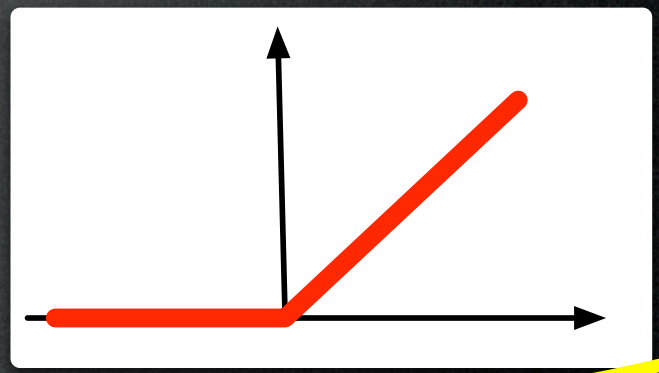
$\vec{x}_i$ input vectors

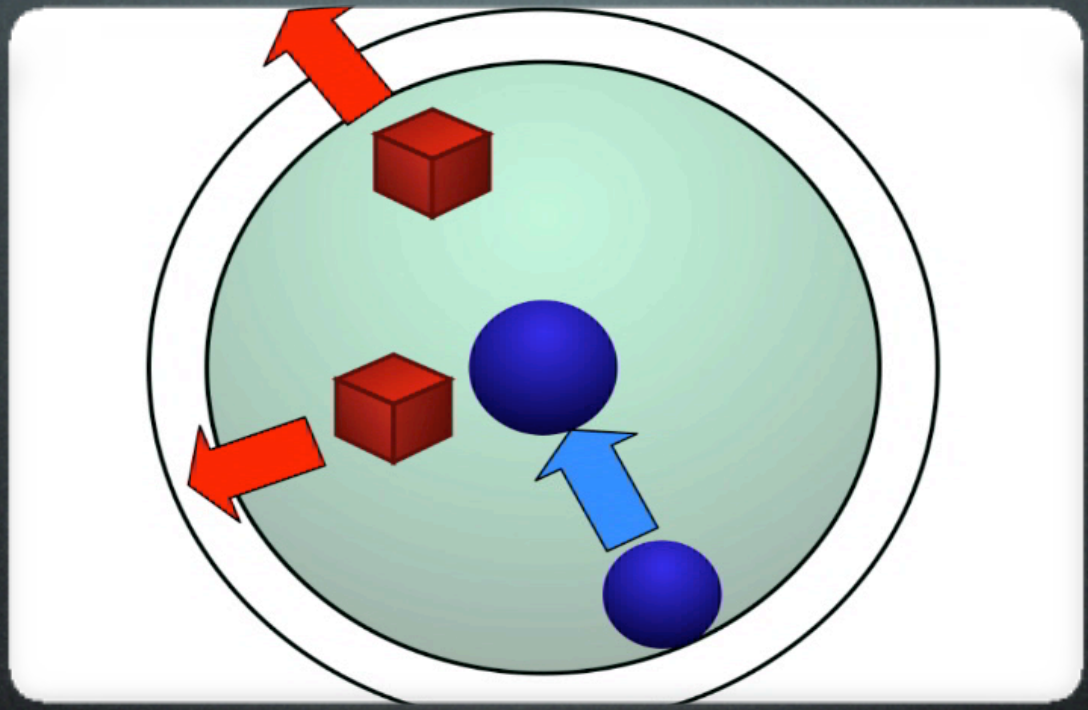
y_{ij} $\begin{cases} 1 \text{ if } \vec{x}_i \text{ and } \vec{x}_j\text{'s labels match} \\ 0 \text{ otherwise} \end{cases}$

η_{ij} $\begin{cases} 1 \text{ if } \vec{x}_j \text{ is ``target neighbor'' of } \vec{x}_i \\ 0 \text{ otherwise} \end{cases}$

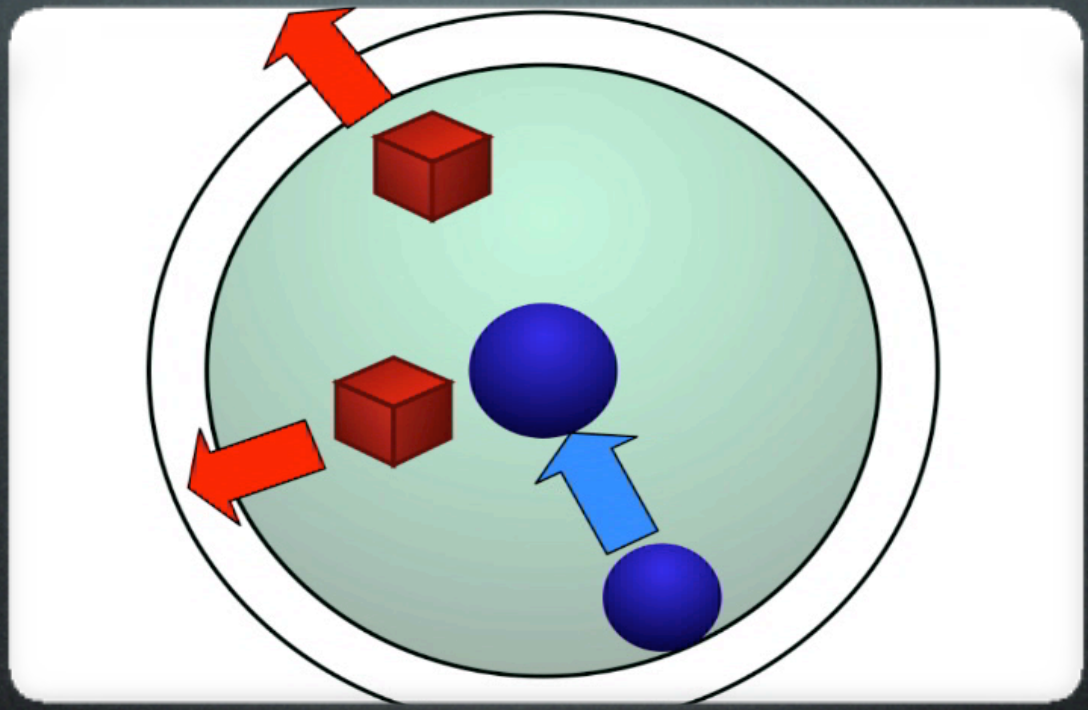
$[a]_+ = \max(a, 0)$

(hinge loss)



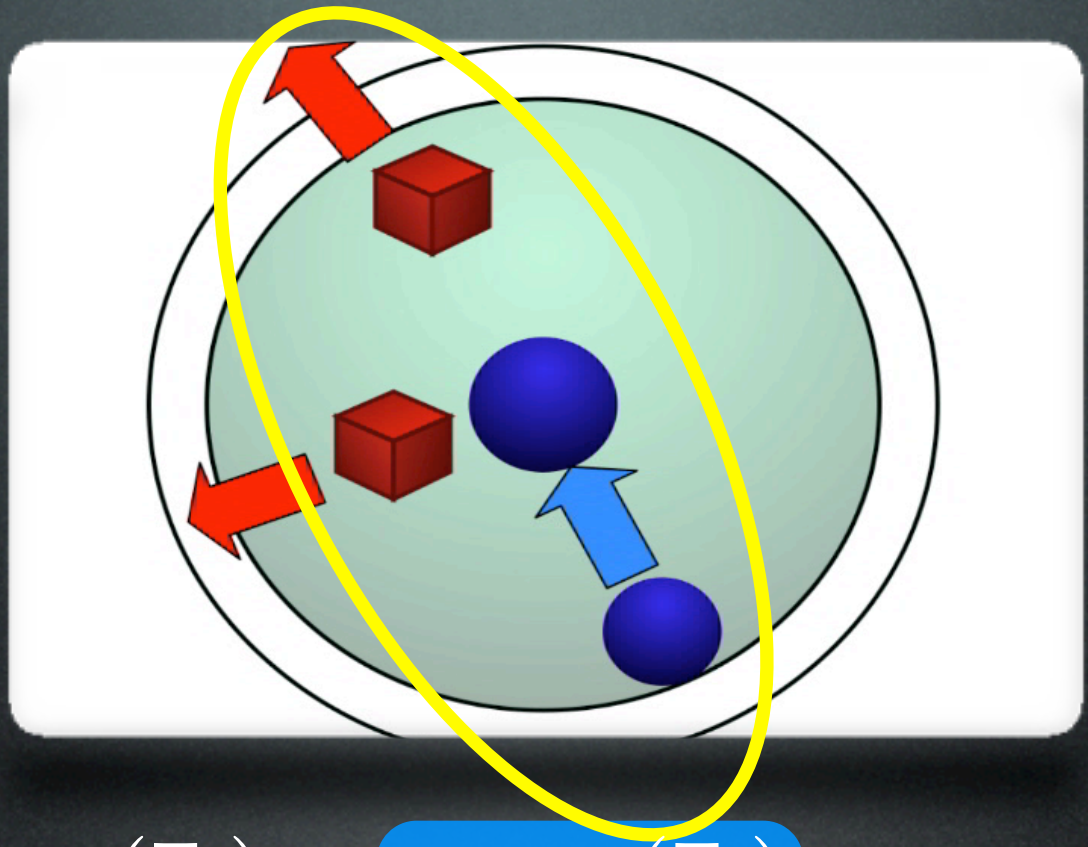


minimize: $\varepsilon(\mathbf{L}) = \varepsilon_{\text{pull}}(\mathbf{L}) + \varepsilon_{\text{push}}(\mathbf{L})$



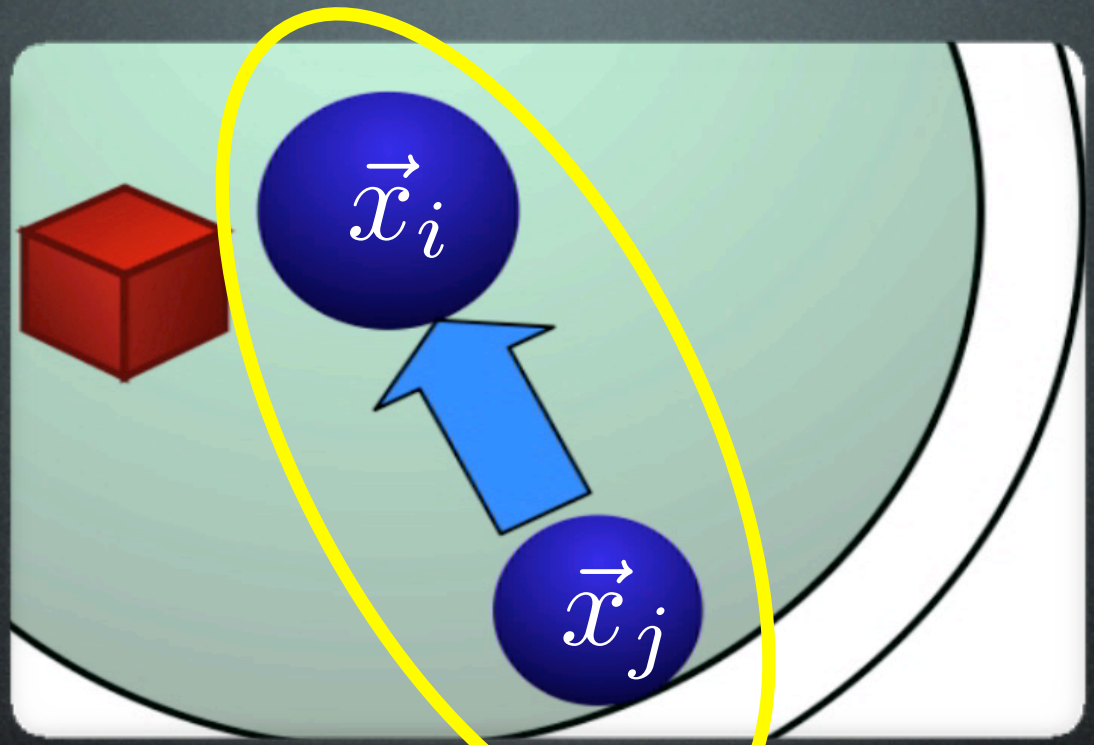
minimize: $\varepsilon(\mathbf{L}) = \varepsilon_{\text{pull}}(\mathbf{L}) + \varepsilon_{\text{push}}(\mathbf{L})$

$$\varepsilon_{\text{pull}}(\mathbf{L}) = \sum_{ij} \eta_{ij} \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|^2$$



minimize: $\varepsilon(\mathbf{L}) = \varepsilon_{\text{pull}}(\mathbf{L}) + \varepsilon_{\text{push}}(\mathbf{L})$

