Support Vector Metric Learning

Eddie Xu, **Kilian Q. Weinberger**, Olivier Chapelle, Fei Sha

Washington University in St. Louis

YAHOO! RESEARCH

Warning this talk may contain
small amounts of **neural networks.**

Support Vector Metric Learning

Eddie Xu, **Kilian Q. Weinberger**, Olivier Chapelle, Fei Sha

Washington University in St. Louis

YAHOO! RESEARCH

What is similarity?

Who is most similar?

Similar gender?

Similar age?

Similar hair cut?

Similarity depends on the context!

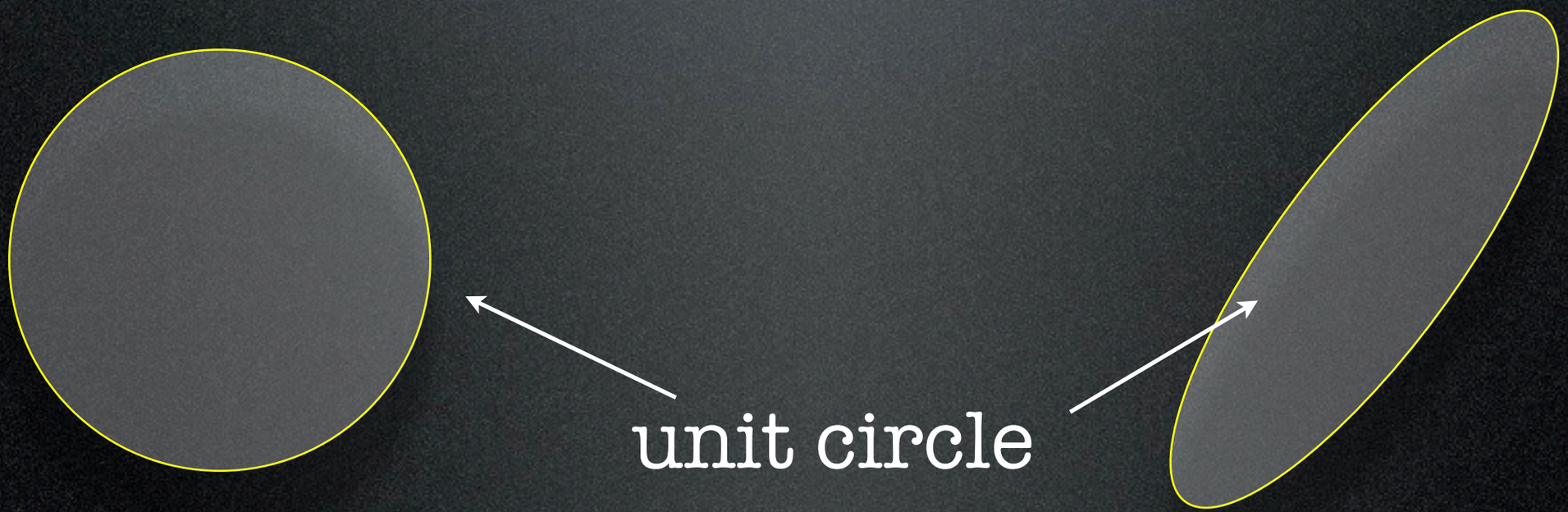
Mahalanobis distance

Euclidean

$$\|\vec{x}_i - \vec{x}_j\|_2 = \sqrt{(\vec{x}_i - \vec{x}_j)^\top (\vec{x}_i - \vec{x}_j)}$$

Mahalanobis

$$\|\vec{x}_i - \vec{x}_j\|_{\mathbf{M}} = \sqrt{(\vec{x}_i - \vec{x}_j)^\top \mathbf{M} (\vec{x}_i - \vec{x}_j)}$$



Mahalanobis distance

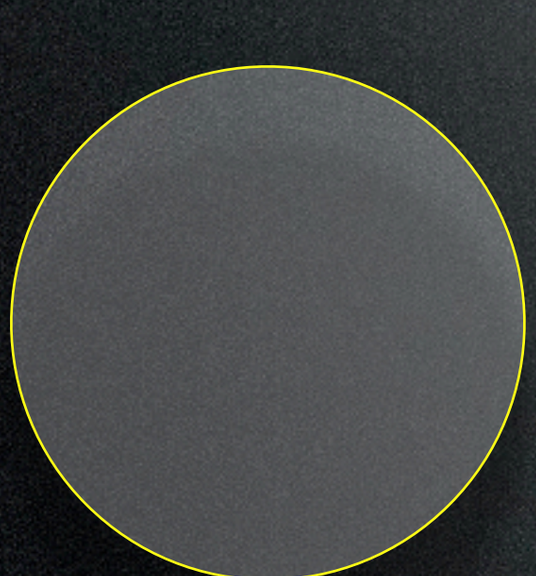
Euclidean

$$\|\vec{x}_i - \vec{x}_j\|_2 = \sqrt{(\vec{x}_i - \vec{x}_j)^\top (\vec{x}_i - \vec{x}_j)}$$

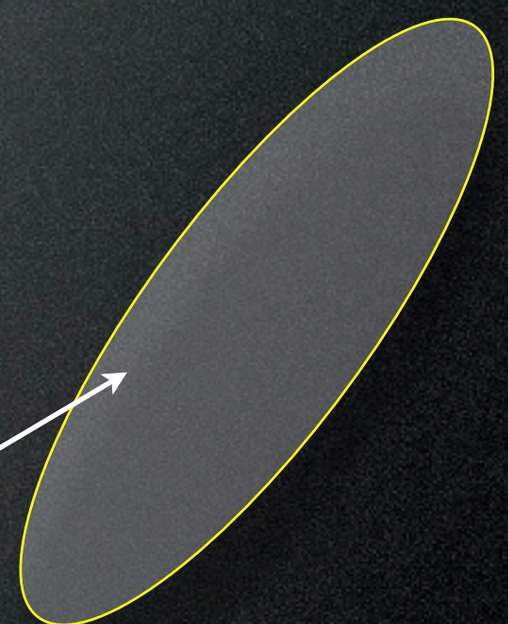
Mahalanobis

$$\|\vec{x}_i - \vec{x}_j\|_{\mathbf{M}} = \sqrt{(\vec{x}_i - \vec{x}_j)^\top \mathbf{M} (\vec{x}_i - \vec{x}_j)}$$

$\mathbf{M} \succeq 0$
positive
semi-definite



unit circle



Mahalanobis distance

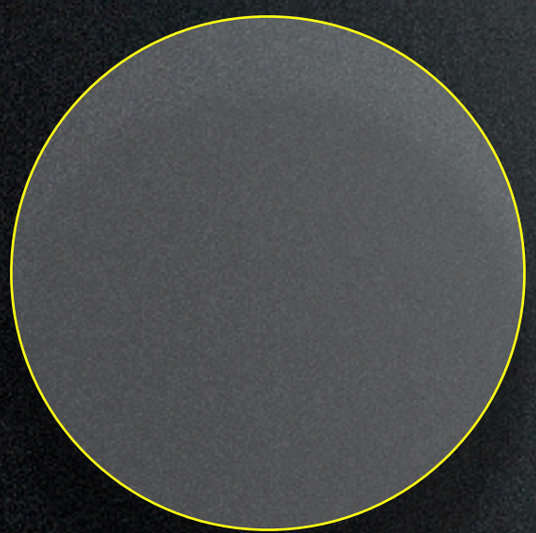
Euclidean

$$\|\vec{x}_i - \vec{x}_j\|_2 = \sqrt{(\vec{x}_i - \vec{x}_j)^\top (\vec{x}_i - \vec{x}_j)}$$

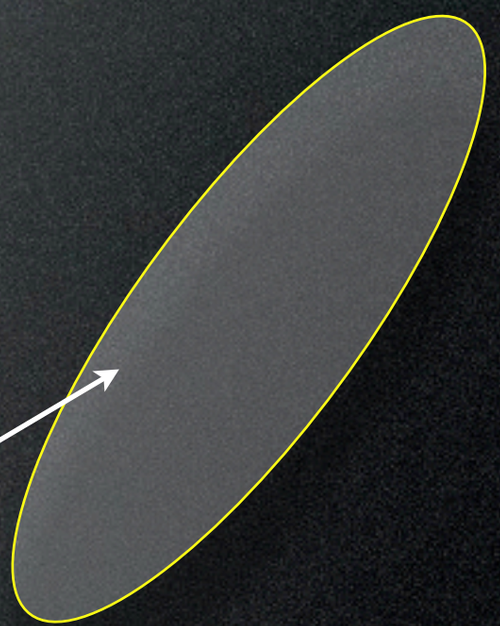
Mahalanobis

$$\|\vec{x}_i - \vec{x}_j\|_{\mathbf{M}} = \sqrt{(\vec{x}_i - \vec{x}_j)^\top \mathbf{M} (\vec{x}_i - \vec{x}_j)}$$

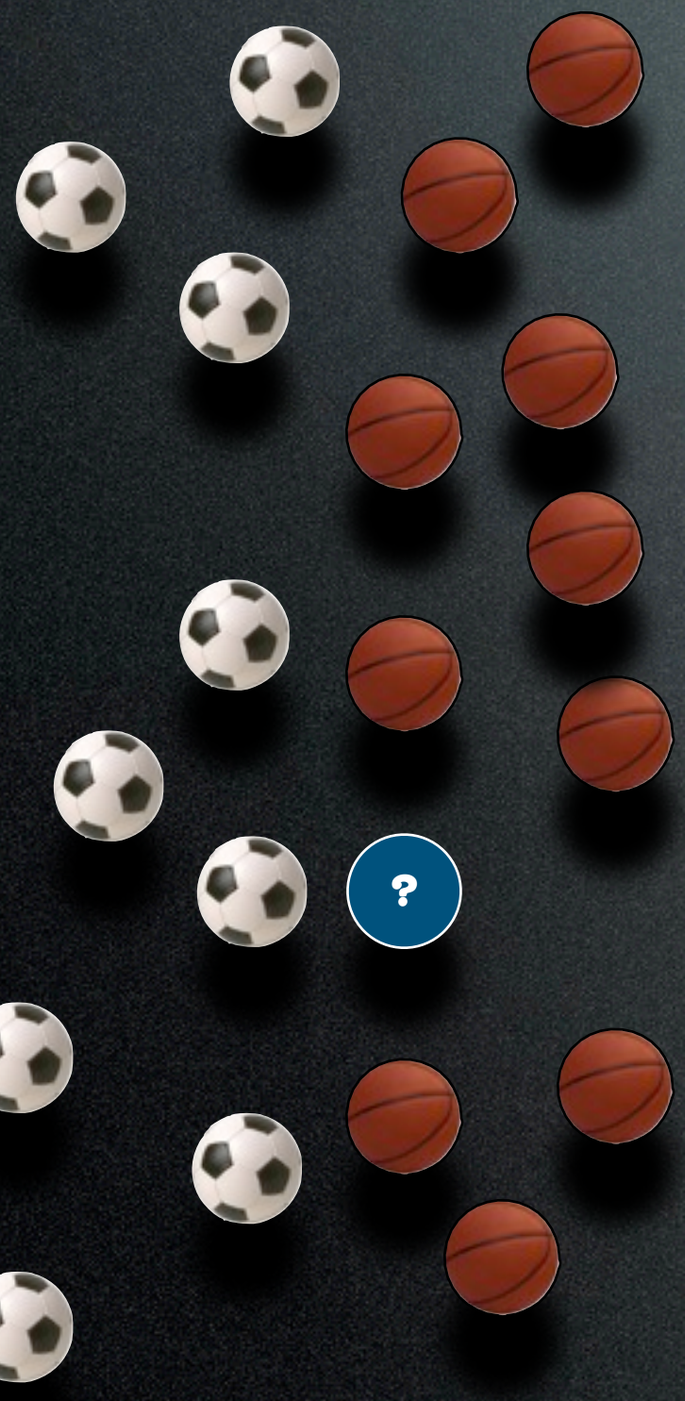
$\mathbf{M} \succeq 0$
positive
semi-definite



unit circle



k-NN classification



- Adapt metric to the data
  - Amplify informative directions
  - Squash non-informative directions

k-NN classification

- Adapt metric to the data
  - Amplify informative directions
  - Squash non-informative directions

k-NN classification

- Adapt metric to the data
  - Amplify informative directions
  - Squash non-informative directions

RCA	**Shental et al [eccv 2002]**
MPCK	**Bilenko et al [icml 2004]**
POLAR	**Shalev-Shwartz et al [icml 2004]**
NCA	**Goldberger et al [nips 2005]**
LMNN	**Weinberger et al [nips 2006]**
MLCC	**Globerson and Roweis [nips 2006]**
ITML	**Davis et al [icml 2007]**
mtLMNN	**Parameswaran and Weinberger [nips 2010]**

Support Vector Metric Learning

How about SVMs?

