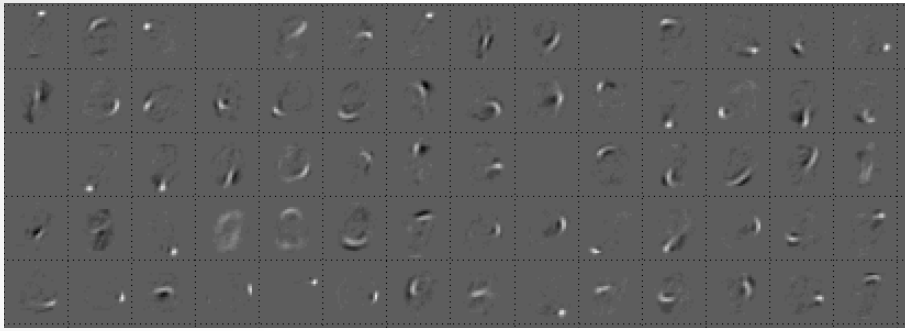
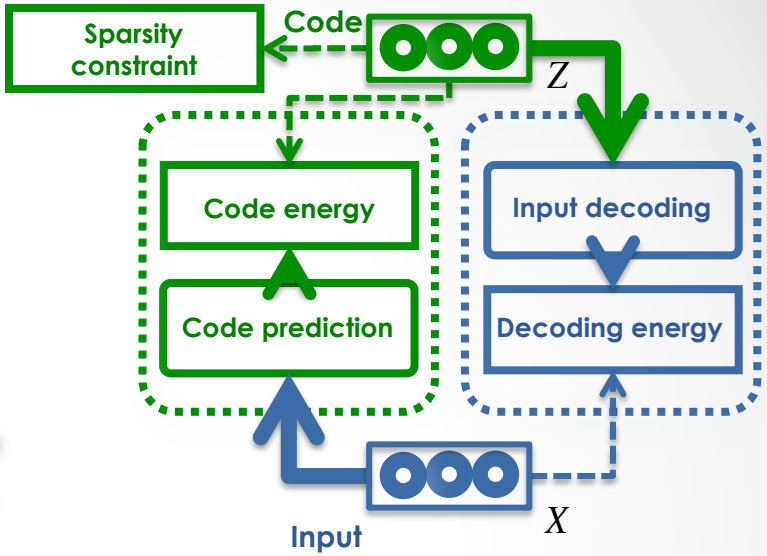





# Review of auto-encoders

**Piotr Mirowski**, Microsoft Bing London

Deep Learning Meetup
March 26, 2014

# Outline

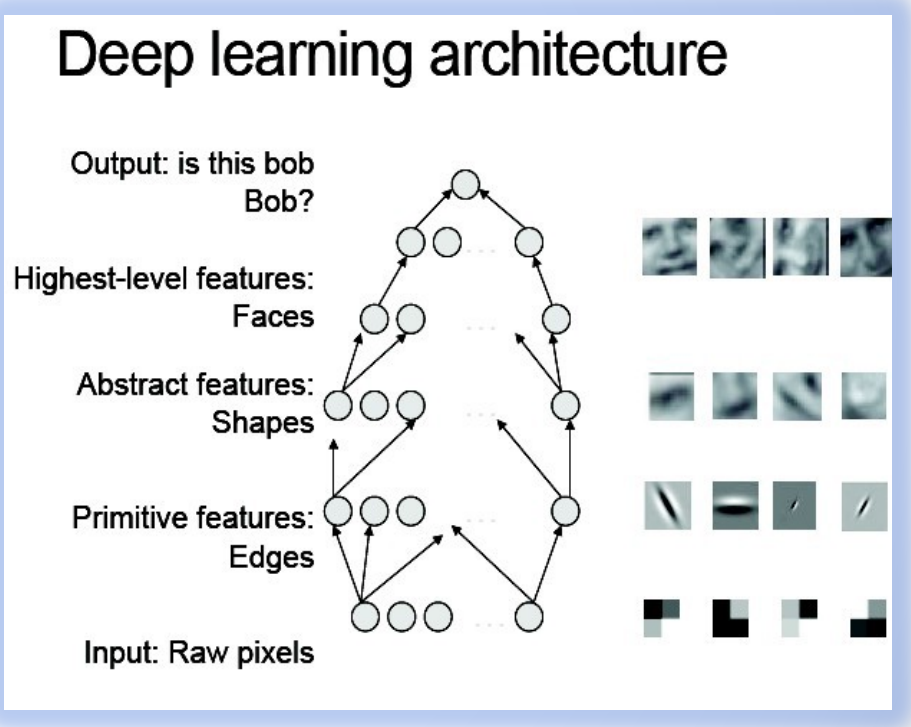
- Deep learning concepts covered
  - **Hierarchical** representations
  - **Sparse** and/or **distributed** representations
  - Supervised vs. **unsupervised** learning
- Auto-encoder
  - Architecture
  - **Inference** and **learning**
  - Sparse coding
  - Sparse auto-encoders
- Illustration: handwritten digits
  - **Stacking** auto-encoders
  - Learning representations of digits
  - Impact on **classification**
- Applications to text
  - Semantic hashing
  - Semi-supervised learning
  - Moving away from auto-encoders
- Topics not covered in this talk