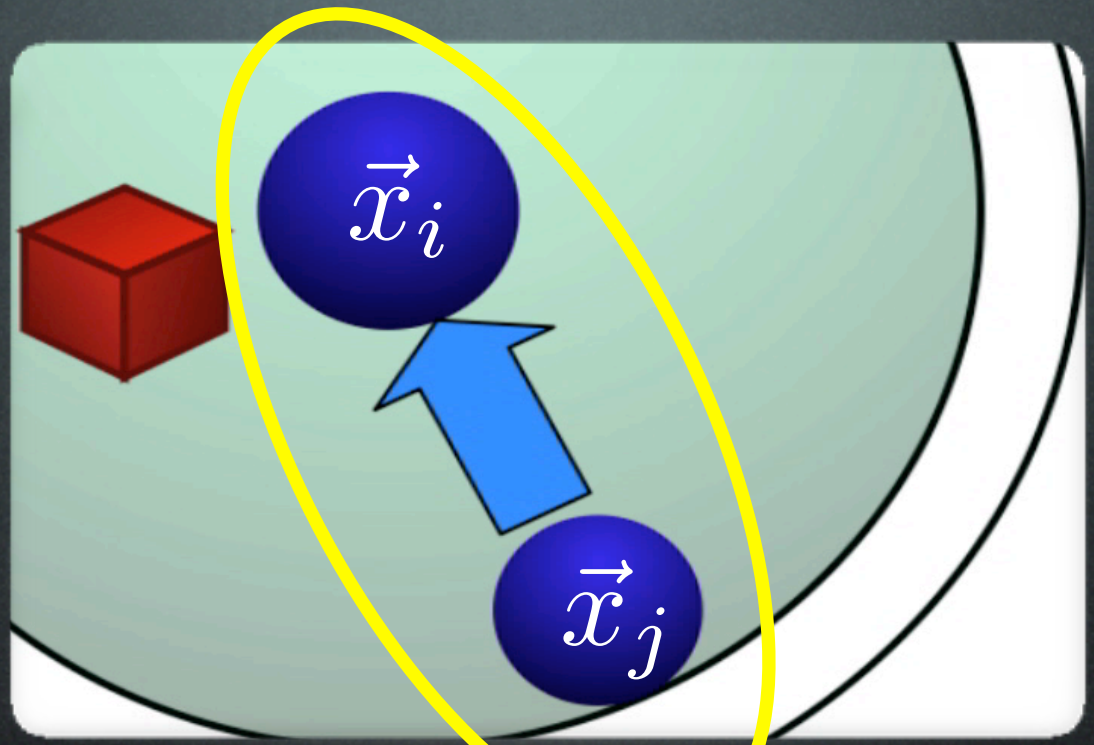
$$\varepsilon_{\text{pull}}(\mathbf{L}) = \sum_{ij} \eta_{ij} \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|^2$$



minimize: $\varepsilon(\mathbf{L}) = \varepsilon_{\text{pull}}(\mathbf{L}) + \varepsilon_{\text{push}}(\mathbf{L})$

$$\varepsilon_{\text{pull}}(\mathbf{L}) = \sum_{ij} \eta_{ij} \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|^2$$

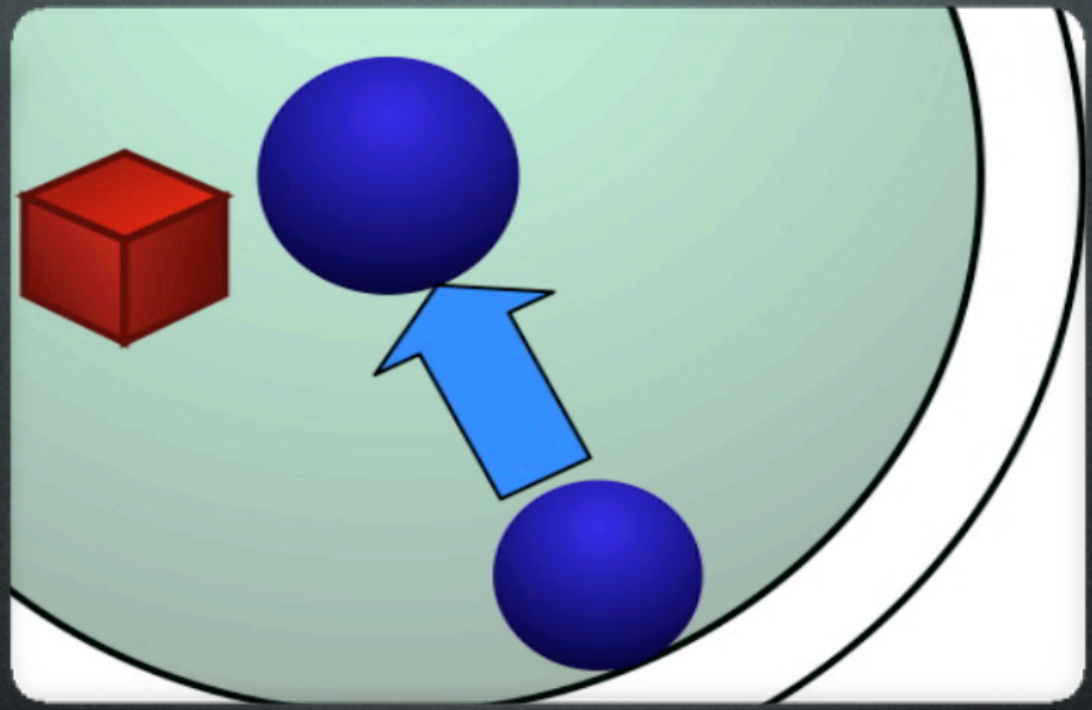
Neighborhood indicator



minimize: $\varepsilon(\mathbf{L}) = \varepsilon_{\text{pull}}(\mathbf{L}) + \varepsilon_{\text{push}}(\mathbf{L})$

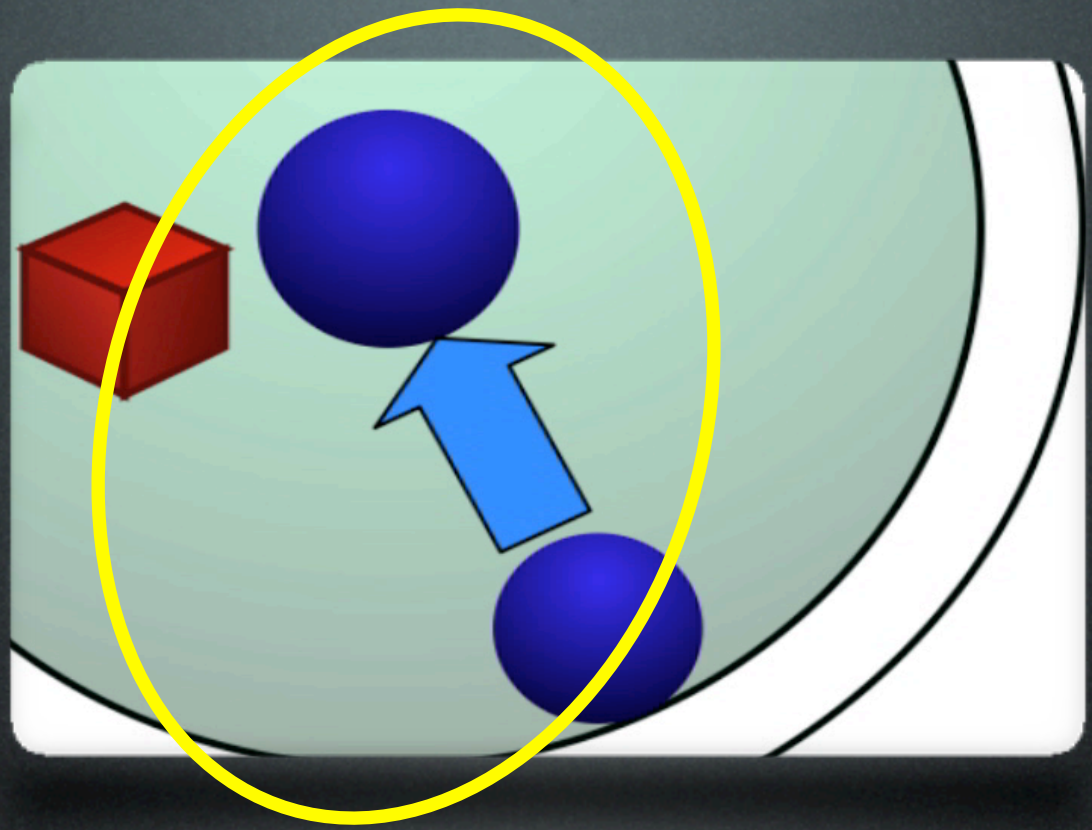
$$\varepsilon_{\text{pull}}(\mathbf{L}) = \sum_{ij} \eta_{ij} \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|^2$$

Distance to target neighbor



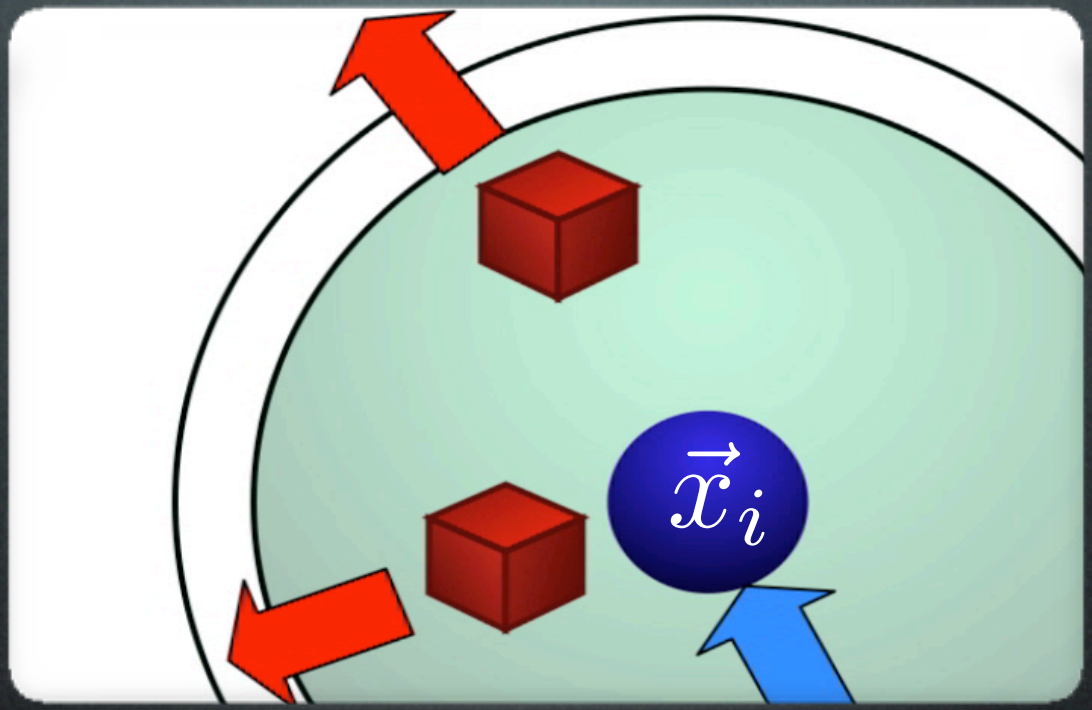
minimize: $\varepsilon(\mathbf{L}) = \varepsilon_{\text{pull}}(\mathbf{L}) + \varepsilon_{\text{push}}(\mathbf{L})$

$$\varepsilon_{\text{push}}(\mathbf{L}) = \sum_{ijl} \eta_{ij}(1-y_{il}) \left[1 + \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|^2 - \|\mathbf{L}(\vec{x}_i - \vec{x}_l)\|^2\right]_+$$



minimize: $\varepsilon(\mathbf{L}) = \varepsilon_{\text{pull}}(\mathbf{L}) + \varepsilon_{\text{push}}(\mathbf{L})$

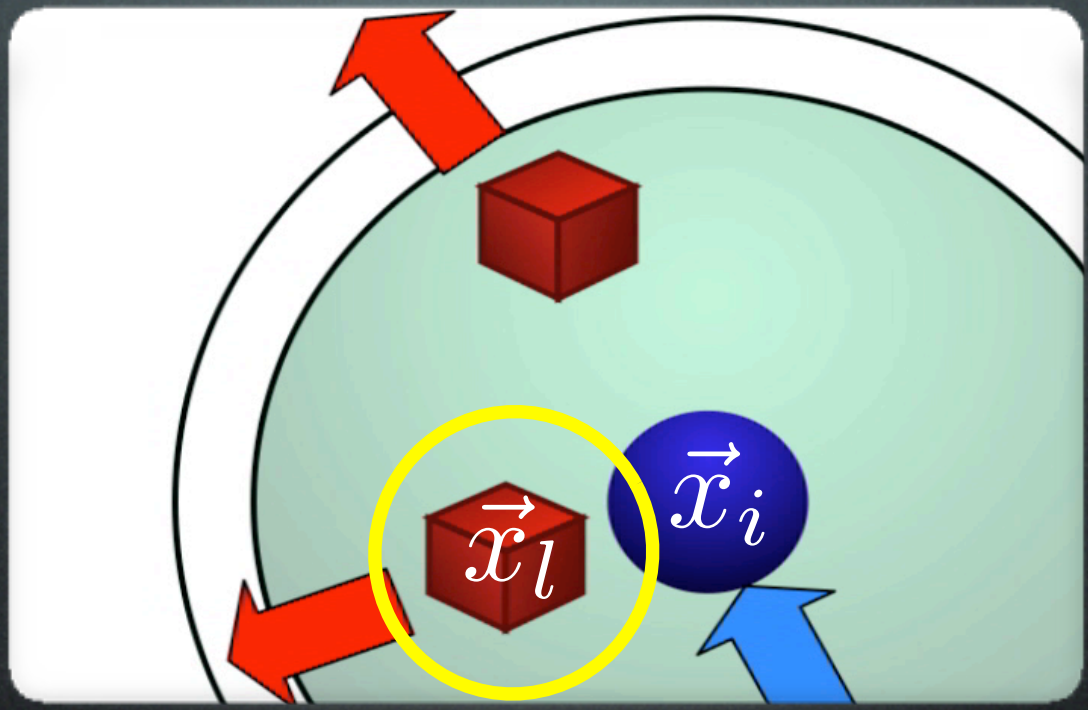
$$\varepsilon_{\text{push}}(\mathbf{L}) = \sum_{ijl} \eta_{ij}(1-y_{il}) \left[1 + \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|^2 - \|\mathbf{L}(\vec{x}_i - \vec{x}_l)\|^2\right]_+$$



minimize: $\varepsilon(\mathbf{L}) = \varepsilon_{\text{pull}}(\mathbf{L}) + \varepsilon_{\text{push}}(\mathbf{L})$

$$\varepsilon_{\text{push}}(\mathbf{L}) = \sum_{ijl} \eta_{ij}(1-y_{il}) \left[1 + \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|^2 - \|\mathbf{L}(\vec{x}_i - \vec{x}_l)\|^2\right]_+$$