- SVM with RBF Kernel is very similar to nearest neighbor classifier

$$k(\vec{x}_i, \vec{x}) = e^{-\frac{1}{2\sigma^2}(\vec{x}_i - \vec{x})^\top (\vec{x}_i - \vec{x})}$$

How about SVMs?

- SVM with RBF Kernel is very similar to nearest neighbor classifier

$$k(\vec{x}_i, \vec{x}) = e^{-\frac{1}{2\sigma^2}(\vec{x}_i - \vec{x})^\top (\vec{x}_i - \vec{x})}$$

(squared) **Euclidean Distance**

How about SVMs?

- SVM with RBF Kernel is very similar to nearest neighbor classifier

$$k(\vec{x}_i, \vec{x}) = e^{-\frac{1}{2\sigma^2}(\vec{x}_i - \vec{x})^\top (\vec{x}_i - \vec{x})}$$

$$k(\vec{x}_i, \vec{x}) = e^{-\frac{1}{2\sigma^2}(\vec{x}_i - \vec{x})^\top \mathbf{M} (\vec{x}_i - \vec{x})}$$

(squared) **Mahalanobis Distance**

First Idea

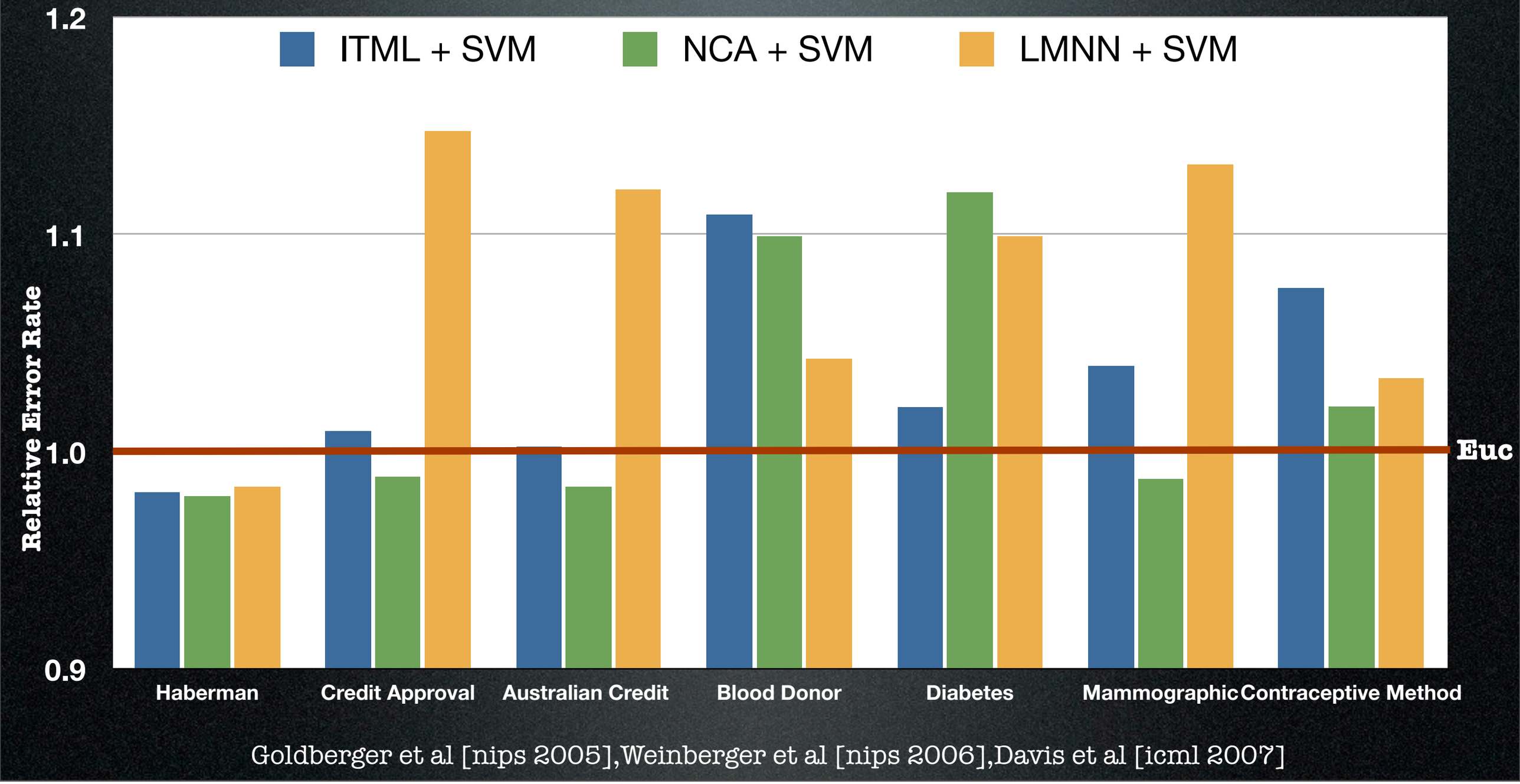
First Metric Learning, then SVM.

Baseline:
RBF Kernel with cross validation
(Features normalized)

Euc

Naive Experiments

First Metric Learning, then SVM.



Goldberger et al [nips 2005],Weinberger et al [nips 2006],Davis et al [icml 2007]

SVM

$$\min_{\alpha_1,\ldots,\alpha_n} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{K}_{ij}$$

subject to:

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0$$

SVML

$$\min_{\mathbf{M}} \text{Validation Error}$$

$$\min_{\alpha_1,\ldots,\alpha_n} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{K}_{ij}$$

subject to:

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0$$

Chapelle et al 2002, Kloft et al 2010

SVML

$$\min_{\mathbf{M}} \frac{1}{|V|} \sum_{(\mathbf{x},y)\in V} 0.5(\text{sign}(-y \sum_i \alpha_i y_i \mathbf{K}^{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}) + b) + 1)$$

$$\min_{\alpha_1,\ldots,\alpha_n} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{K}^{\mathbf{M}}_{ij}$$

subject to:

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0$$

Chapelle et al 2002, Kloft et al 2010

SVML

$$\min_{\mathbf{M}} \frac{1}{|V|} \sum_{(\mathbf{x},y)\in V} s_a(-y \sum_i \alpha_i y_i \mathbf{K}^{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}) + b)$$

$$\min_{\alpha_1,\ldots,\alpha_n} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{K}^{\mathbf{M}}_{ij}$$

subject to:

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0$$

Chapelle et al 2002, Kloft et al 2010

SVML

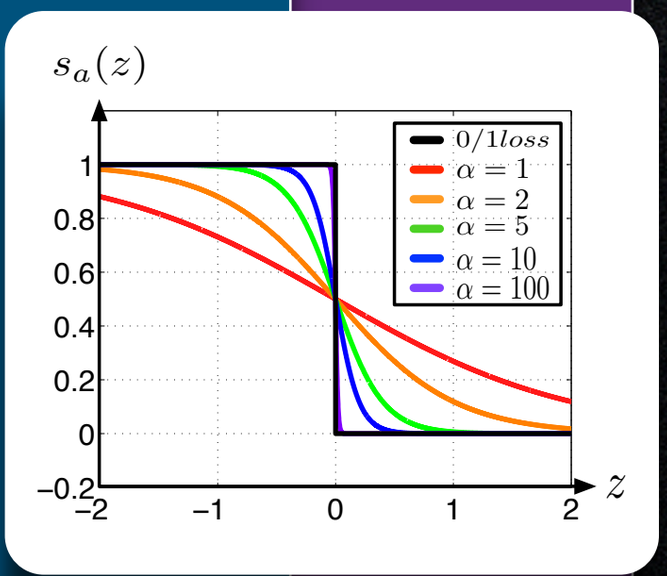
$$\min_{\mathbf{M}} \frac{1}{|V|} \sum_{(\mathbf{x},y)\in V} s_a(-y \sum_i \alpha_i y_i \mathbf{K}^{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}) + b)$$

$$\min_{\alpha_1,\ldots,\alpha_n} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{K}^{\mathbf{M}}_{ij}$$

subject to:

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0$$



Chapelle et al 2002, Kloft et al 2010

SVML

$$\min_{\mathbf{M}} \frac{1}{|V|} \sum_{(\mathbf{x},y)\in V} s_a(-y \sum_i \alpha_i y_i \mathbf{K}^{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}) + b)$$

$$\min_{\alpha_1,\ldots,\alpha_n} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{K}^{\mathbf{M}}_{ij}$$

subject to:

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0$$

Chapelle et al 2002, Kloft et al 2010

Two side notes:

1. We use the squared hinge-loss
2. We learn C together with $\mathbf{M}$

Two side notes:

$$\mathbf{K}^{\mathrm{M}} \leftarrow \mathbf{K}^{\mathrm{M}} + \frac{1}{C}\mathbf{I}$$

Removes slack variables.

1. We use the squared hinge-loss
2. We learn C together with $\mathbf{M}$

Two side notes:

$$\mathbf{K}^{\mathrm{M}} \leftarrow \mathbf{K}^{\mathrm{M}} + \frac{1}{C}\mathbf{I}$$

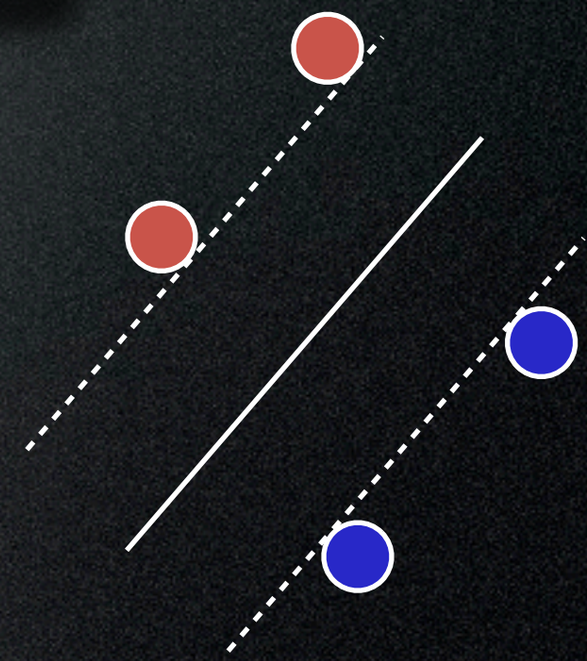
Removes slack variables.

1. We use the squared hinge-loss
2. We learn C together with M

Why not?