# Outline

- Deep learning concepts covered
  - **Hierarchical** representations
  - **Sparse** and/or **distributed** representations
  - Supervised vs. **unsupervised** learning
- Auto-encoder
  - Architecture
  - **Inference** and **learning**
  - Sparse coding
  - Sparse auto-encoders
- Illustration: handwritten digits
  - **Stacking** auto-encoders
  - Learning representations of digits
  - Impact on **classification**
- Applications to text
  - Semantic hashing
  - Semi-supervised learning
  - Moving away from auto-encoders
- Topics not covered in this talk

# Hierarchical representations

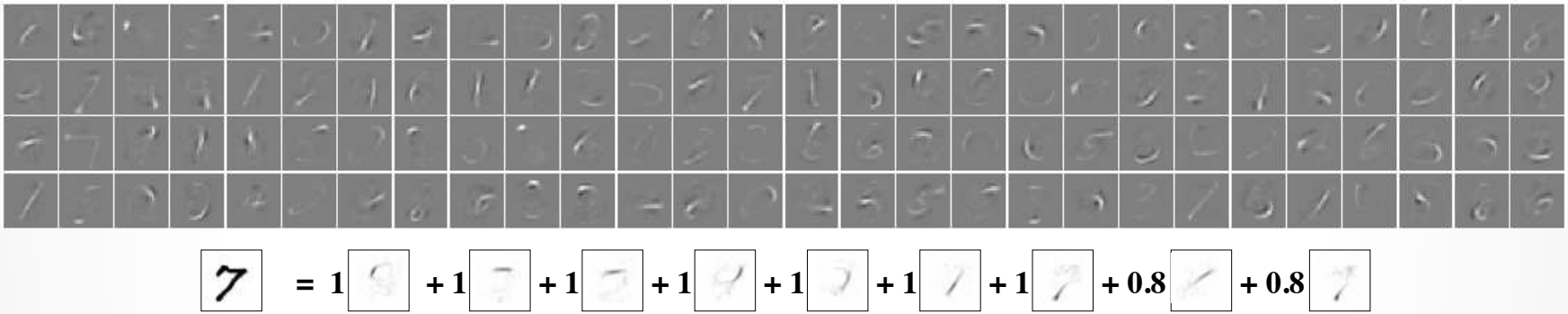
"Deep learning methods aim at **learning feature hierarchies** with features from higher levels of the hierarchy formed by the composition of lower level features.
**Automatically learning features** at multiple levels of abstraction allows a system to learn **complex functions** mapping the input to the output directly from data, without depending completely on human-crafted features."
— Yoshua Bengio




[Bengio, "On the expressive power of deep architectures", *Talk at ALT*, 2011]
[Bengio, *Learning Deep Architectures for AI*, 2009]