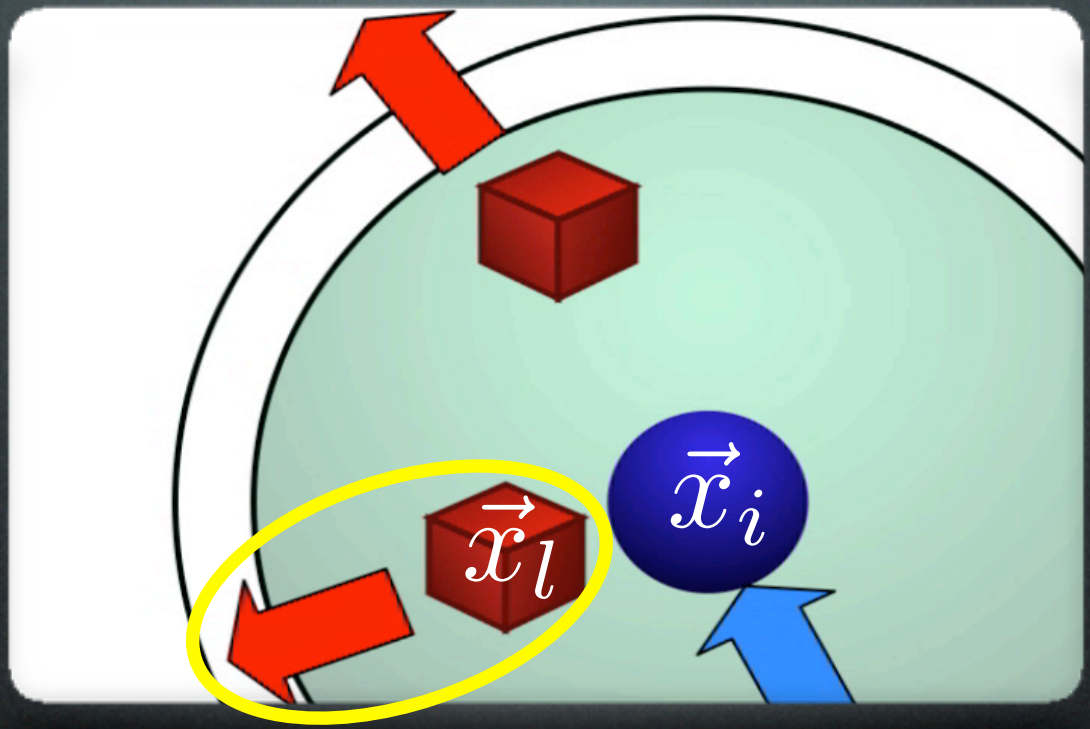
neighborhood indicator



minimize: $\varepsilon(\mathbf{L}) = \varepsilon_{\text{pull}}(\mathbf{L}) + \varepsilon_{\text{push}}(\mathbf{L})$

$$\varepsilon_{\text{push}}(\mathbf{L}) = \sum_{ijl} \eta_{ij} (1 - y_{il}) \left[1 + \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|^2 - \|\mathbf{L}(\vec{x}_i - \vec{x}_l)\|^2 \right]_+$$

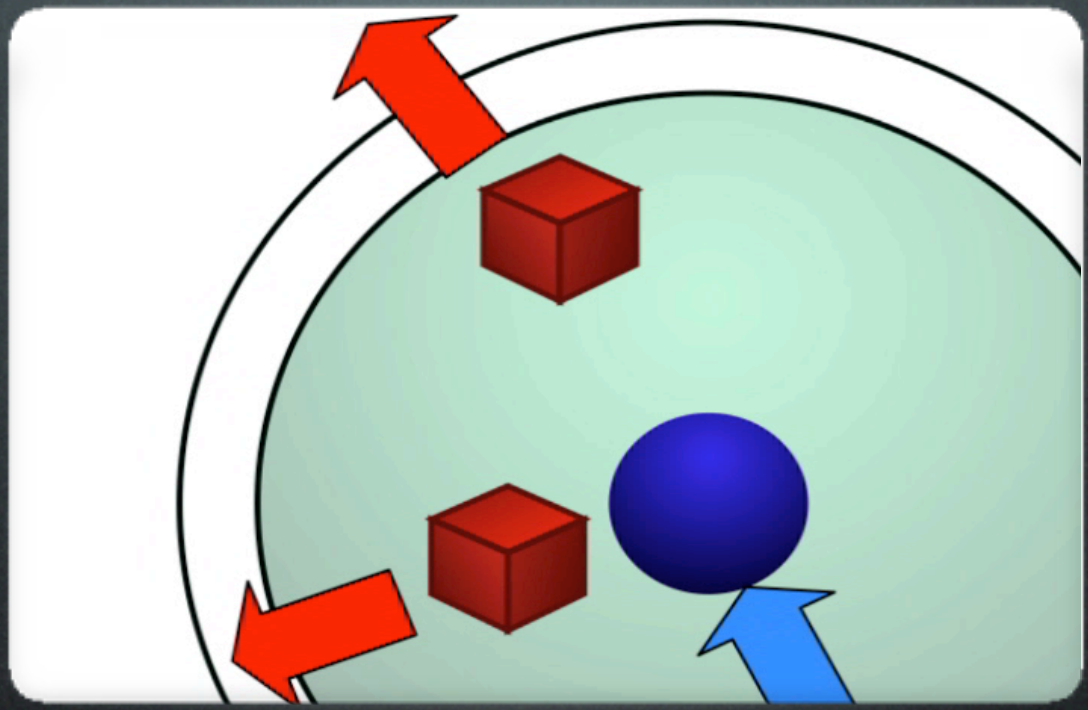
impostor



minimize: $\varepsilon(\mathbf{L}) = \varepsilon_{\text{pull}}(\mathbf{L}) + \varepsilon_{\text{push}}(\mathbf{L})$

$$\varepsilon_{\text{push}}(\mathbf{L}) = \sum_{ijl} \eta_{ij}(1-y_{il}) \left[1 + \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|^2 - \|\mathbf{L}(\vec{x}_i - \vec{x}_l)\|^2\right]_+$$

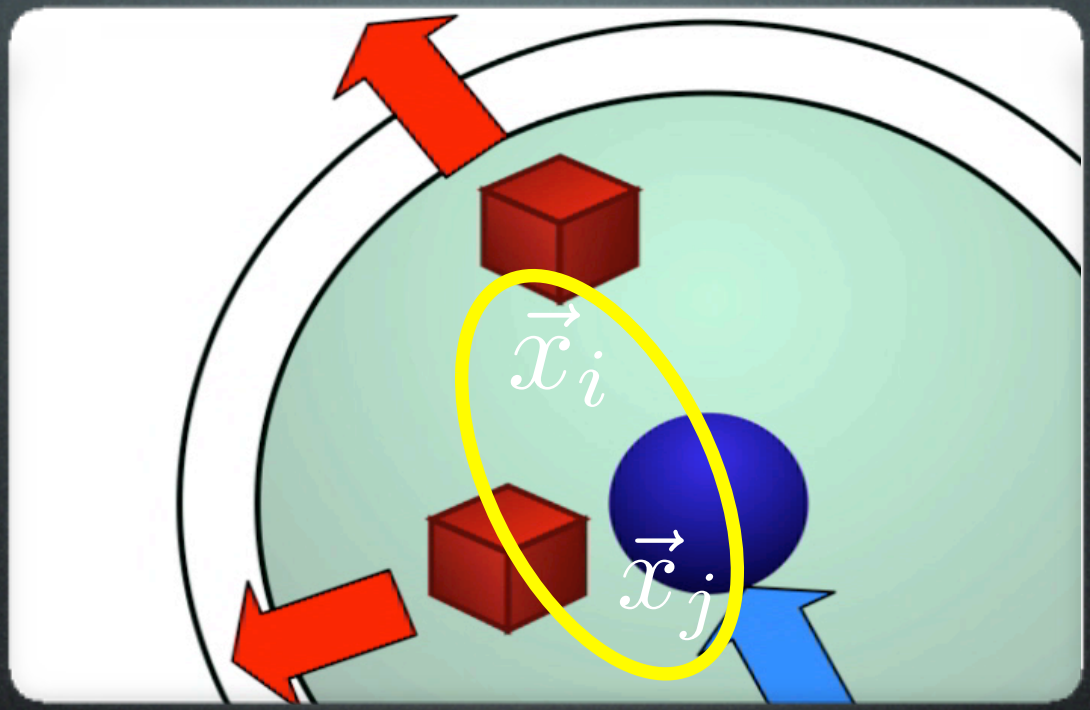
hinge loss



minimize: $\varepsilon(\mathbf{L}) = \varepsilon_{\text{pull}}(\mathbf{L}) + \varepsilon_{\text{push}}(\mathbf{L})$

$$\varepsilon_{\text{push}}(\mathbf{L}) = \sum_{ijl} \eta_{ij}(1-y_{il}) \left[1 + \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|^2 - \|\mathbf{L}(\vec{x}_i - \vec{x}_l)\|^2\right]_+$$

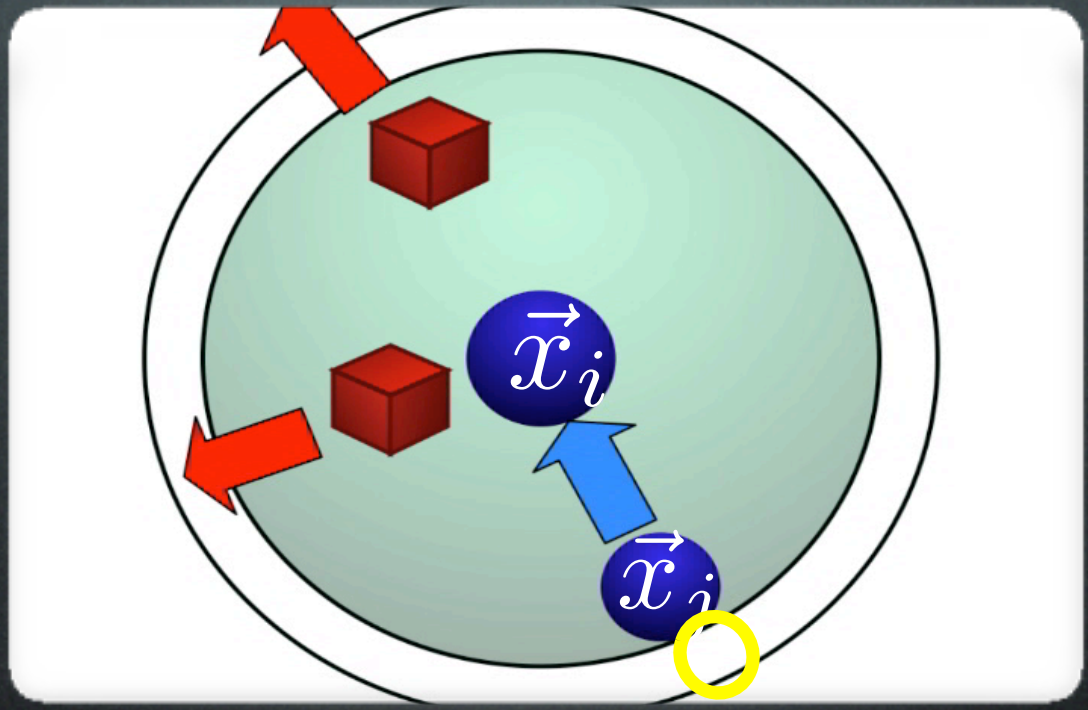
neighborhood radius



minimize: $\varepsilon(\mathbf{L}) = \varepsilon_{\text{pull}}(\mathbf{L}) + \varepsilon_{\text{push}}(\mathbf{L})$

$$\varepsilon_{\text{push}}(\mathbf{L}) = \sum_{ijl} \eta_{ij}(1-y_{il}) \left[1 + \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|^2 - \|\mathbf{L}(\vec{x}_i - \vec{x}_l)\|^2\right]_+$$