Gradient Descent

$$\frac{\partial h}{\partial \mathbf{M}} = \frac{\partial h}{\partial \alpha}\frac{\partial \alpha}{\partial \mathbf{M}} + \frac{\partial h}{\partial \mathbf{K}}\frac{\partial \mathbf{K}}{\partial \mathbf{M}} + \frac{\partial h}{\partial b}\frac{\partial b}{\partial \mathbf{M}}$$

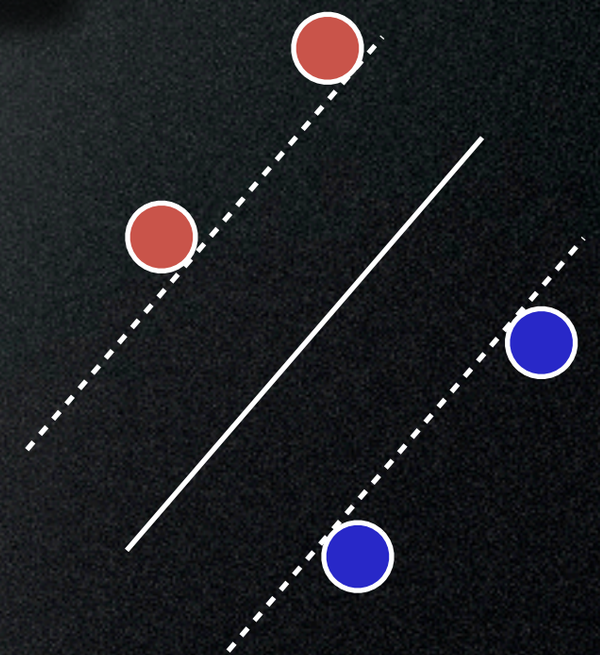


Chapelle et al [Machine Learning, 2002]

Gradient Descent

$$\frac{\partial h}{\partial \mathbf{M}} = \frac{\partial h}{\partial \alpha}\frac{\partial \alpha}{\partial \mathbf{M}} + \frac{\partial h}{\partial \mathbf{K}}\frac{\partial \mathbf{K}}{\partial \mathbf{M}} + \frac{\partial h}{\partial b}\frac{\partial b}{\partial \mathbf{M}}$$

$$\underbrace{\begin{pmatrix} \bar{\mathbf{K}} & \mathbf{y} \\ \mathbf{y}^\top & 0 \end{pmatrix}}_{\mathbf{H}} \begin{pmatrix} \alpha \\ b \end{pmatrix} = \begin{pmatrix} \mathbf{1} \\ 0 \end{pmatrix}$$



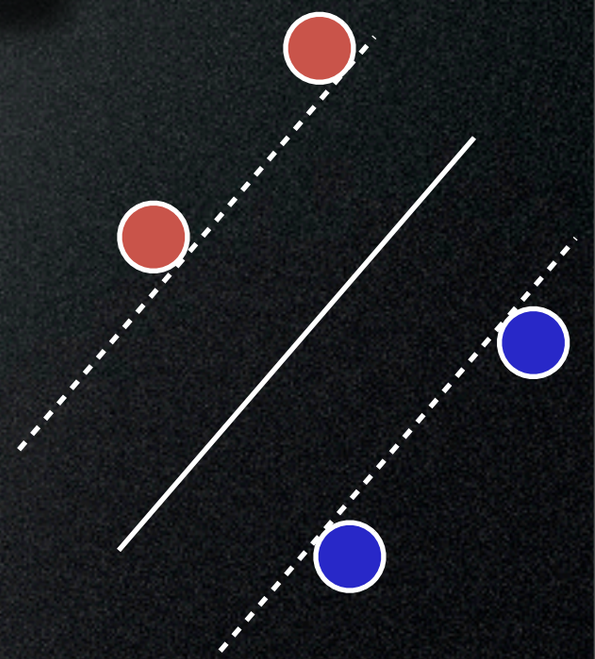
Chapelle et al [Machine Learning, 2002]

Gradient Descent

$$\frac{\partial h}{\partial \mathbf{M}} = \frac{\partial h}{\partial \alpha}\frac{\partial \alpha}{\partial \mathbf{M}} + \frac{\partial h}{\partial \mathbf{K}}\frac{\partial \mathbf{K}}{\partial \mathbf{M}} + \frac{\partial h}{\partial b}\frac{\partial b}{\partial \mathbf{M}}$$

$$\underbrace{\begin{pmatrix} \bar{\mathbf{K}} & \mathbf{y} \\ \mathbf{y}^\top & 0 \end{pmatrix}}_{\mathbf{H}} \begin{pmatrix} \alpha \\ b \end{pmatrix} = \begin{pmatrix} \mathbf{1} \\ 0 \end{pmatrix}$$

$$(\alpha, b)^\top = \mathbf{H}^{-1}(1 \cdots 1, 0)^\top$$



Chapelle et al [Machine Learning, 2002]

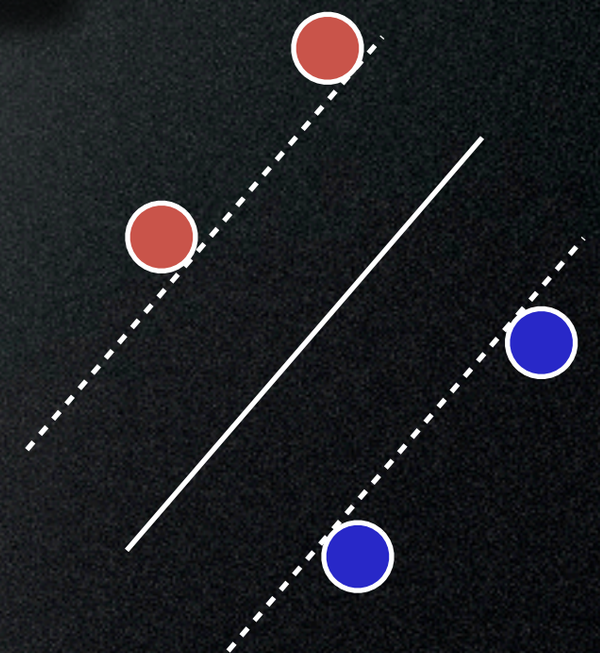
Gradient Descent

$$\frac{\partial h}{\partial \mathbf{M}} = \frac{\partial h}{\partial \alpha}\frac{\partial \alpha}{\partial \mathbf{M}} + \frac{\partial h}{\partial \mathbf{K}}\frac{\partial \mathbf{K}}{\partial \mathbf{M}} + \frac{\partial h}{\partial b}\frac{\partial b}{\partial \mathbf{M}}$$

$$\underbrace{\begin{pmatrix} \bar{\mathbf{K}} & \mathbf{y} \\ \mathbf{y}^\top & 0 \end{pmatrix}}_{\mathbf{H}} \begin{pmatrix} \alpha \\ b \end{pmatrix} = \begin{pmatrix} \mathbf{1} \\ 0 \end{pmatrix}$$

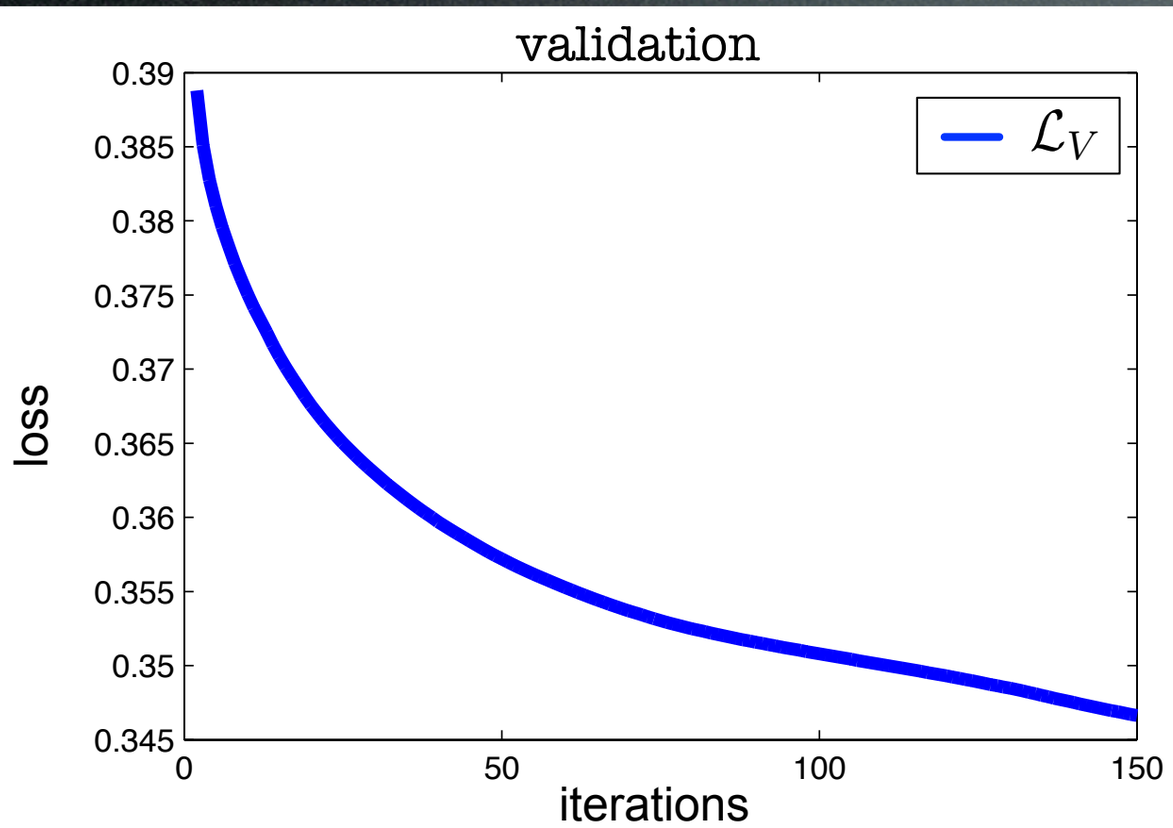
$$(\alpha, b)^\top = \mathbf{H}^{-1}(1 \cdots 1, 0)^\top$$