# Sparse and/or distributed representations

Biological motivation: V1 visual cortex



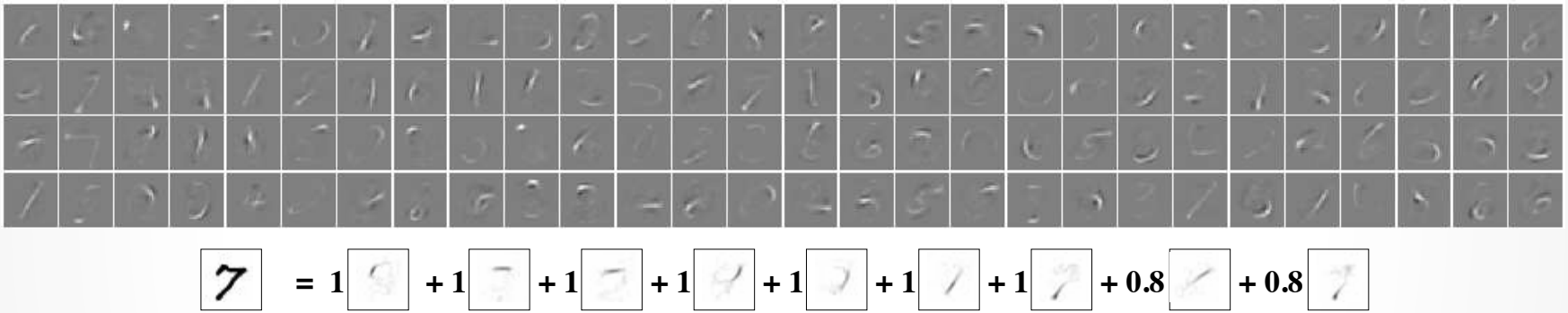

Example on MNIST handwritten digits
An image of size 28x28 pixels can be represented using a small combination of **codes** from a **basis set**.



[Ranzato, Poultney, Chopra & LeCun, "Efficient Learning of Sparse Representations with an Energy-Based Model ", *NIPS*, 2006; Ranzato, Boureau & LeCun, "Sparse Feature Learning for Deep Belief Networks ", *NIPS*, 2007]

# Sparse and/or distributed representations

Biological motivation: V1 visual cortex (backup slides)