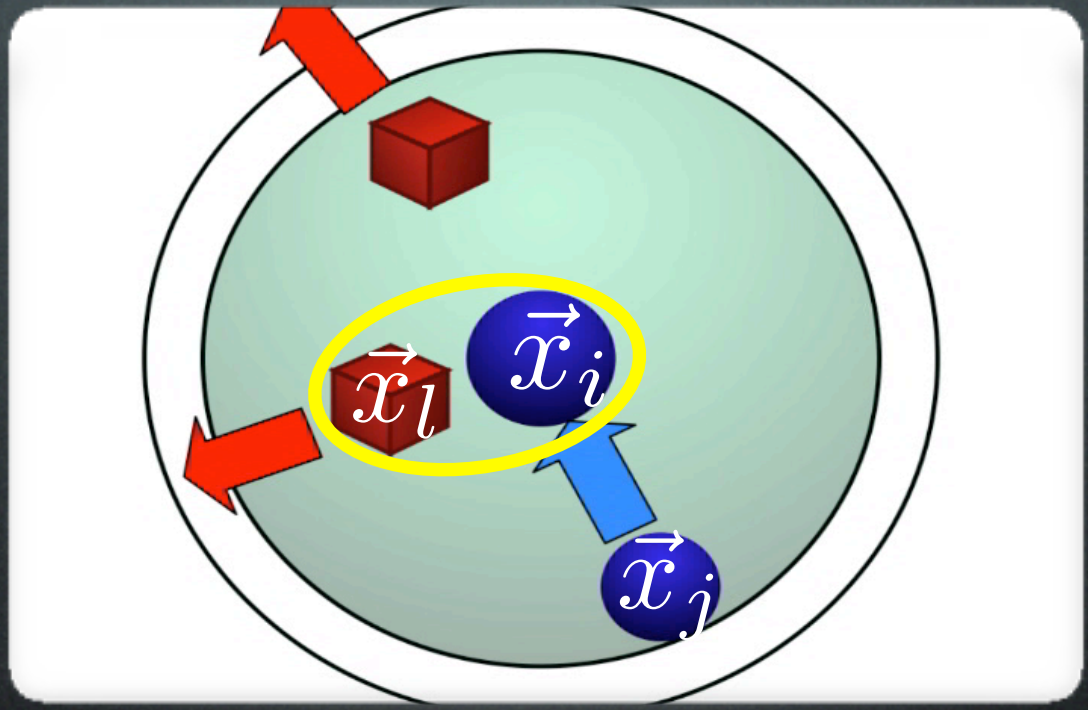
neighborhood radius



minimize: $\varepsilon(\mathbf{L}) = \varepsilon_{\text{pull}}(\mathbf{L}) + \varepsilon_{\text{push}}(\mathbf{L})$

$$\varepsilon_{\text{push}}(\mathbf{L}) = \sum_{ijl} \eta_{ij}(1-y_{il}) \left[1 + \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|^2 - \|\mathbf{L}(\vec{x}_i - \vec{x}_l)\|^2\right]_+$$

safety margin



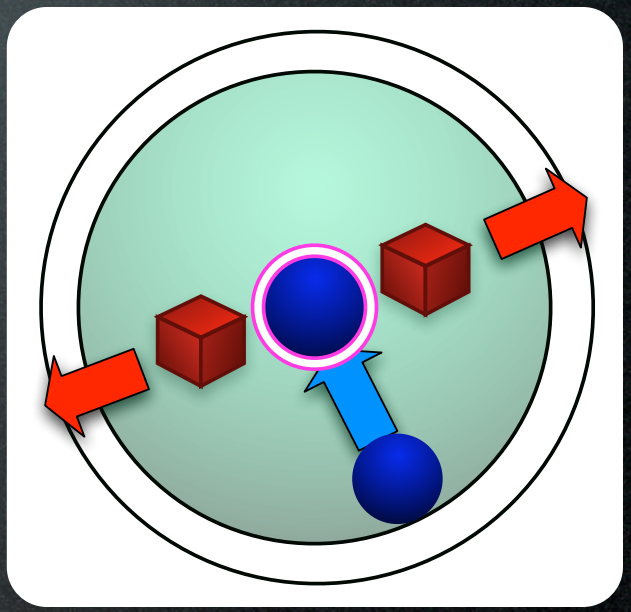
minimize: $\varepsilon(\mathbf{L}) = \varepsilon_{\text{pull}}(\mathbf{L}) + \varepsilon_{\text{push}}(\mathbf{L})$

$$\varepsilon_{\text{push}}(\mathbf{L}) = \sum_{ijl} \eta_{ij}(1-y_{il}) \left[1 + \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|^2 - \|\mathbf{L}(\vec{x}_i - \vec{x}_l)\|^2\right]_+$$

distance to impostor

Loss function

1. pull target neighbors closer
2. push impostors away



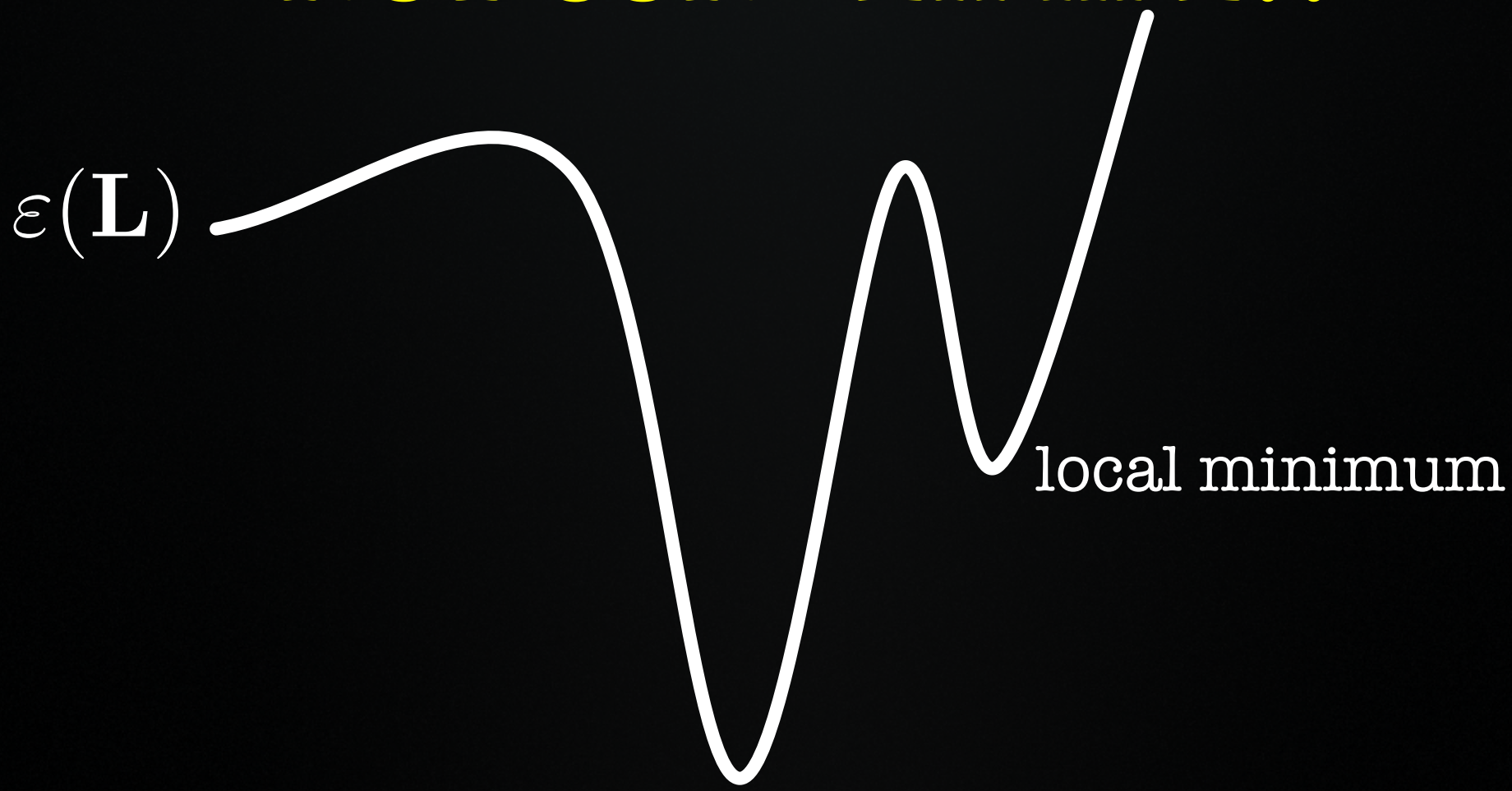
$$\varepsilon(\mathbf{L}) = \varepsilon_{\text{pull}}(\mathbf{L}) + \varepsilon_{\text{push}}(\mathbf{L})$$

$$\varepsilon_{\text{pull}}(\mathbf{L}) = \sum_{ij} \eta_{ij} \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|^2$$

$$\varepsilon_{\text{push}}(\mathbf{L}) = \sum_{ijl} \eta_{ij}(1 - y_{il}) \left[1 + \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|^2 - \|\mathbf{L}(\vec{x}_i - \vec{x}_l)\|^2\right]_+$$

(unit margin)

NOT CONVEX in L!!



Change of variable

Instead of learning $\mathbf{L}$ we learn $\mathbf{M} = \mathbf{L}^\top \mathbf{L}$.

with $\mathbf{M} \succeq 0$

$$\begin{aligned}\|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|^2 &= (\vec{x}_i - \vec{x}_j)^\top \mathbf{L}^\top \mathbf{L}(\vec{x}_i - \vec{x}_j) \\ &= (\vec{x}_i - \vec{x}_j)^\top \mathbf{M}(\vec{x}_i - \vec{x}_j) \\ &= \|\vec{x}_i - \vec{x}_j\|^2_{\mathbf{M}}\end{aligned}$$

M defines Mahalanobis distance

Loss function is convex in M!

Convex optimization

minimize:

$$\underset{\mathrm{M}}{\text{minimize}}: \quad \varepsilon(\mathrm{M}) = \varepsilon_{\mathrm{pull}}(\mathrm{M}) + \varepsilon_{\mathrm{push}}(\mathrm{M})$$

subject to:

$$\mathrm{M} \succeq 0$$

Semidefinite Programming Problem

minimize:
M

$$\varepsilon(\mathrm{M}) = \varepsilon_{\mathrm{pull}}(\mathrm{M}) + \varepsilon_{\mathrm{push}}(\mathrm{M})$$

subject to:

$$\mathrm{M} \succeq 0$$

Linear programming problem with positive-semidefinite constraint.

Semidefinite Programming Problem

minimize:
M

$$\varepsilon(\mathrm{M}) = \varepsilon_{\mathrm{pull}}(\mathrm{M}) + \varepsilon_{\mathrm{push}}(\mathrm{M})$$

subject to:

$$\mathrm{M} \succeq 0$$

Linear programming problem with positive-semidefinite constraint.

Semidefinite Programmin[illegible]m

Can be solved efficiently with alternating projection or cutting plane algorithm.

$$\succeq 0$$

Line[illegible]rogramming problem with positive-semidefinite constraint.