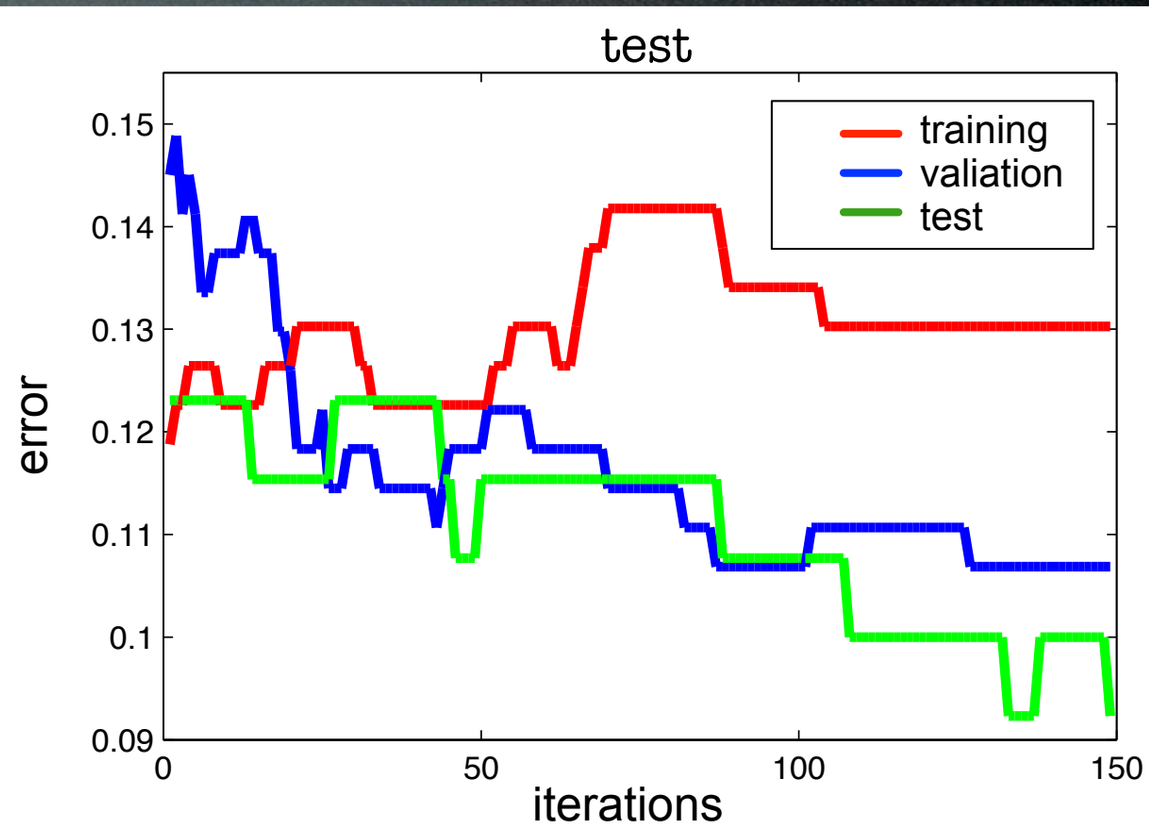
$$\frac{\partial(\alpha, b)}{\partial \mathbf{M}_{ij}} = -\mathbf{H}^{-1}\frac{\partial \mathbf{H}}{\partial \mathbf{M}_{ij}}(\alpha, b)^\top$$



Chapelle et al [Machine Learning, 2002]

Optimization





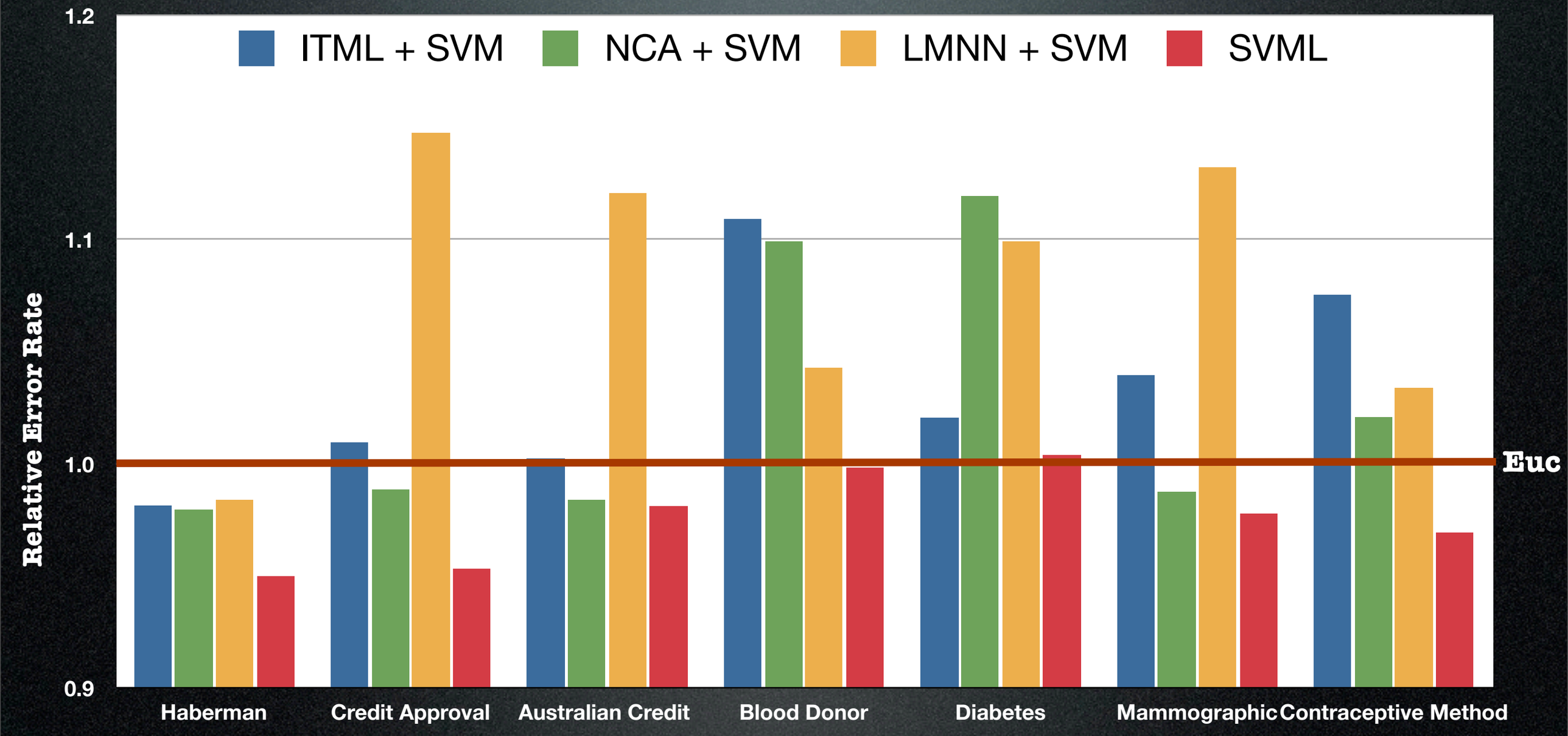
UCI credit card data set

Speed?

- Each gradient step involves solving the SVM
- But with conjugate gradient, a few steps are enough
- On average 4x faster than CV.

Results

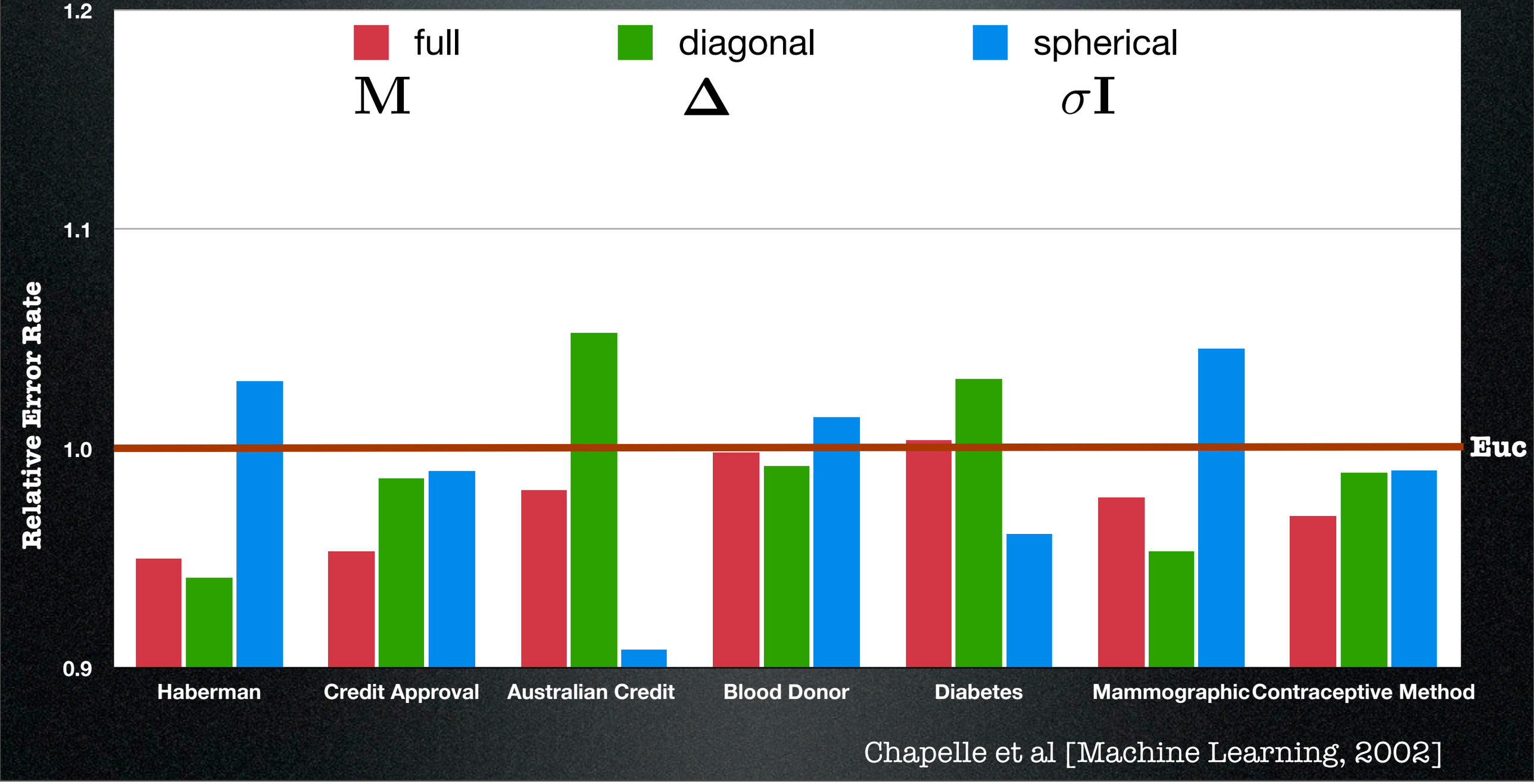
Joint Metric and SVM Learning.



Goldberger et al [nips 2005],Weinberger et al [nips 2006],Davis et al [icml 2007]

Results

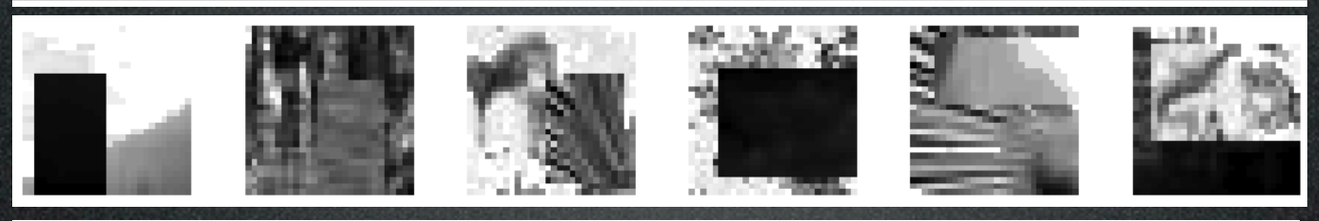
Diagonal and isotropic matrices.