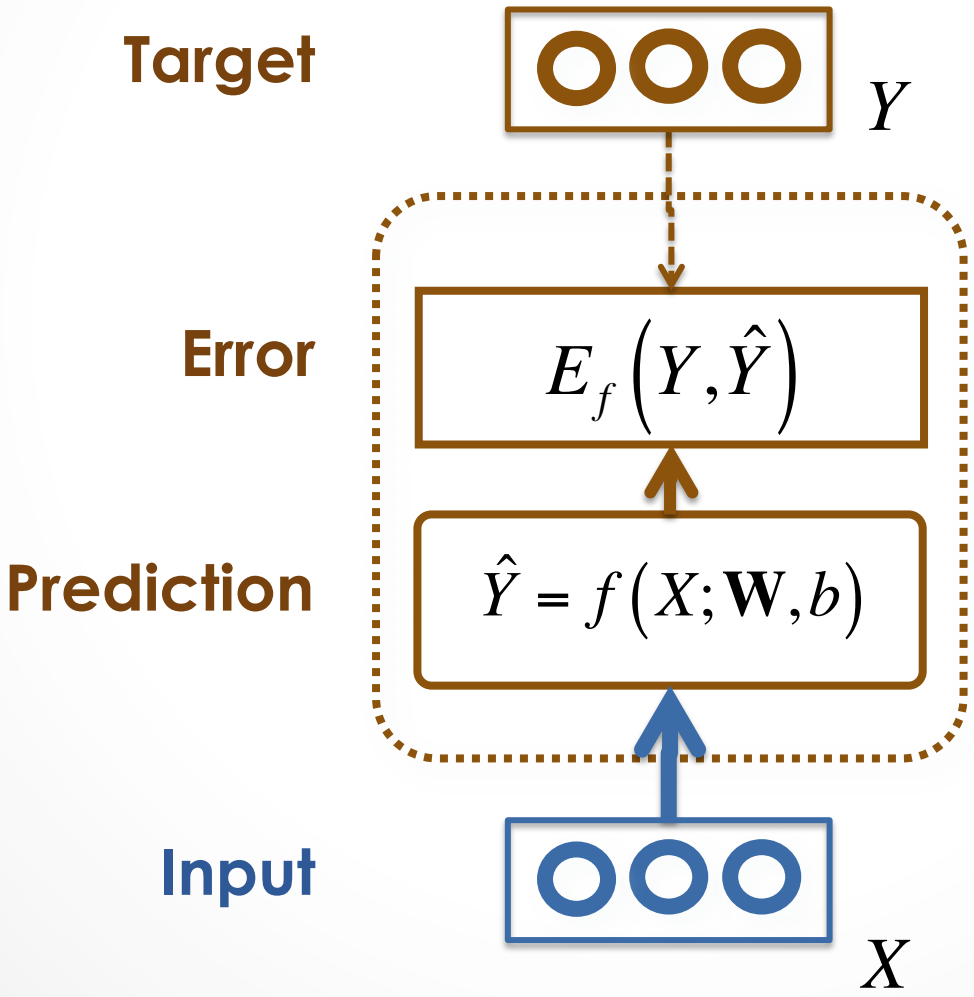



Example on MNIST handwritten digits
An image of size 28x28 pixels can be represented using a small combination of **codes** from a **basis set**.

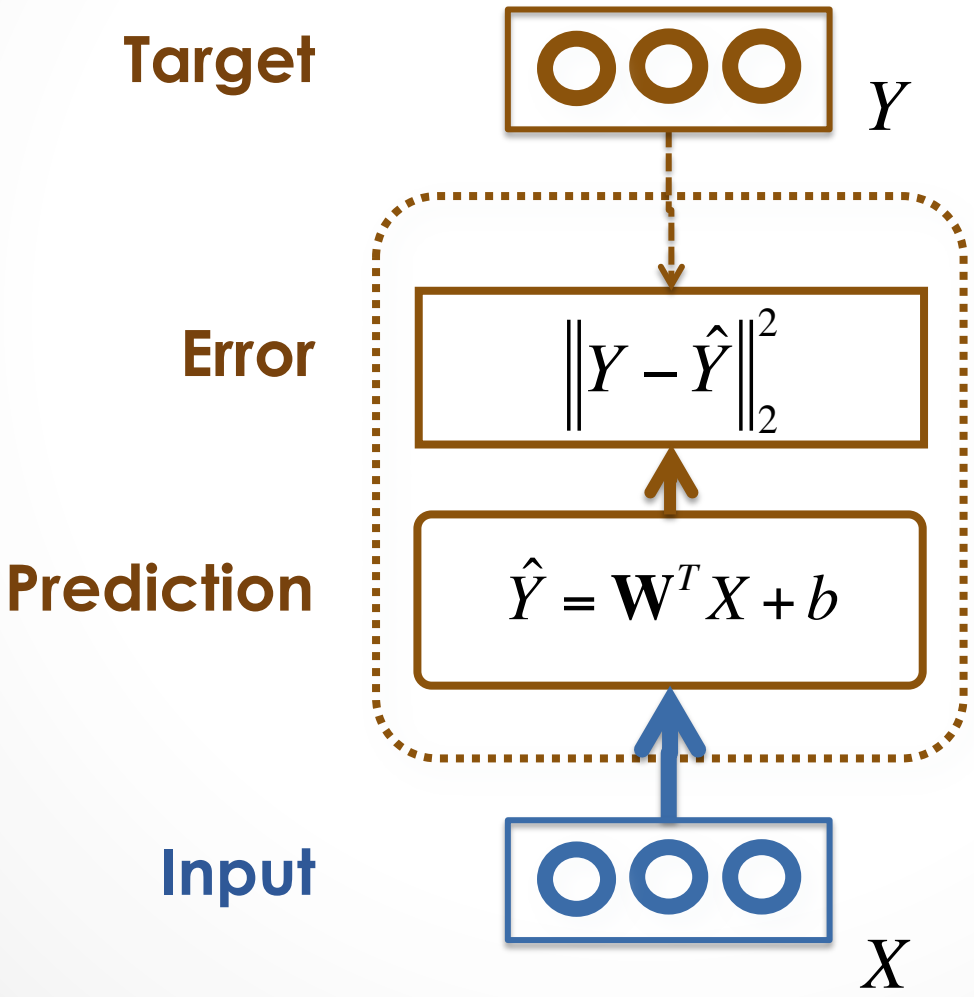
*At the end of this talk, you should know how to learn that basis set and how to infer the codes, in a 2-layer auto-encoder architecture. Matlab/Octave code and the MNIST dataset will be provided.*