Two ways of classification

k-NN using Mahalanobis distance

1. Learn M that minimizes $\varepsilon(\mathrm{M})$
2. Use k-NN with Mahalanobis distance

Two ways of classification

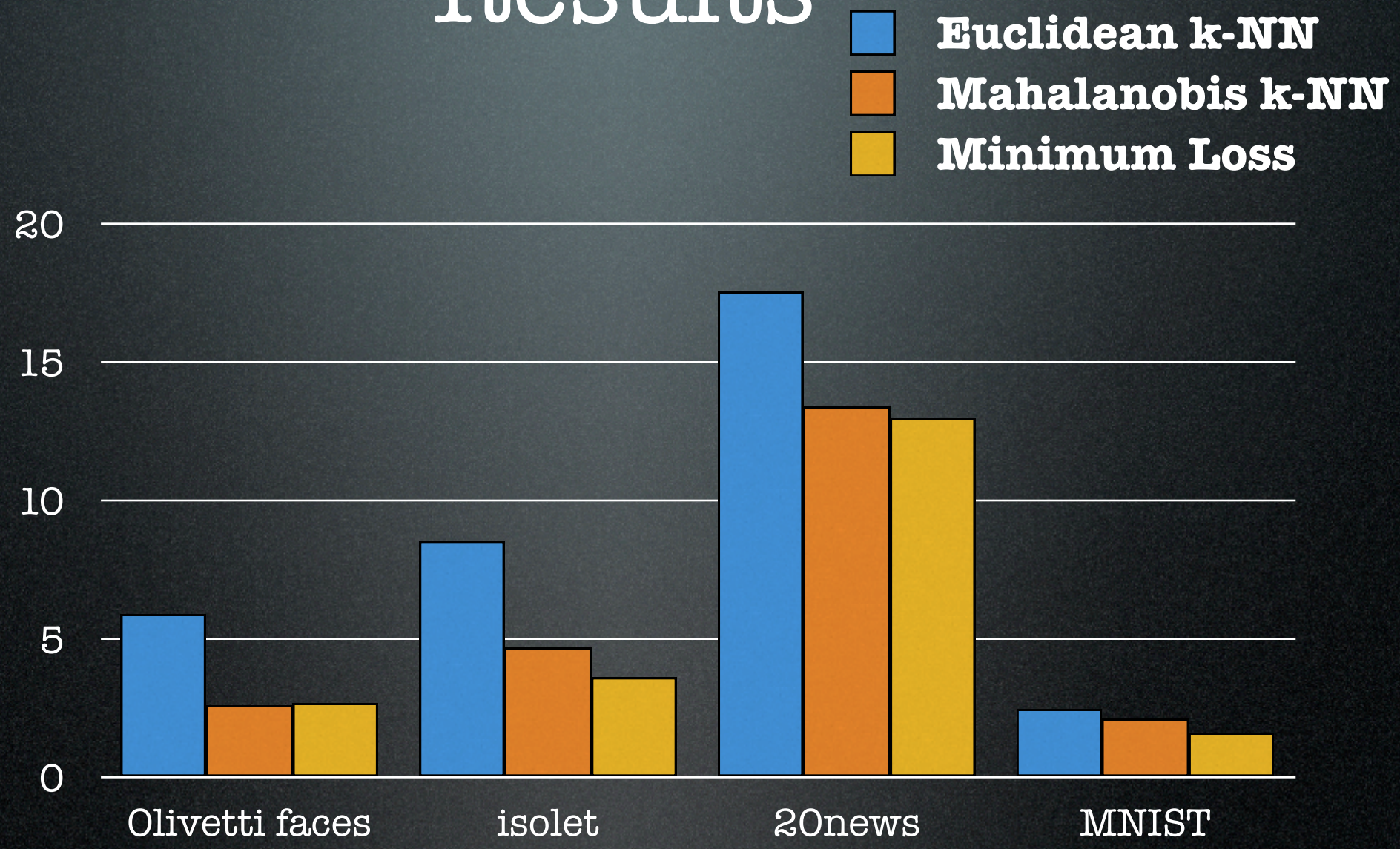
k-NN using Mahalanobis distance

1. Learn $\mathbf{M}$ that minimizes $\varepsilon(\mathbf{M})$
2. Use k-NN with Mahalanobis distance

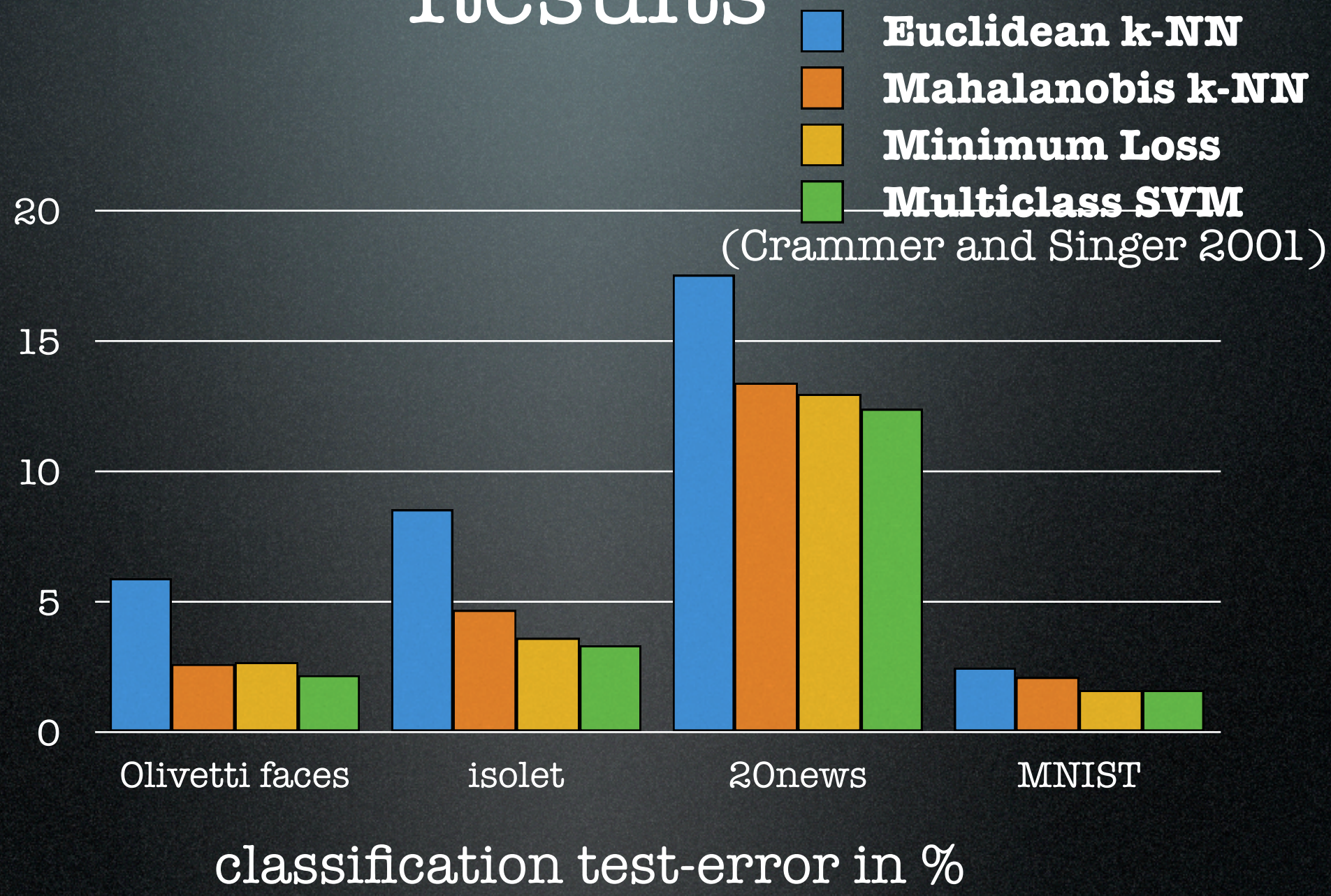
Minimum loss classification

1. Learn $\mathbf{M}$ that minimizes $\varepsilon(\mathbf{M})$
2. For each test point choose label that minimizes the loss function

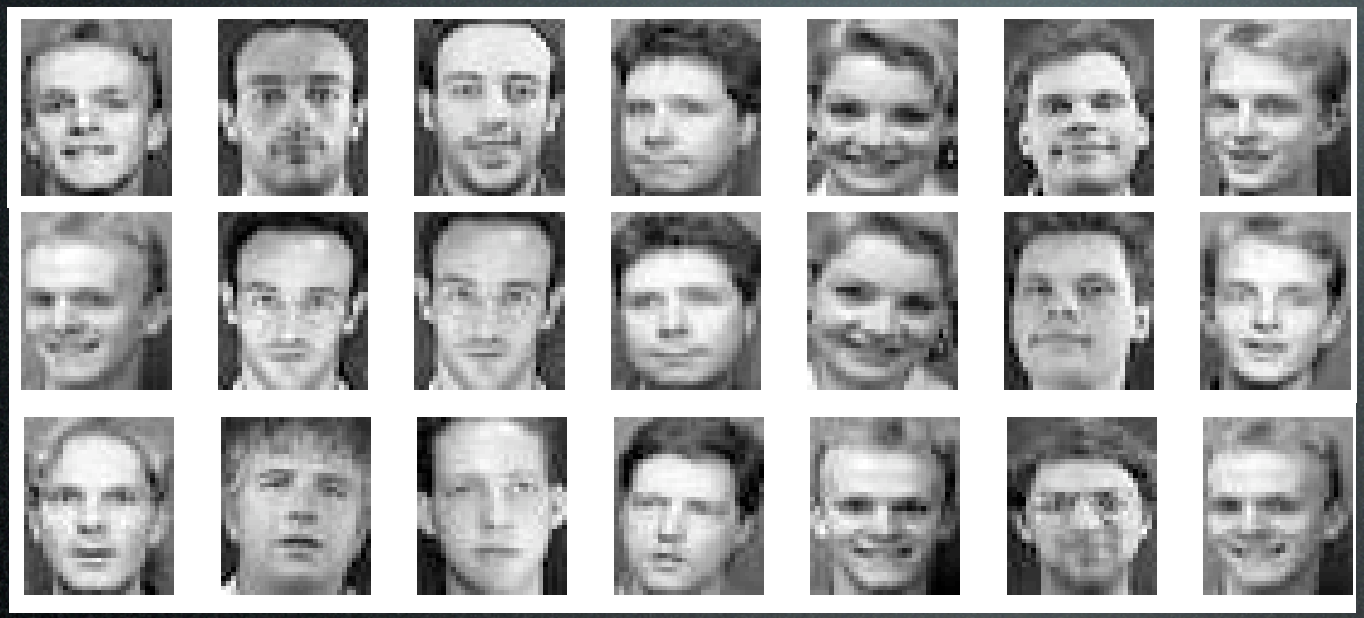
Results


Euclidean k-NN
Mahalanobis k-NN
Minimum Loss
20
15
10
5
0
Olivetti faces
isolet
20news
MNIST
classification test error in %

Results


Euclidean k-NN
Mahalanobis k-NN
Minimum Loss
Multiclass SVM
(Crammer and Singer 2001)
20
15
10
5
0
Olivetti faces
isolet
20news
MNIST
classification test-error in %

Olivetti Faces



test image

Mahalanobis NN

Euclidean NN

Train / Test images	**= 280/120**
Total constraints	**= 76 440**
Active constraints	**= 2 680**
Training time	**= 2 mins**

MNIST

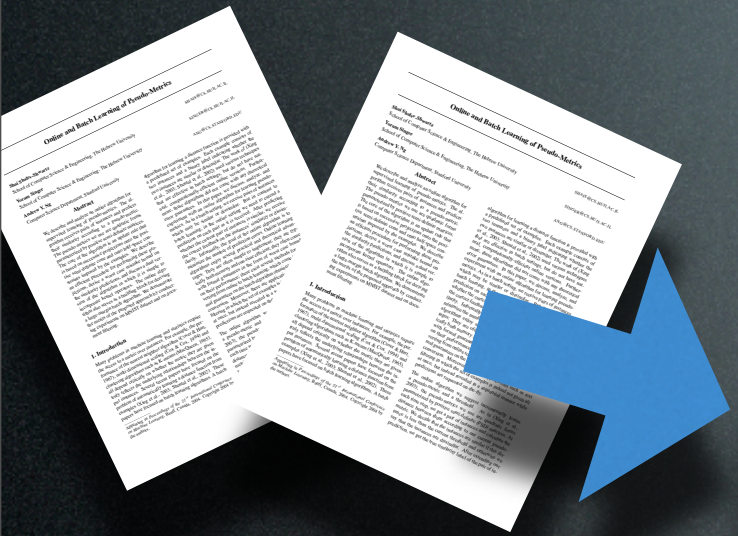
test digit

Mahalanobis NN

Euclidean NN

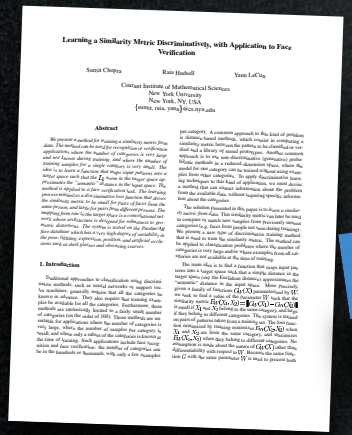
Train / Test images	=	60k/10k
Total constraints	=	3.3 Billion
Active constraints	=	243 596
Training time	=	3 hours