Chapelle et al [Machine Learning, 2002]

Harder data sets

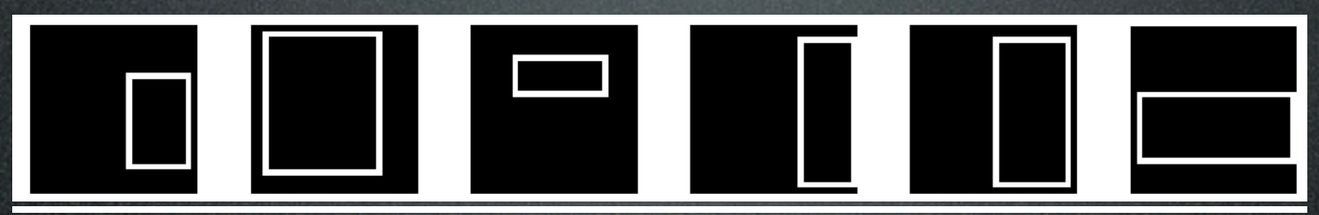
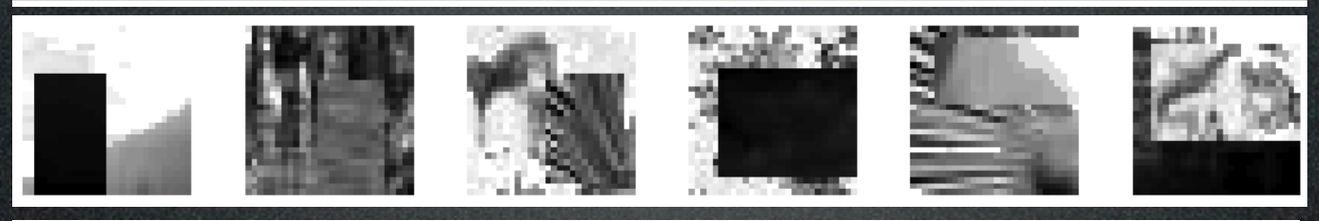
Deep Learning benchmarks!
(possibly designed to break SVMs)

		Train/Test
rectangles		1200/50000
rect-images		12000/50000
convex		8000/50000

[Vincent et al, 2008]

Harder data sets

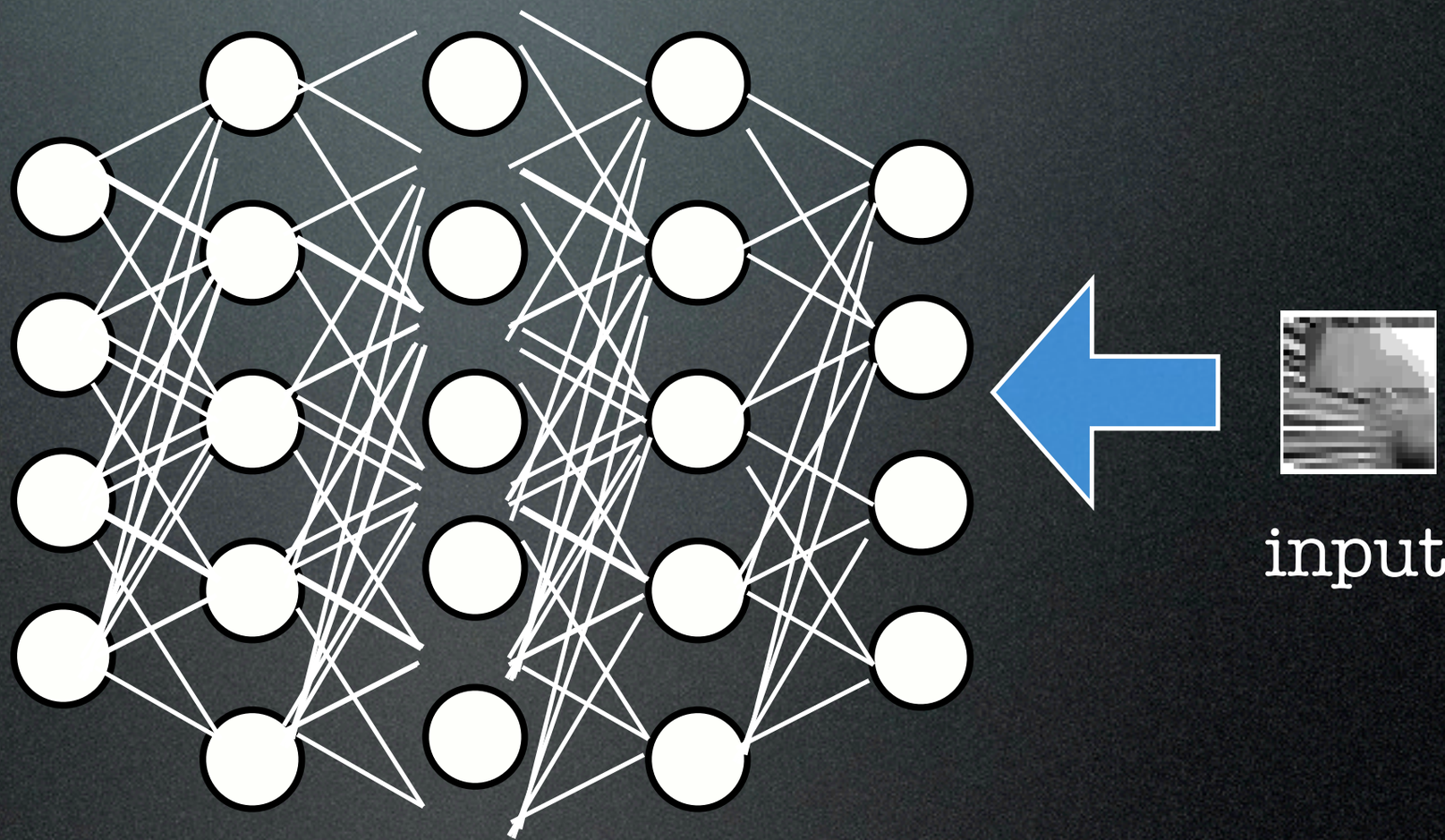
Deep Learning benchmarks!
(possibly designed to break SVMs)

		Train/Test
rectangles		1200/50000
rect-images		12000/50000
convex		8000/50000

But we need good features!!

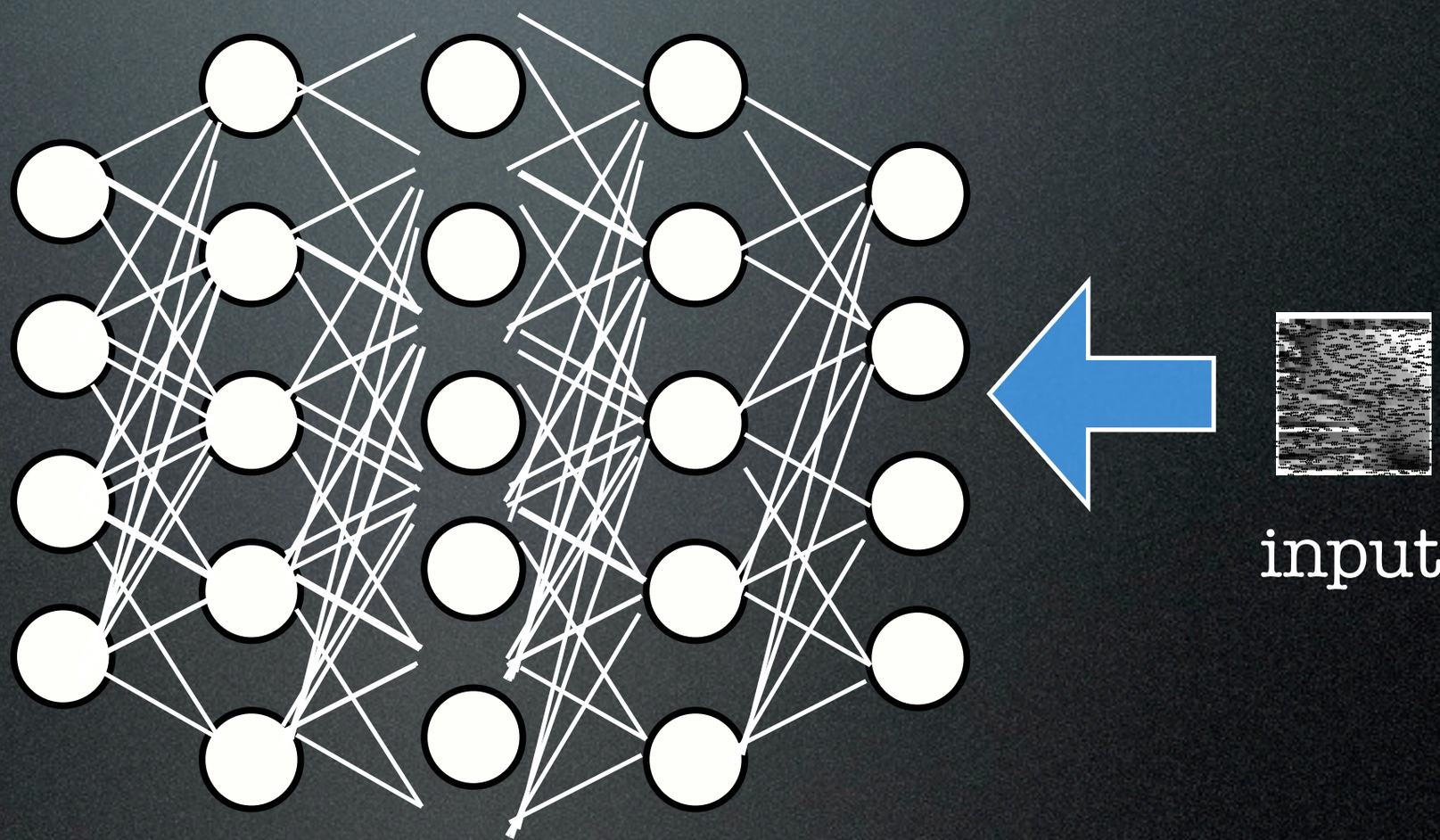
[Vincent et al, 2008]

Feature Generator: Denoising Autoencoder



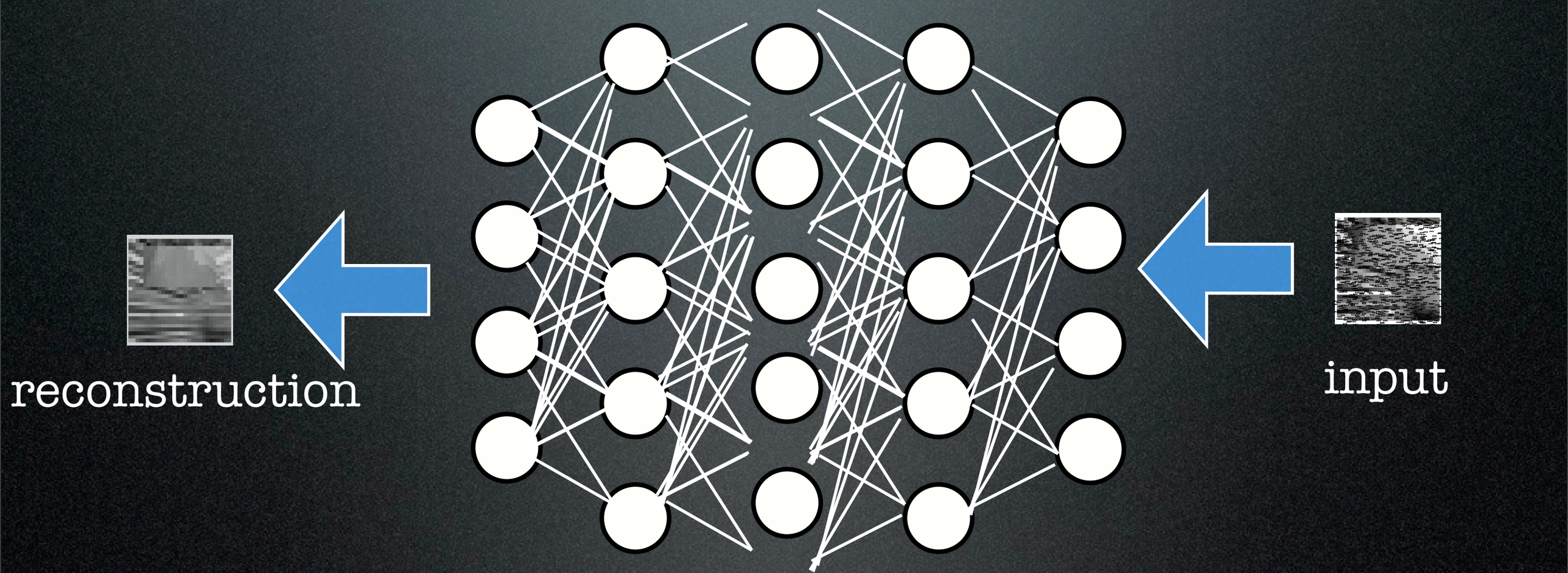
[Vincent et al, 2008]