[Ranzato, Poultney, Chopra & LeCun, "Efficient Learning of Sparse Representations with an Energy-Based Model ", *NIPS*, 2006; Ranzato, Boureau & LeCun, "Sparse Feature Learning for Deep Belief Networks ", *NIPS*, 2007]

# Supervised learning

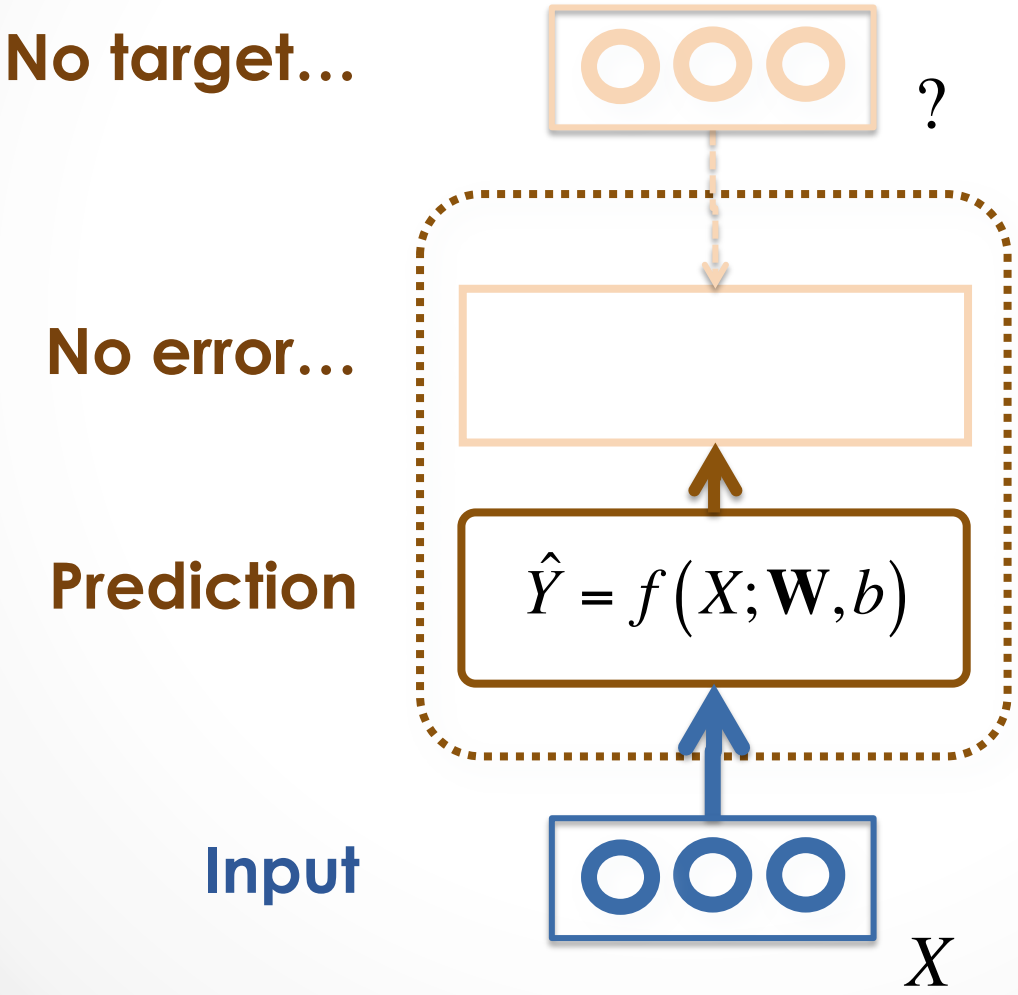


# Supervised learning

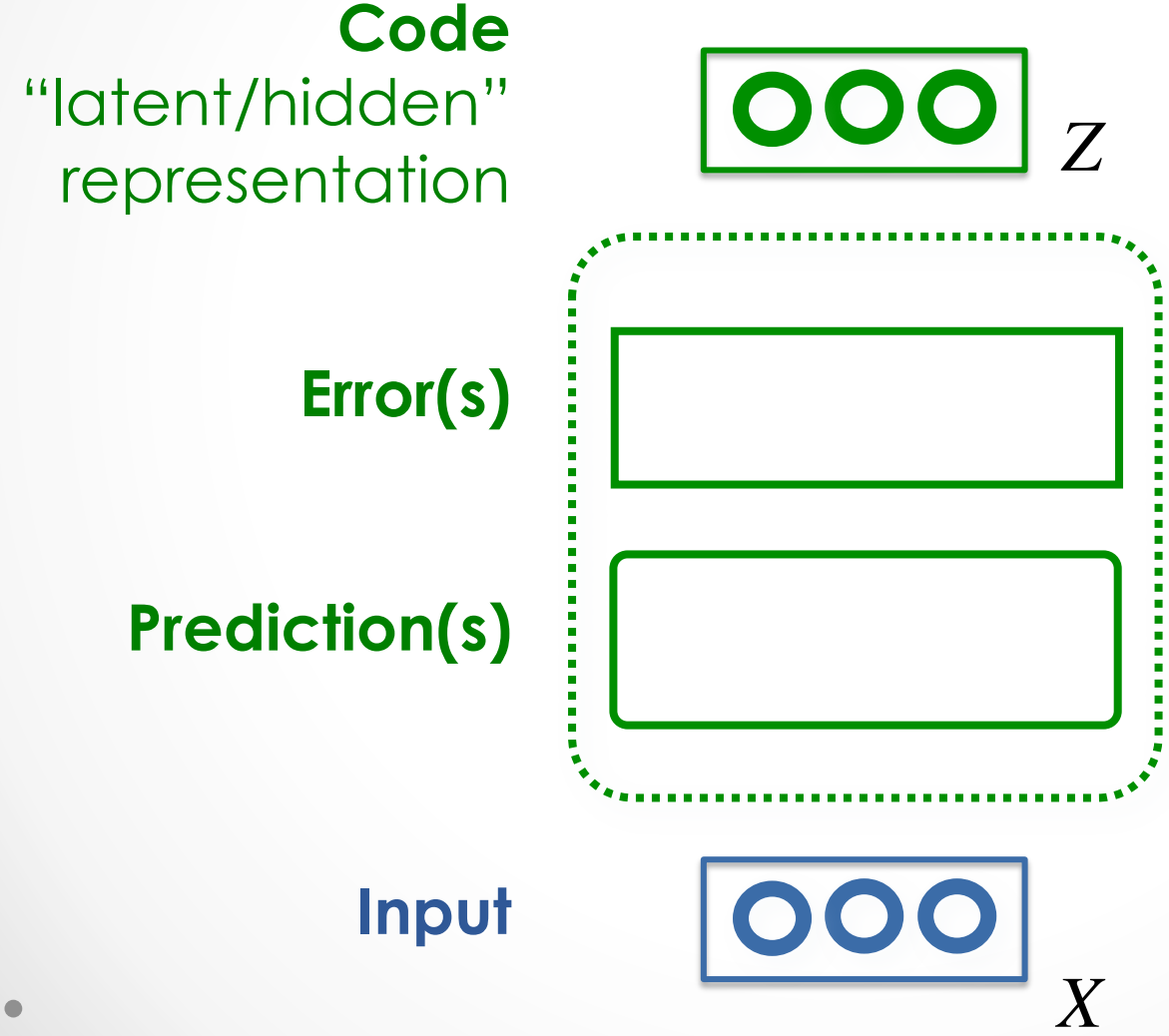


# Why not exploit unlabeled data?