Connection to previous work

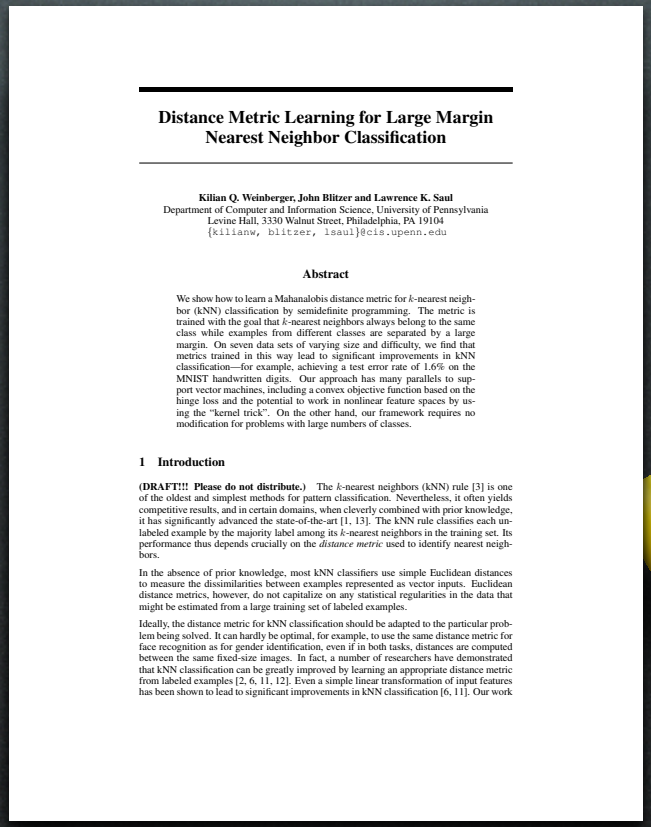


Xing et al, NIPS 2003

Shalev-Shwartz et al, ICML 2004



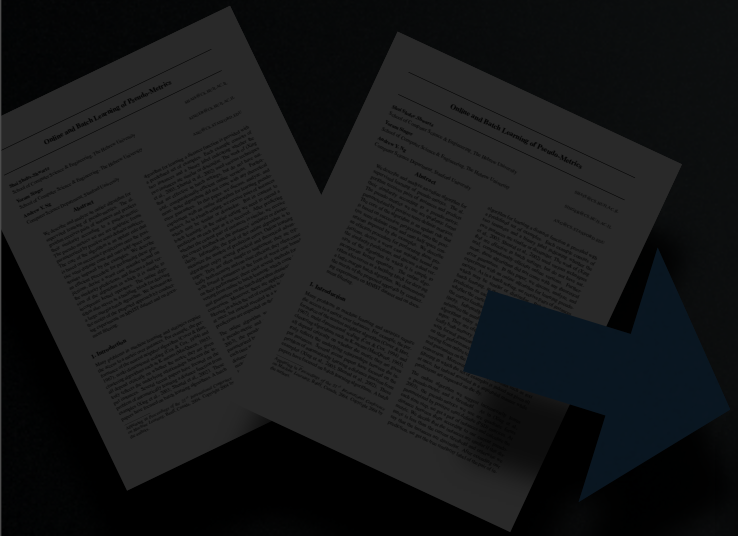
Chopra et al, CVPR 2005



Goldberger et al, NIPS 2005

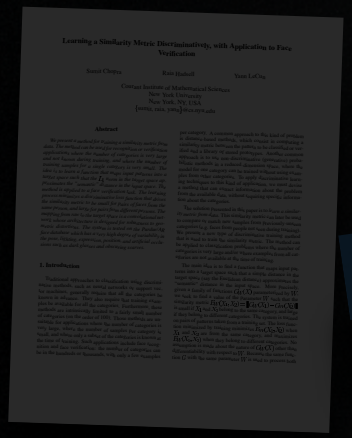
Vapnik 1998

Connection to previous work

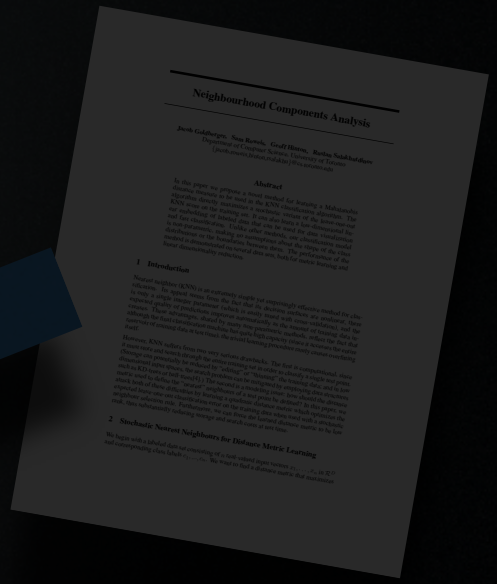


Xing et al, NIPS 2003

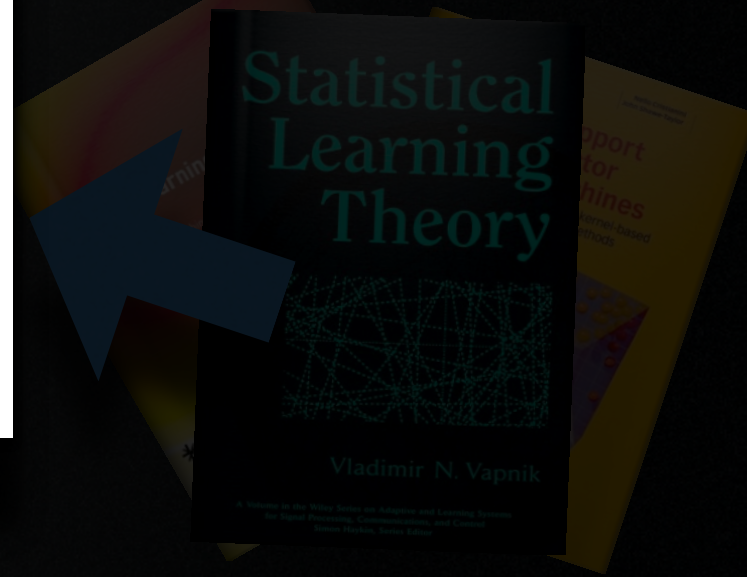
Shalev-Shwartz et al, ICML 2004



Chopra et al, CVPR 2005



Goldberger et al, NIPS 2005



Vapnik 1998

Distance Metric Learning for Large Margin Nearest Neighbor Classification

Kilian Q. Weinberger, John Blitzer and Lawrence K. Saul
Department of Computer and Information Science, University of Pennsylvania
Levine Hall, 3330 Walnut Street, Philadelphia, PA 19104
{kilianw, blitzer, lsaul}@cis.upenn.edu

Abstract

We show how to learn a Mahanalobis distance metric for k-nearest neighbor (kNN) classification by semidefinite programming. The metric is trained with the goal that k-nearest neighbors always belong to the same class while examples from different classes are separated by a large margin. On seven data sets of varying size and difficulty, we find that metrics trained in this way lead to significant improvements in kNN classification—for example, achieving a test error rate of 1.6% on the MNIST handwritten digits. Our approach has many parallels to support vector machines, including a convex objective function based on the hinge loss and the potential to work in nonlinear feature spaces by using the "kernel trick". On the other hand, our framework requires no modification for problems with large numbers of classes.

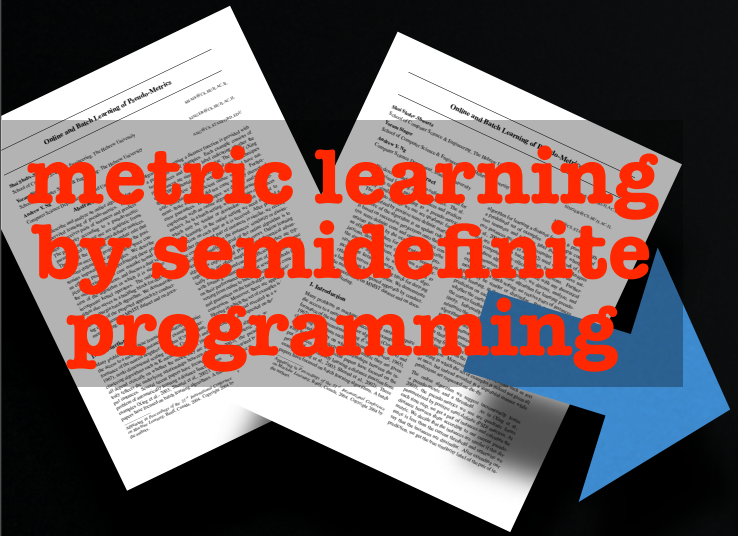
1 Introduction

(DRAFT!!! Please do not distribute.) The k-nearest neighbors (kNN) rule [3] is one of the oldest and simplest methods for pattern classification. Nevertheless, it often yields competitive results, and in certain domains, when cleverly combined with prior knowledge, it has significantly advanced the state-of-the-art [1, 13]. The kNN rule classifies each unlabeled example by the majority label among its k-nearest neighbors in the training set. Its performance thus depends crucially on the *distance metric* used to identify nearest neighbors.

In the absence of prior knowledge, most kNN classifiers use simple Euclidean distances to measure the dissimilarities between examples represented as vector inputs. Euclidean distance metrics, however, do not capitalize on any statistical regularities in the data that might be estimated from a large training set of labeled examples.

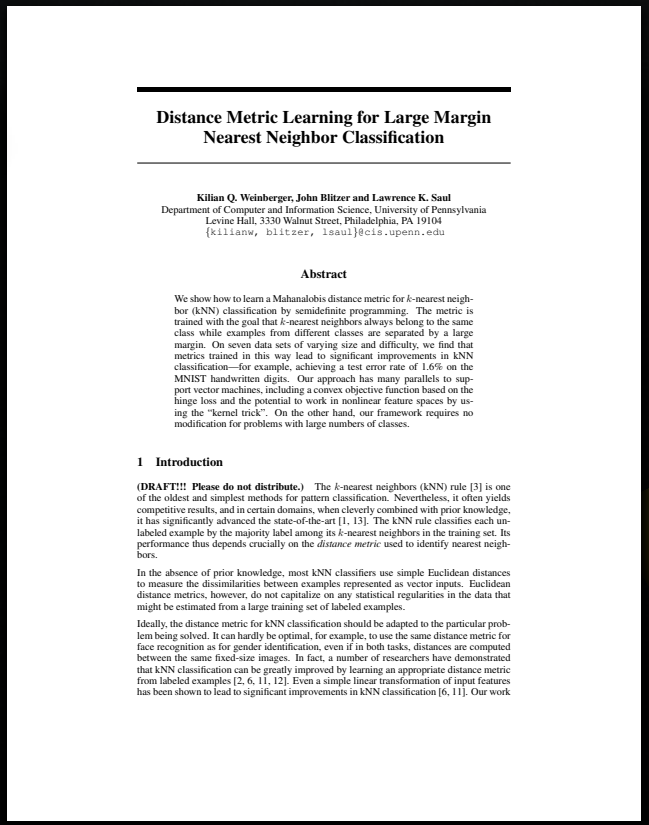
Ideally, the distance metric for kNN classification should be adapted to the particular problem being solved. It can hardly be optimal, for example, to use the same distance metric for face recognition as for gender identification, even if in both tasks, distances are computed between the same fixed-size images. In fact, a number of researchers have demonstrated that kNN classification can be greatly improved by learning an appropriate distance metric from labeled examples [2, 6, 11, 12]. Even a simple linear transformation of input features has been shown to lead to significant improvements in kNN classification [6, 11]. Our work

Connection to previous work



Xing et al, NIPS 2003

Shalev-Shwartz et al, ICML 2004



Distance Metric Learning for Large Margin Nearest Neighbor Classification

Kilian Q. Weinberger, John Blitzer and Lawrence K. Saul
Department of Computer and Information Science, University of Pennsylvania
Levine Hall, 3330 Walnut Street, Philadelphia, PA 19104
{kilianw, blitzer, lsaul}@cis.upenn.edu

Abstract

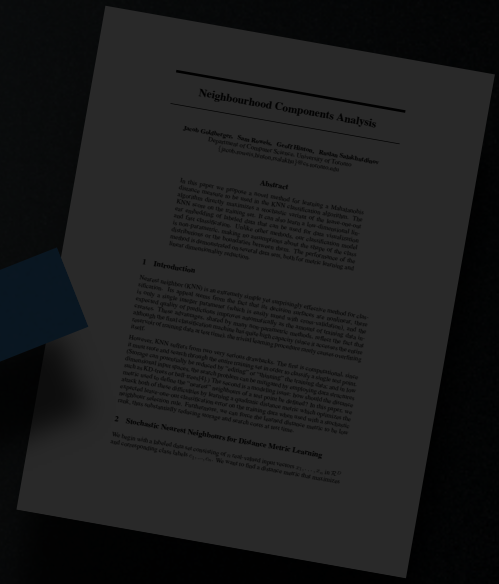
We show how to learn a Mahanalobis distance metric for k-nearest neighbor (kNN) classification by semidefinite programming. The metric is trained with the goal that k-nearest neighbors always belong to the same class while examples from different classes are separated by a large margin. On seven data sets of varying size and difficulty, we find that metrics trained in this way lead to significant improvements in kNN classification—for example, achieving a test error rate of 1.6% on the MNIST handwritten digits. Our approach has many parallels to support vector machines, including a convex objective function based on the hinge loss and the potential to work in nonlinear feature spaces by using the "kernel trick". On the other hand, our framework requires no modification for problems with large numbers of classes.

1 Introduction

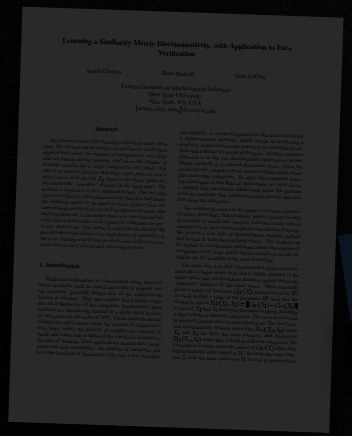
(DRAFT!!! Please do not distribute.) The k-nearest neighbors (kNN) rule [3] is one of the oldest and simplest methods for pattern classification. Nevertheless, it often yields competitive results, and in certain domains, when cleverly combined with prior knowledge, it has significantly advanced the state-of-the-art [1, 13]. The kNN rule classifies each unlabeled example by the majority label among its k-nearest neighbors in the training set. Its performance thus depends crucially on the *distance metric* used to identify nearest neighbors.

In the absence of prior knowledge, most kNN classifiers use simple Euclidean distances to measure the dissimilarities between examples represented as vector inputs. Euclidean distance metrics, however, do not capitalize on any statistical regularities in the data that might be estimated from a large training set of labeled examples.

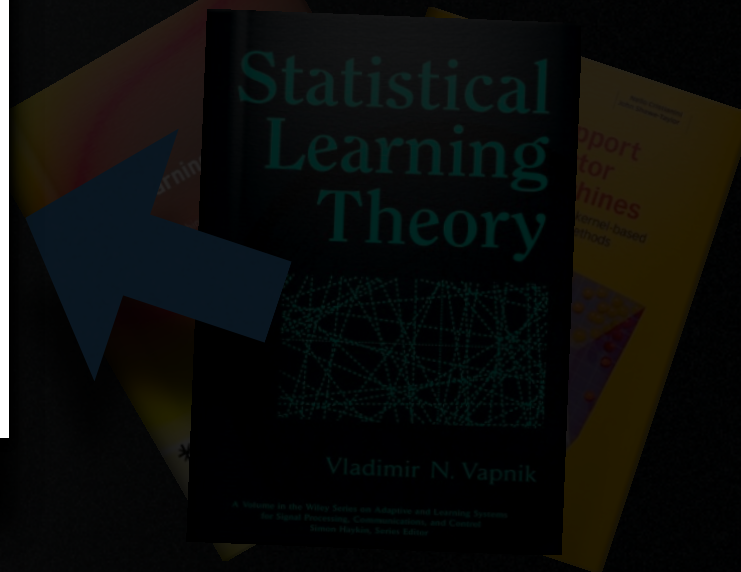
Ideally, the distance metric for kNN classification should be adapted to the particular problem being solved. It can hardly be optimal, for example, to use the same distance metric for face recognition as for gender identification, even if in both tasks, distances are computed between the same fixed-size images. In fact, a number of researchers have demonstrated that kNN classification can be greatly improved by learning an appropriate distance metric from labeled examples [2, 6, 11, 12]. Even a simple linear transformation of input features has been shown to lead to significant improvements in kNN classification [6, 11]. Our work