Feature Generator: Denoising Autoencoder



[Vincent et al, 2008]

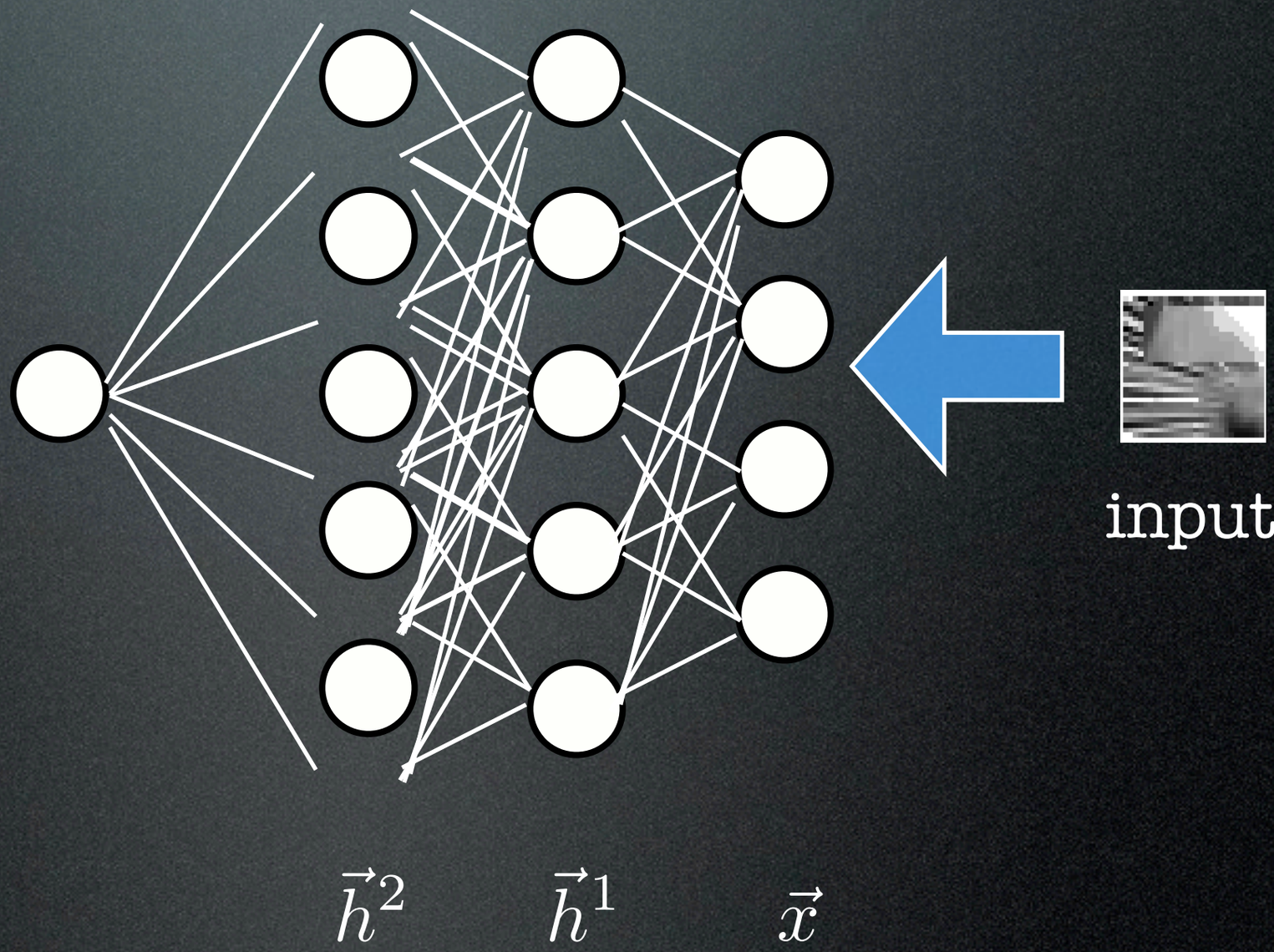
Feature Generator: Denoising Autoencoder



[Vincent et al, 2008]

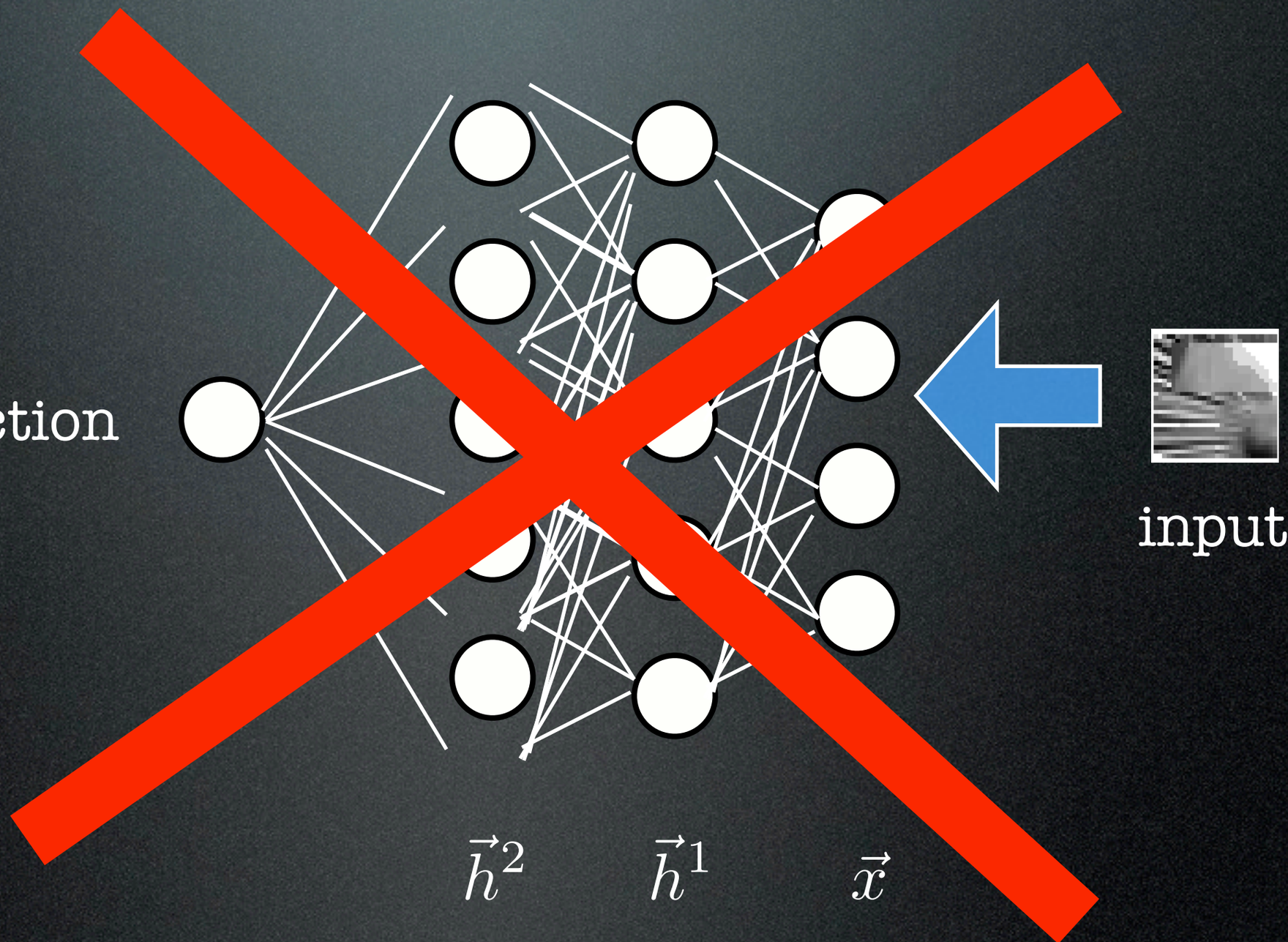
Fine-Tuning

(with backprop)