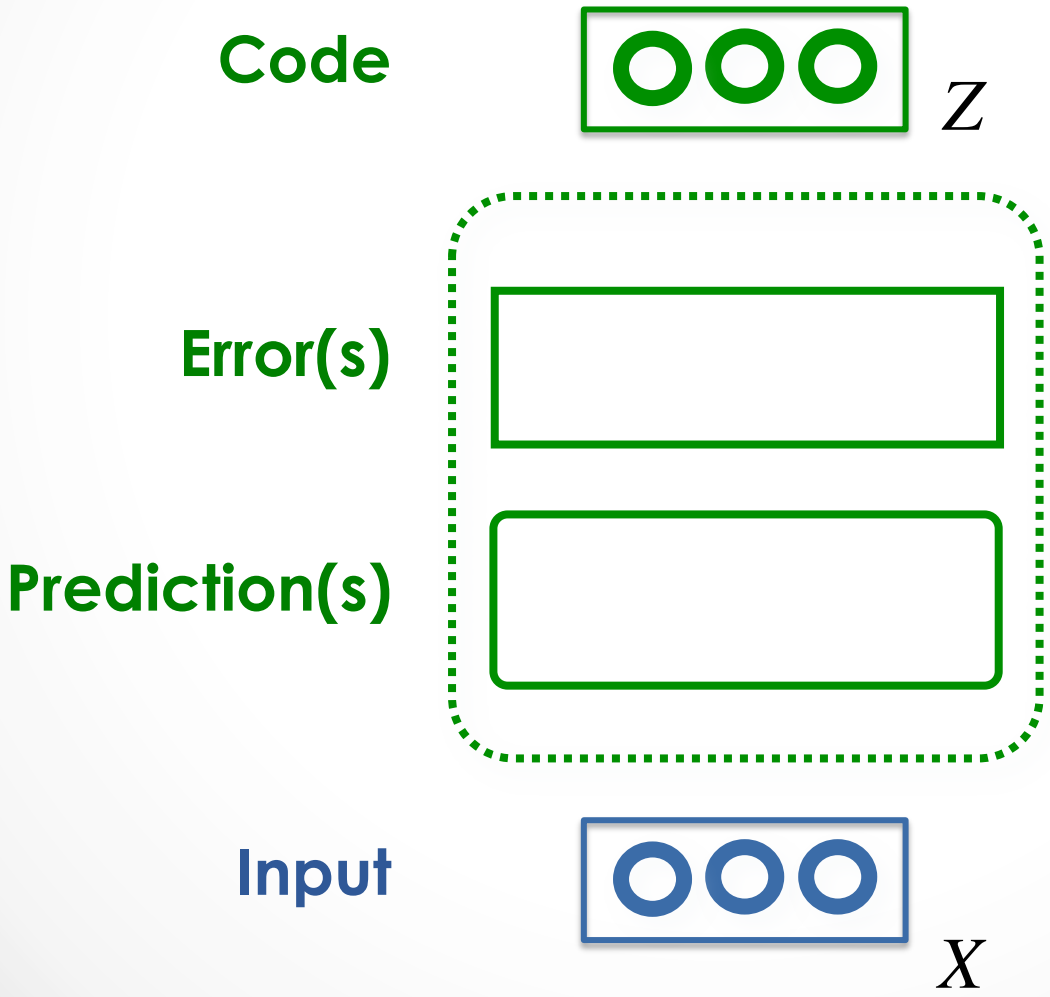
# Unsupervised learning



# Unsupervised learning



# Unsupervised learning




We want the **codes** to represent the **inputs** in the dataset.