Goldberger et al, NIPS 2005

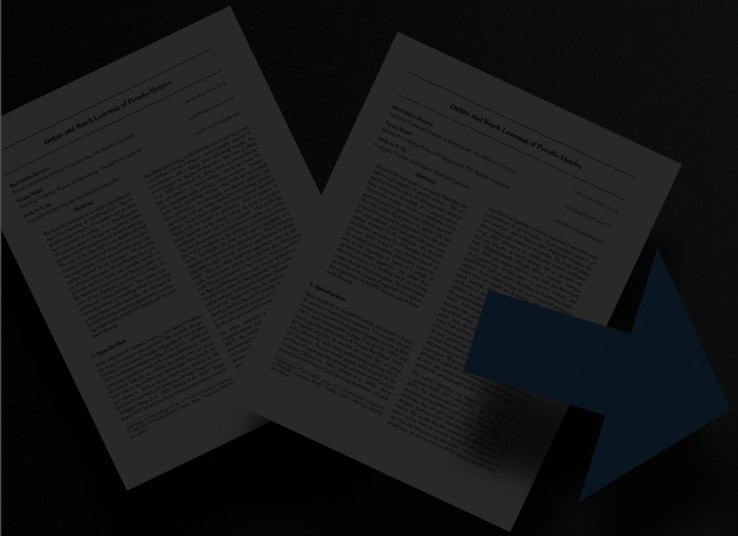


Chopra et al, CVPR 2005



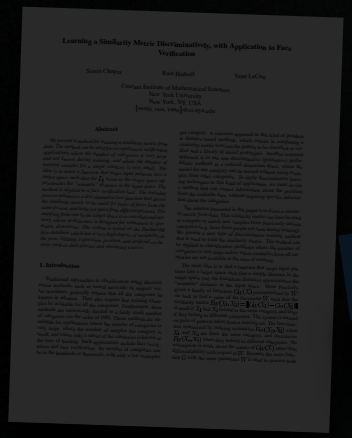
Vapnik 1998

Connection to previous work



Xing et al, NIPS 2003

Shalev-Shwartz et al, ICML 2004



Chopra et al, CVPR 2005

Distance Metric Learning for Large Margin Nearest Neighbor Classification

Kilian Q. Weinberger, John Blitzer and Lawrence K. Saul
Department of Computer and Information Science, University of Pennsylvania
Levine Hall, 3330 Walnut Street, Philadelphia, PA 19104
{kilianw, blitzer, lsaul}@cis.upenn.edu

Abstract

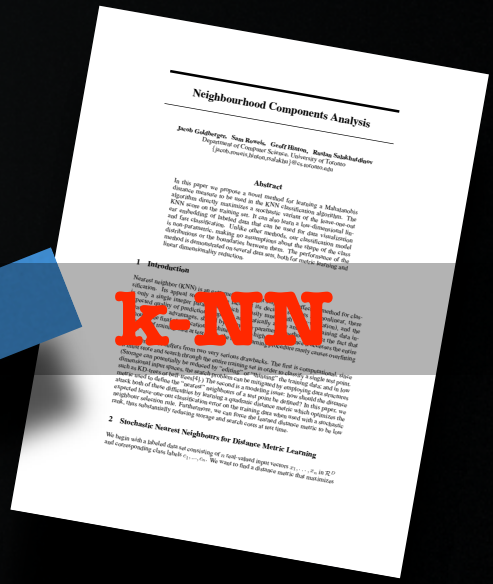
We show how to learn a Mahanalobis distance metric for k-nearest neighbor (kNN) classification by semidefinite programming. The metric is trained with the goal that k-nearest neighbors always belong to the same class while examples from different classes are separated by a large margin. On seven data sets of varying size and difficulty, we find that metrics trained in this way lead to significant improvements in kNN classification—for example, achieving a test error rate of 1.6% on the MNIST handwritten digits. Our approach has many parallels to support vector machines, including a convex objective function based on the hinge loss and the potential to work in nonlinear feature spaces by using the "kernel trick". On the other hand, our framework requires no modification for problems with large numbers of classes.

1 Introduction

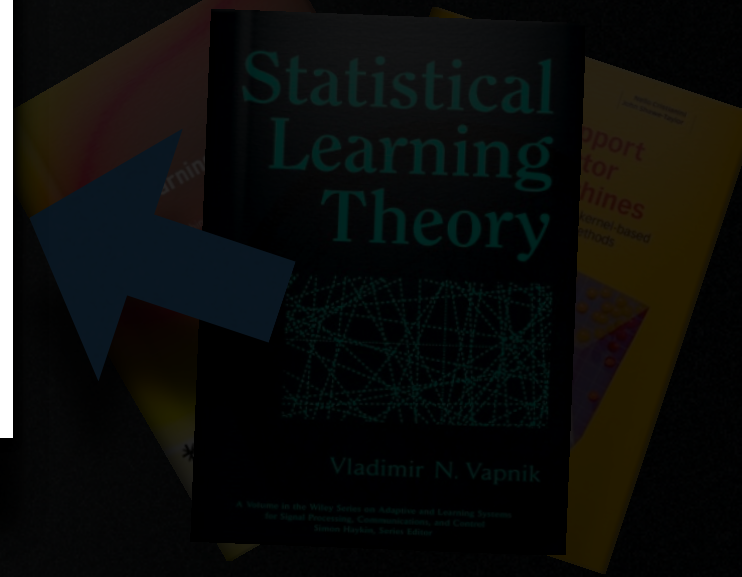
(DRAFT!!! Please do not distribute.) The k-nearest neighbors (kNN) rule [3] is one of the oldest and simplest methods for pattern classification. Nevertheless, it often yields competitive results, and in certain domains, when cleverly combined with prior knowledge, it has significantly advanced the state-of-the-art [1, 13]. The kNN rule classifies each unlabeled example by the majority label among its k-nearest neighbors in the training set. Its performance thus depends crucially on the *distance metric* used to identify nearest neighbors.

In the absence of prior knowledge, most kNN classifiers use simple Euclidean distances to measure the dissimilarities between examples represented as vector inputs. Euclidean distance metrics, however, do not capitalize on any statistical regularities in the data that might be estimated from a large training set of labeled examples.

Ideally, the distance metric for kNN classification should be adapted to the particular problem being solved. It can hardly be optimal, for example, to use the same distance metric for face recognition as for gender identification, even if in both tasks, distances are computed between the same fixed-size images. In fact, a number of researchers have demonstrated that kNN classification can be greatly improved by learning an appropriate distance metric from labeled examples [2, 6, 11, 12]. Even a simple linear transformation of input features has been shown to lead to significant improvements in kNN classification [6, 11]. Our work



Goldberger et al, NIPS 2005

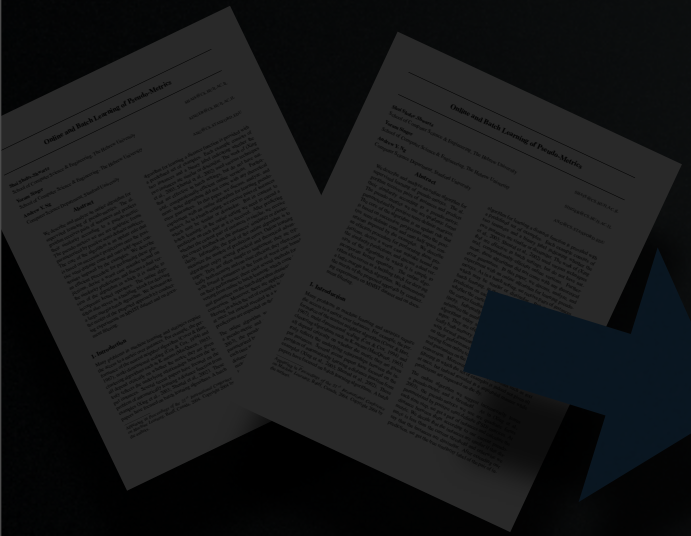


Vapnik 1998

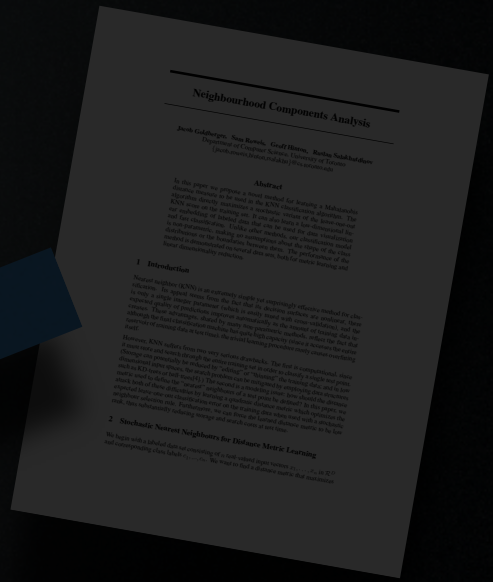
Connection to previous work

Xing et al, NIPS 2003

Shalev-Shwartz et al, ICML 2004



Goldberger et al, NIPS 2005



Distance Metric Learning for Large Margin Nearest Neighbor Classification

Kilian Q. Weinberger, John Blitzer and Lawrence K. Saul
Department of Computer and Information Science, University of Pennsylvania
Levine Hall, 3330 Walnut Street, Philadelphia, PA 19104
{kilianw, blitzer, lsaul}@cis.upenn.edu

Abstract

We show how to learn a Mahanalobis distance metric for k-nearest neighbor (kNN) classification by semidefinite programming. The metric is trained with the goal that k-nearest neighbors always belong to the same class while examples from different classes are separated by a large margin. On seven data sets of varying size and difficulty, we find that metrics trained in this way lead to significant improvements in kNN classification—for example, achieving a test error rate of 1.6% on the MNIST handwritten digits. Our approach has many parallels to support vector machines, including a convex objective function based on the hinge loss and the potential to work in nonlinear feature spaces by using the "kernel trick". On the other hand, our framework requires no modification for problems with large numbers of classes.

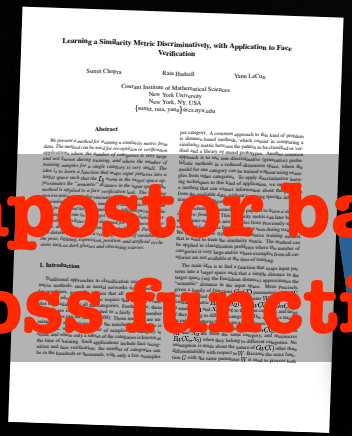
1 Introduction

(DRAFT!!! Please do not distribute.) The k-nearest neighbors (kNN) rule [3] is one of the oldest and simplest methods for pattern classification. Nevertheless, it often yields competitive results, and in certain domains, when cleverly combined with prior knowledge, it has significantly advanced the state-of-the-art [1, 13]. The kNN rule classifies each unlabeled example by the majority label among its k-nearest neighbors in the training set. Its performance thus depends crucially on the *distance metric* used to identify nearest neighbors.

In the absence of prior knowledge, most kNN classifiers use simple Euclidean distances to measure the dissimilarities between examples represented as vector inputs. Euclidean distance metrics, however, do not capitalize on any statistical regularities in the data that might be estimated from a large training set of labeled examples.

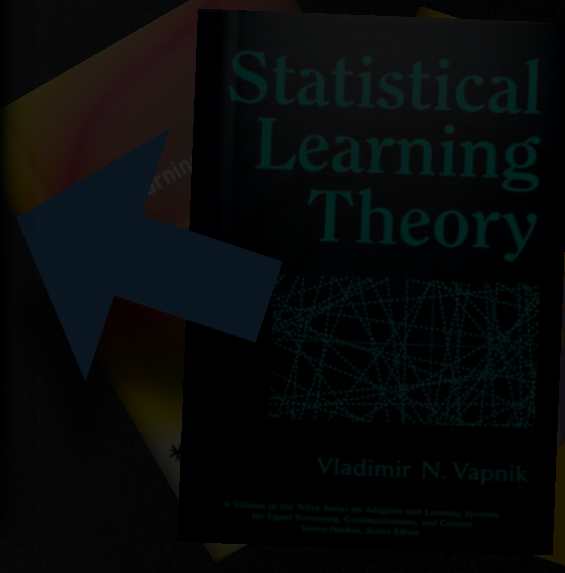
Ideally, the distance metric for kNN classification should be adapted to the particular problem being solved. It can hardly be optimal, for example, to use the same distance metric for face recognition as for gender identification, even if in both tasks, distances are computed between the same fixed-size images. In fact, a number of researchers have demonstrated that kNN classification can be greatly improved by learning an appropriate distance metric from labeled examples [2, 6, 11, 12]. Even a simple linear transformation of input features has been shown to lead to significant improvements in kNN classification [6, 11]. Our work