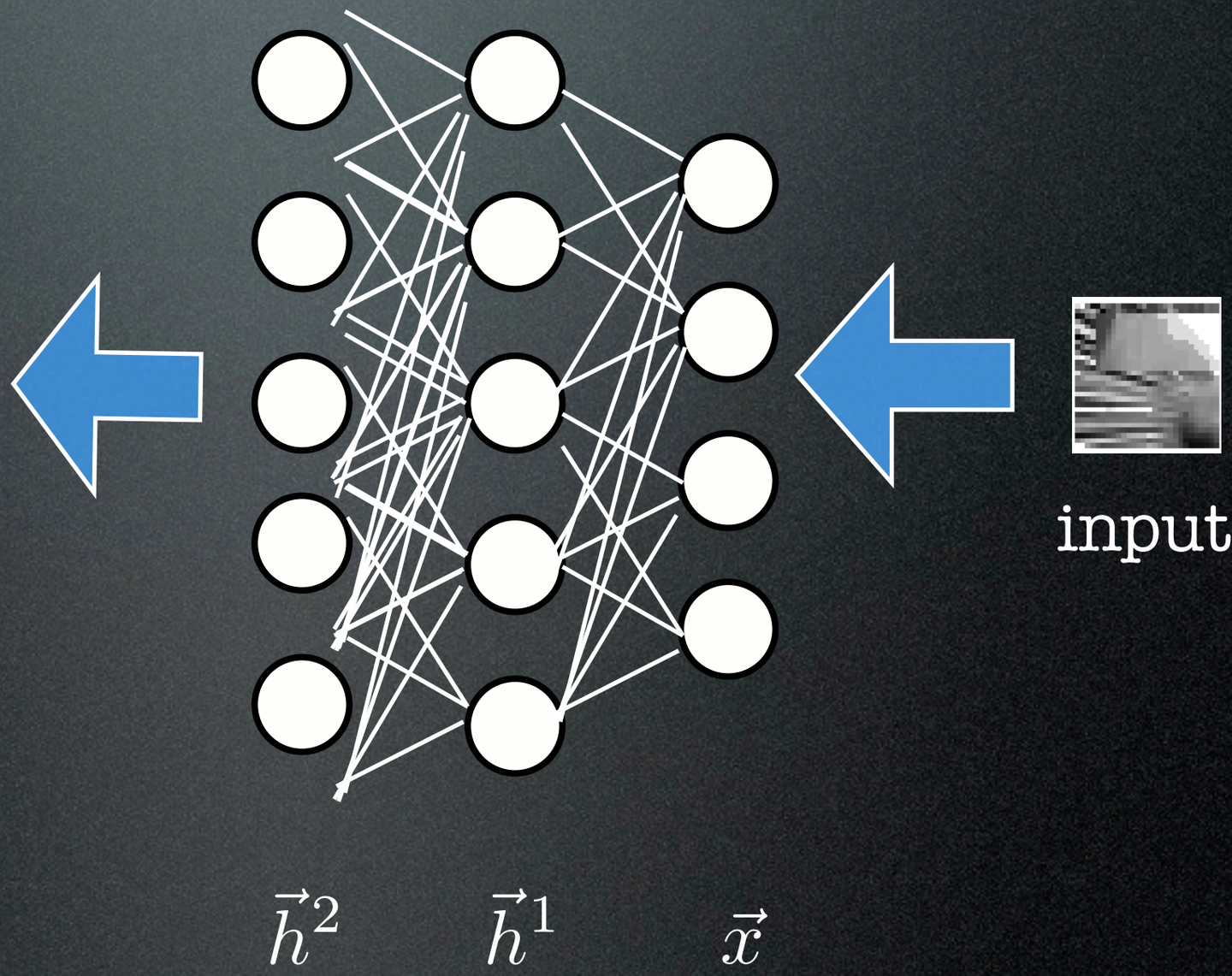


Fine-Tuning

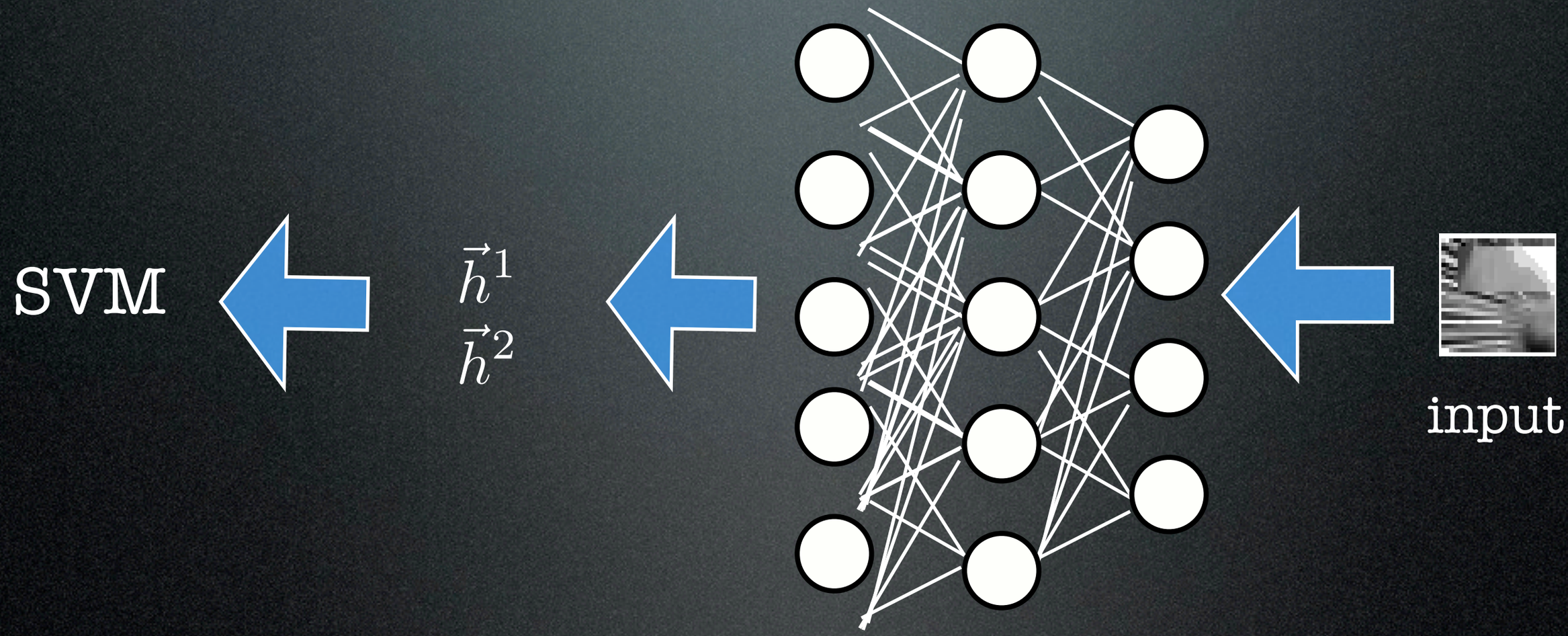
(with backprop)



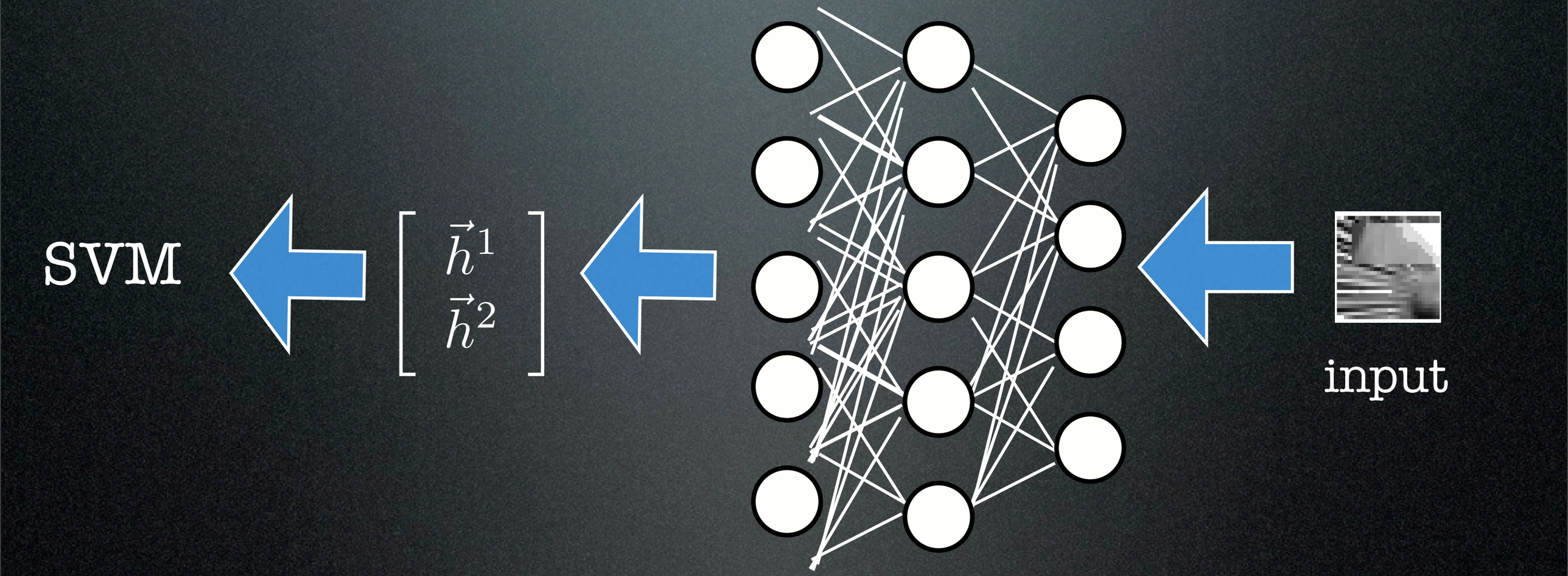
Features for SVMs



Features for SVMs



Features for SVMs



Deep Learning 101

$$\begin{bmatrix} \vec{h}^1 \\ \vec{h}^2 \end{bmatrix} \Rightarrow \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} \begin{bmatrix} \vec{h}^1 \\ \vec{h}^2 \end{bmatrix}$$

- Hidden layers are incompatible.
- Need to scale them appropriately.
- We use a diagonal matrix.

Deep Learning 101

$$\begin{bmatrix} \vec{h}^1 \\ \vec{h}^2 \end{bmatrix} \Rightarrow \underbrace{\begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}}_{\mathbf{M}} \begin{bmatrix} \vec{h}^1 \\ \vec{h}^2 \end{bmatrix}$$

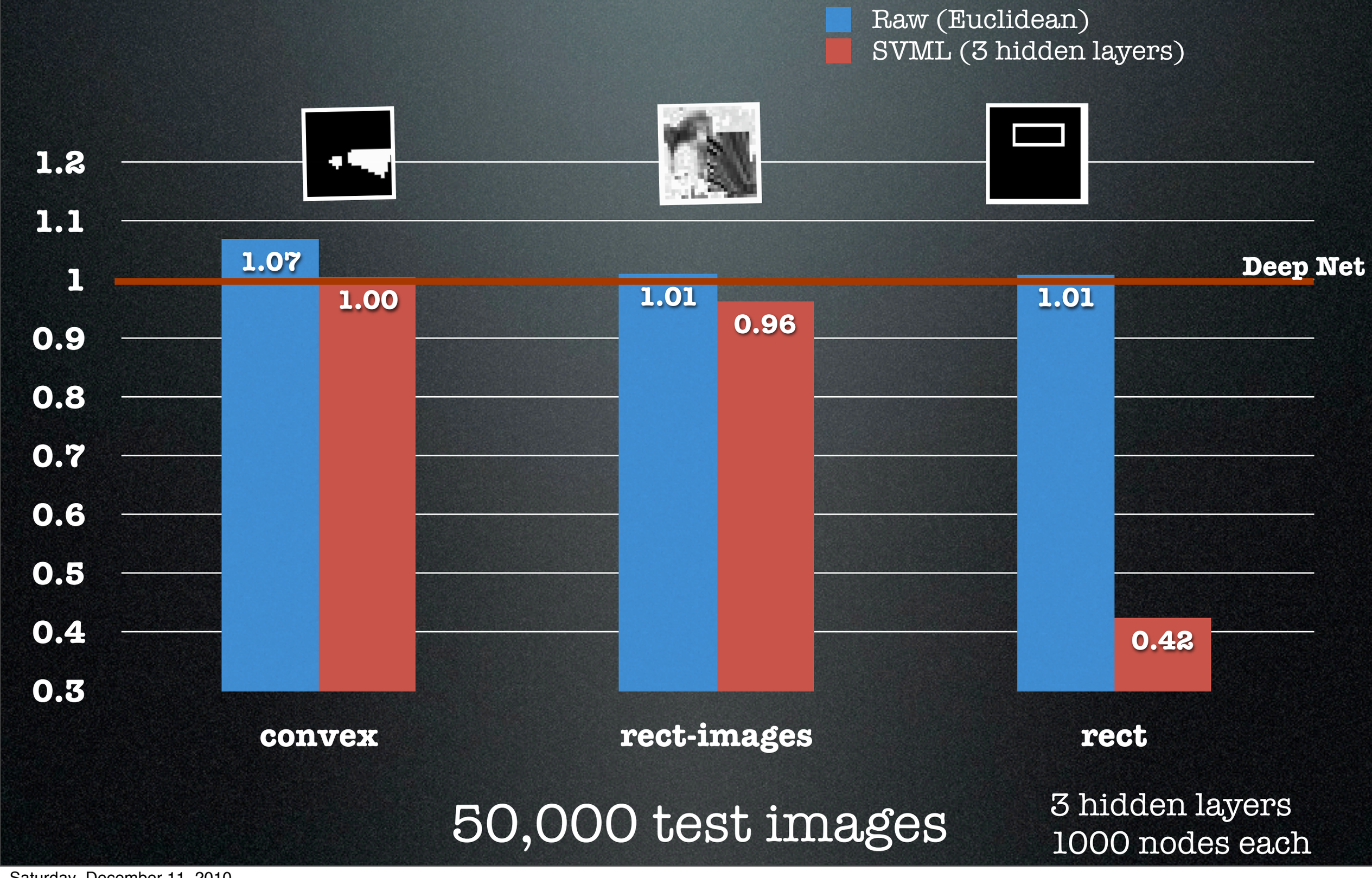
- Hidden layers are incompatible.
- Need to scale them appropriately.
- We use a diagonal matrix.

Deep Learning 101

$$k(\vec{x}_i, \vec{x}_j) = \exp\left(-\sum_{t=0}^{l} \frac{\|\vec{h}_i^t - \vec{h}_j^t\|_2^2}{\sigma_t^2}\right)$$

- Hidden layers are incompatible.
- Need to scale them appropriately.
- We use a diagonal matrix.