impostor based loss function

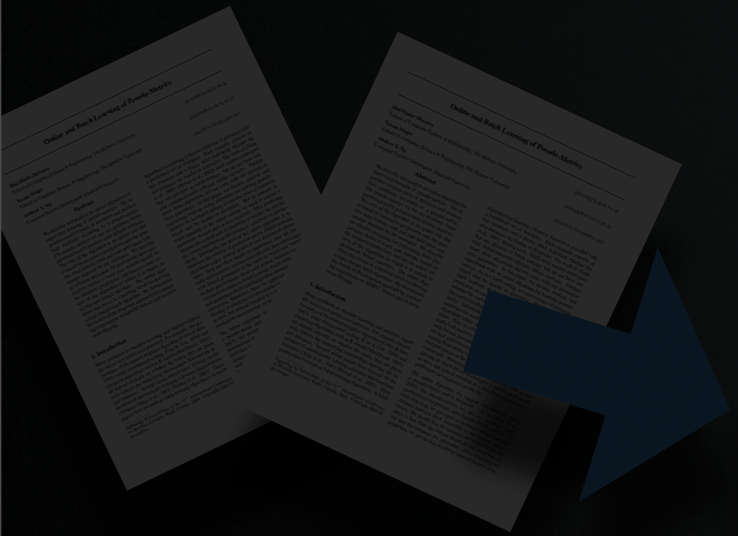


Chopra et al, CVPR 2005

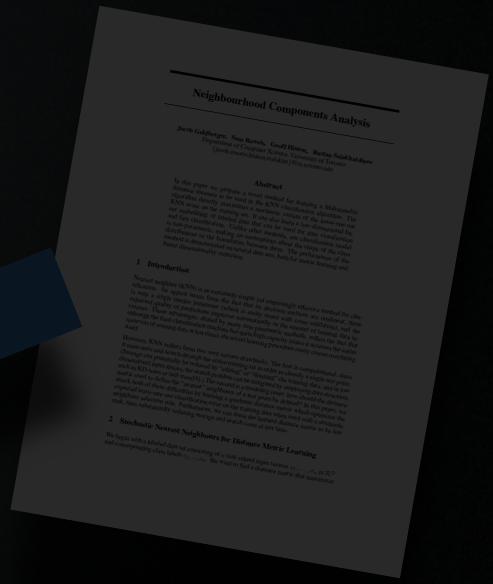
Vapnik 1998



Connection to previous work

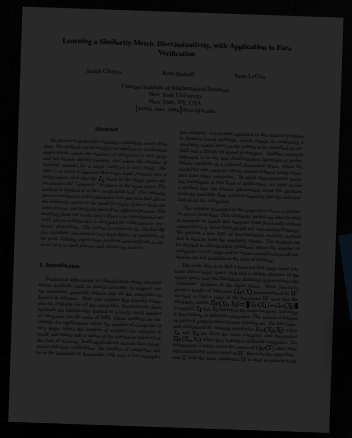


Xing et al, NIPS 2003

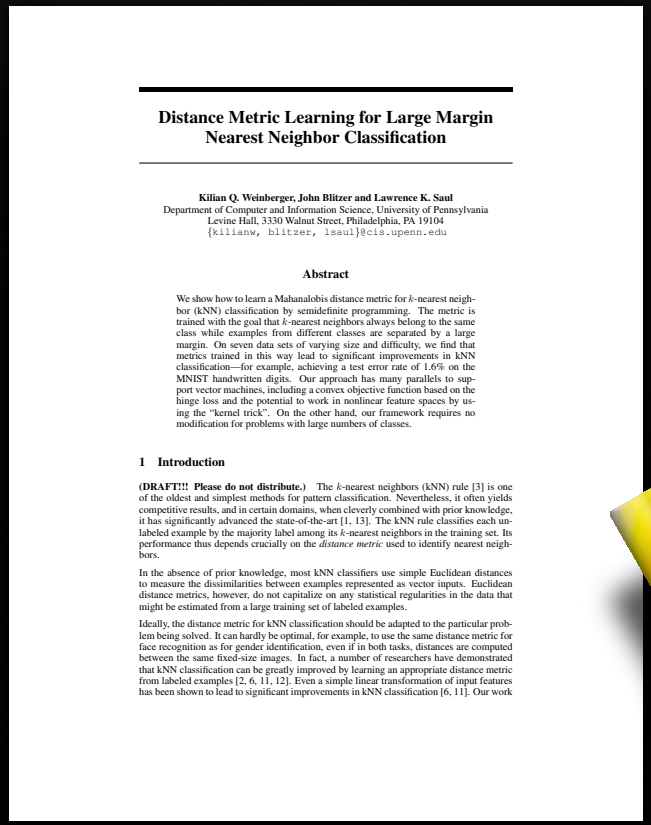


Goldberger et al, NIPS 2005

Shalev-Shwartz et al, ICML 2004



Chopra et al, CVPR 2005



Distance Metric Learning for Large Margin Nearest Neighbor Classification

Kilian Q. Weinberger, John Blitzer and Lawrence K. Saul
Department of Computer and Information Science, University of Pennsylvania
Levine Hall, 3330 Walnut Street, Philadelphia, PA 19104
{kilianw, blitzer, lsaul}@cis.upenn.edu

Abstract

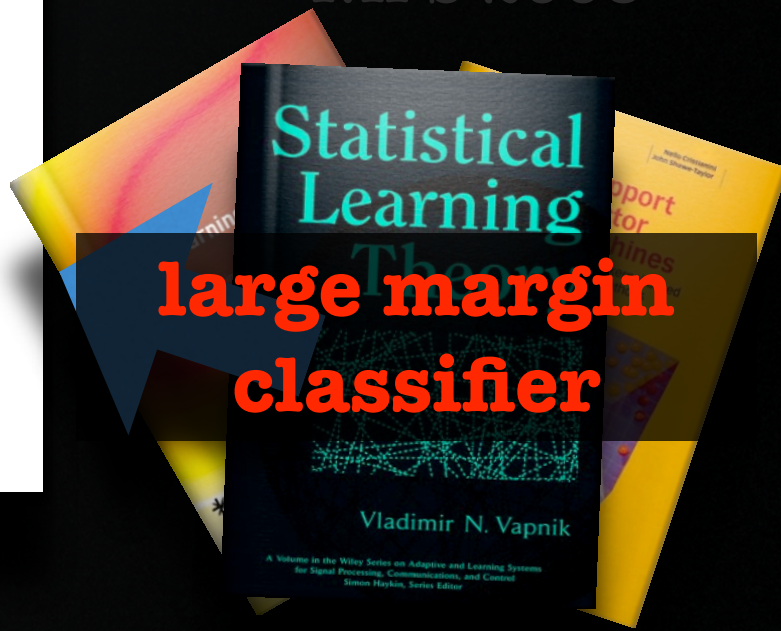
We show how to learn a Mahanalobis distance metric for k-nearest neighbor (kNN) classification by semidefinite programming. The metric is trained with the goal that k-nearest neighbors always belong to the same class while examples from different classes are separated by a large margin. On seven data sets of varying size and difficulty, we find that metrics trained in this way lead to significant improvements in kNN classification—for example, achieving a test error rate of 1.6% on the MNIST handwritten digits. Our approach has many parallels to support vector machines, including a convex objective function based on the hinge loss and the potential to work in nonlinear feature spaces by using the "kernel trick". On the other hand, our framework requires no modification for problems with large numbers of classes.

1 Introduction

(DRAFT!!! Please do not distribute.) The k-nearest neighbors (kNN) rule [3] is one of the oldest and simplest methods for pattern classification. Nevertheless, it often yields competitive results, and in certain domains, when cleverly combined with prior knowledge, it has significantly advanced the state-of-the-art [1, 13]. The kNN rule classifies each unlabeled example by the majority label among its k-nearest neighbors in the training set. Its performance thus depends crucially on the *distance metric* used to identify nearest neighbors.

In the absence of prior knowledge, most kNN classifiers use simple Euclidean distances to measure the dissimilarities between examples represented as vector inputs. Euclidean distance metrics, however, do not capitalize on any statistical regularities in the data that might be estimated from a large training set of labeled examples.

Ideally, the distance metric for kNN classification should be adapted to the particular problem being solved. It can hardly be optimal, for example, to use the same distance metric for face recognition as for gender identification, even if in both tasks, distances are computed between the same fixed-size images. In fact, a number of researchers have demonstrated that kNN classification can be greatly improved by learning an appropriate distance metric from labeled examples [2, 6, 11, 12]. Even a simple linear transformation of input features has been shown to lead to significant improvements in kNN classification [6, 11]. Our work



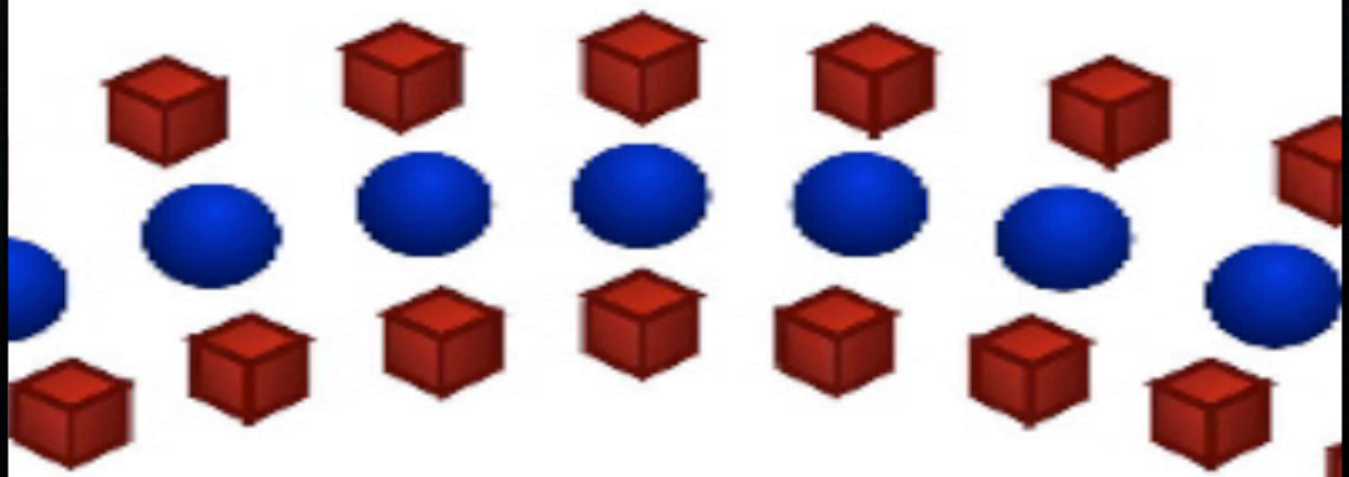
large margin classifier

Vapnik 1998

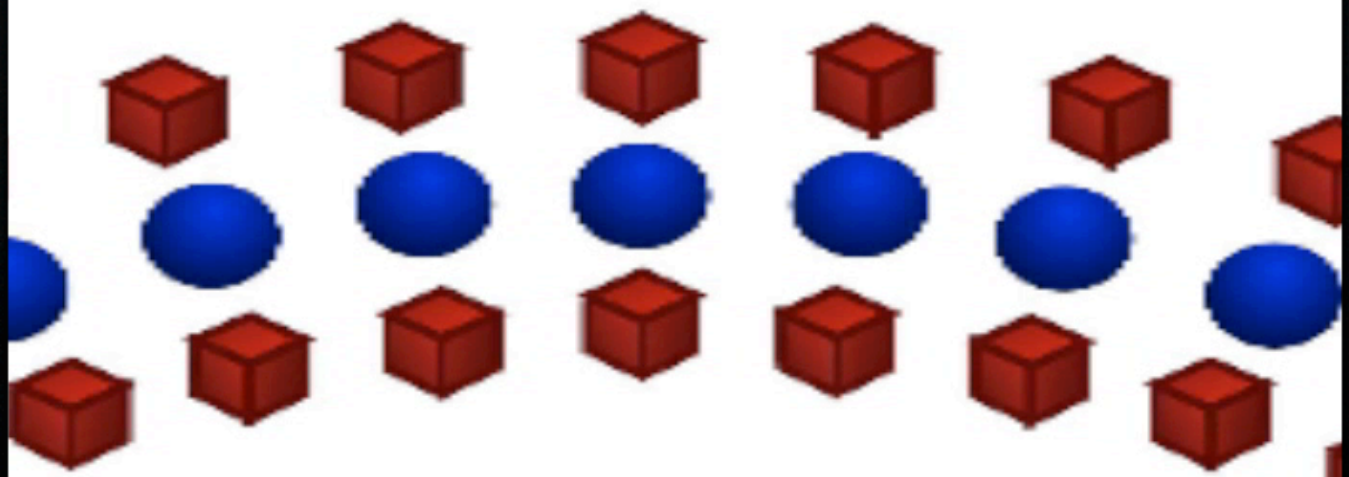
Conclusion

- Novel algorithm for metric learning
- Replaces linear decision boundary in SVM by kNN decision boundary
- Training complexity independent of number of classes (unlike multiclass SVM)
- Optimization is a semidefinite program
- (Can be kernelized)

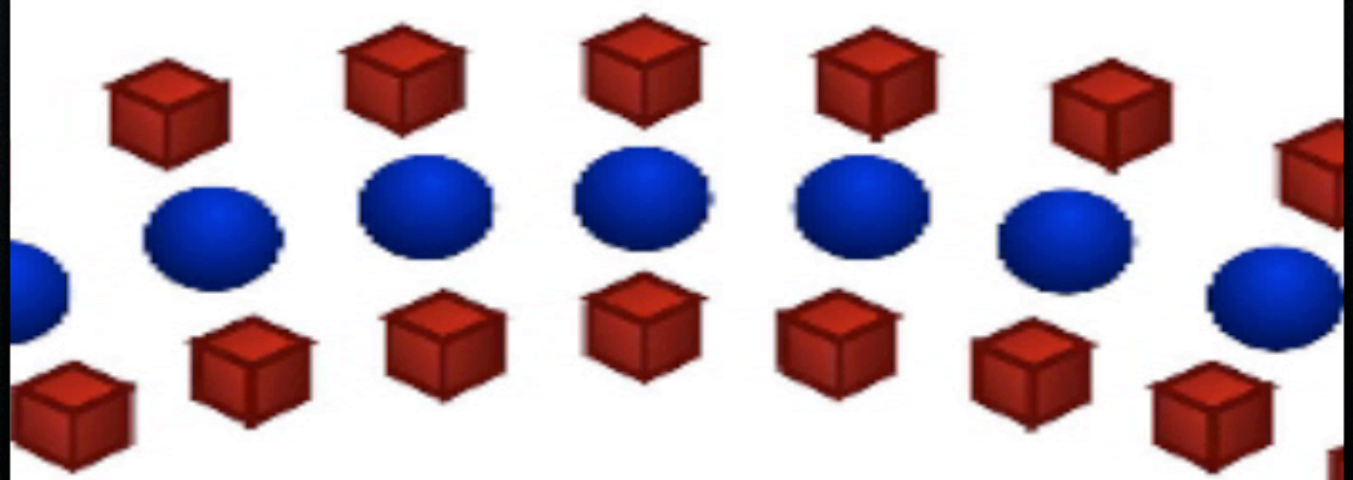
Future work



Future work



Future work



Locally adaptive k-NN that exploits manifold structure

Thank you!!!