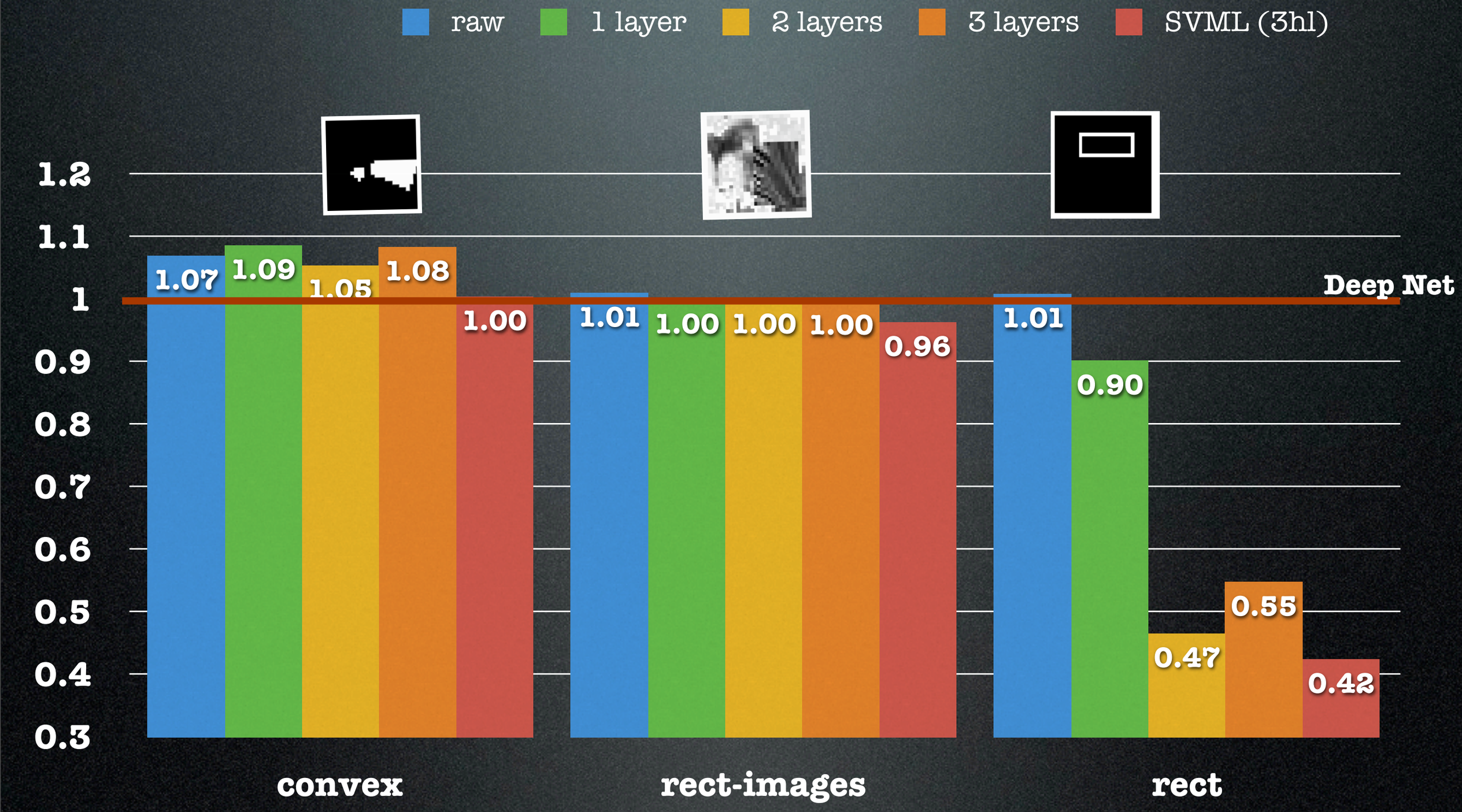
Results



50,000 test images

3 hidden layers
1000 nodes each

Results

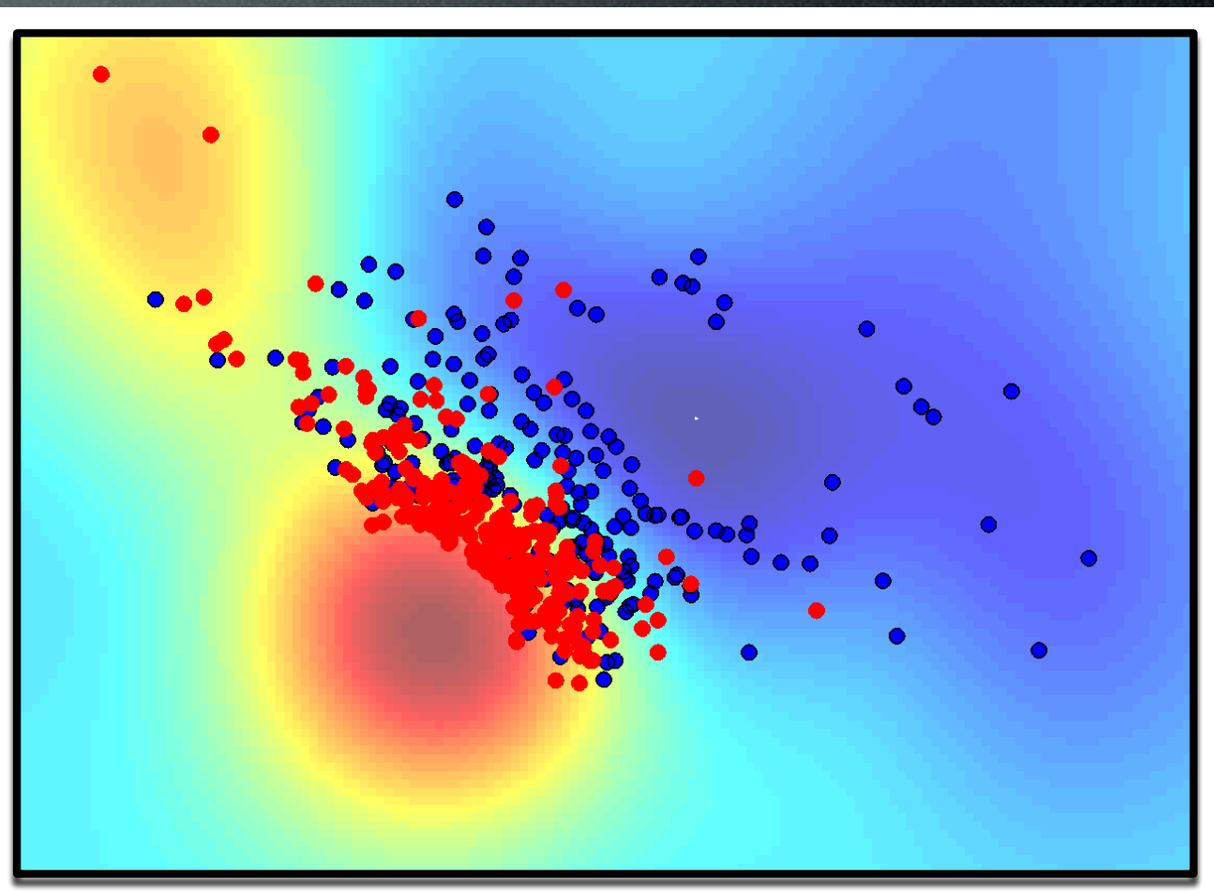


50,000 test images

3 hidden layers
1000 nodes each

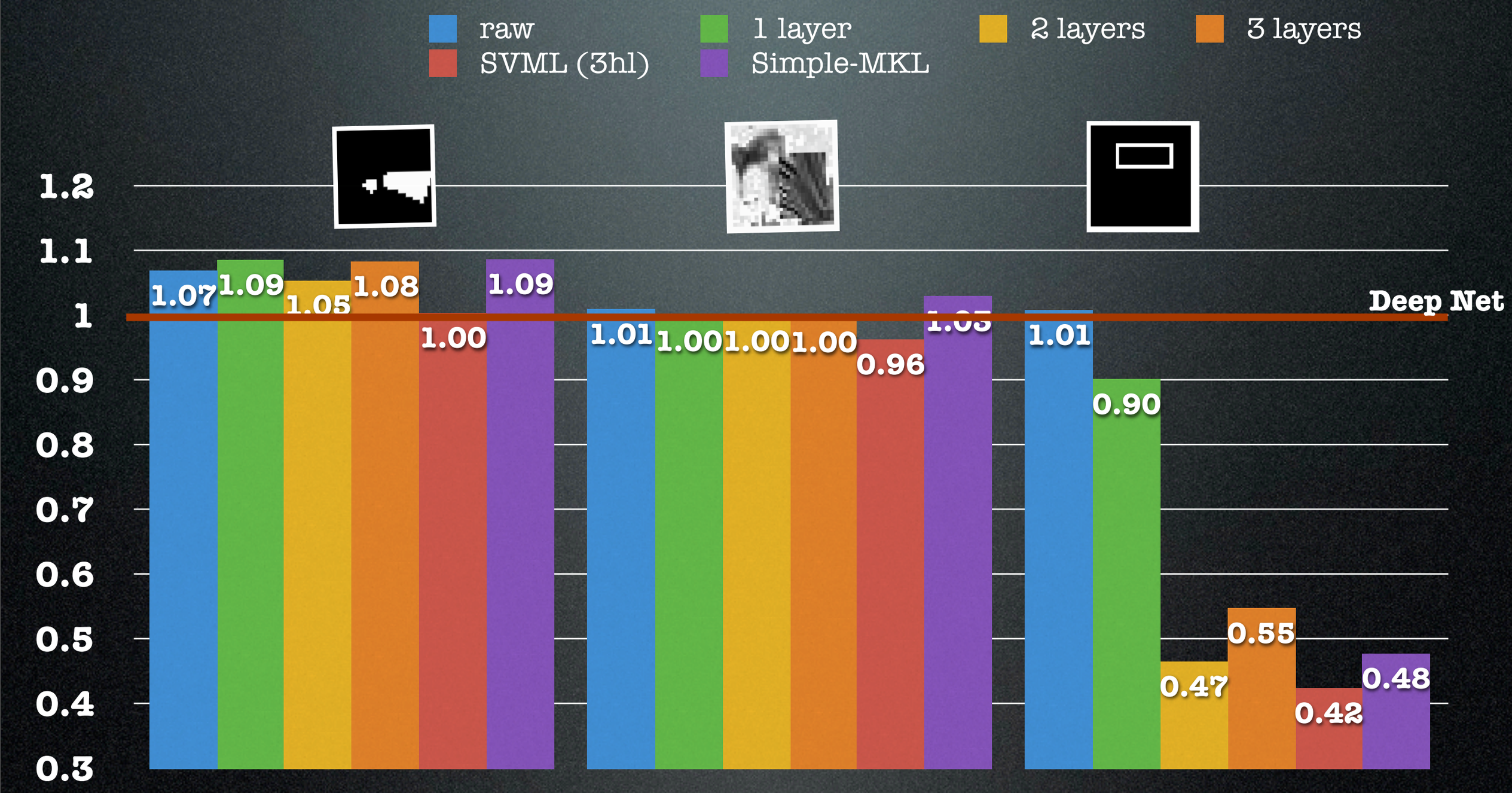
Conclusion

- For Metric Learning to work, it must be integrated into the SVM optimization
- SVML maybe a better alternative to cross validation
- Helps for challenging problems, if features come from incompatible sources
- Denoising auto-encoders are great feature generators for SVMs



SVML-2d

Results



50,000 test images