The **code** should be a compact representation of the **inputs**: low-dimensional and/or sparse.

# Examples of unsupervised learning

- Linear decomposition of the **inputs**:
  - Principal Component Analysis and Singular Value Decomposition
  - Independent Component Analysis [Bell & Sejnowski, 1995]
  - **Sparse coding** [Olshausen & Field, 1997]
  - …
- Fitting a distribution to the **inputs**:
  - Mixtures of Gaussians
  - Use of **Expectation-Maximization algorithm** [Dempster et al, 1977]
  - …
- For text or discrete data:
  - Latent Semantic Indexing [Deerwester et al, 1990]
  - Probabilistic Latent Semantic Indexing [Hofmann et al, 1999]
  - Latent Dirichlet Allocation [Blei et al, 2003]
  - **Semantic Hashing**
  - …

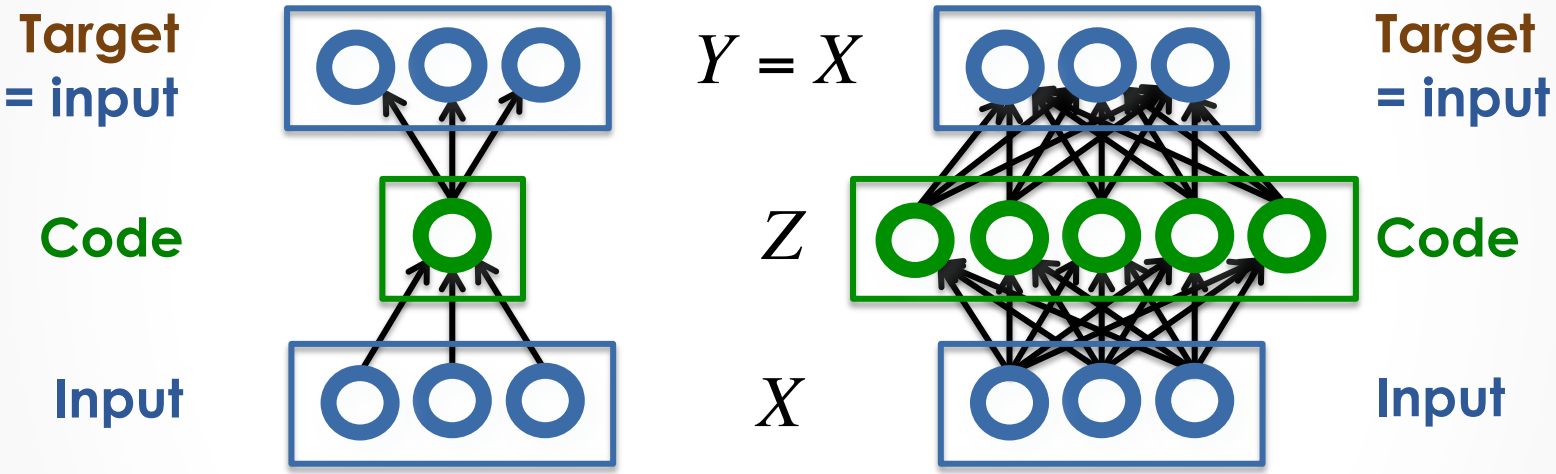
# Objective of this tutorial

Study a fundamental building block
for deep learning,
the **auto-encoder**

# Outline

- Deep learning concepts covered
  - **Hierarchical** representations
  - **Sparse** and/or **distributed** representations
  - Supervised vs. **unsupervised** learning
- Auto-encoder
  - Architecture
  - **Inference** and **learning**
  - Sparse coding
  - Sparse auto-encoders
- Illustration: handwritten digits
  - **Stacking** auto-encoders
  - Learning representations of digits
  - Impact on **classification**
- Applications to text
  - Semantic hashing
  - Semi-supervised learning
  - Moving away from auto-encoders
- Topics not covered in this talk

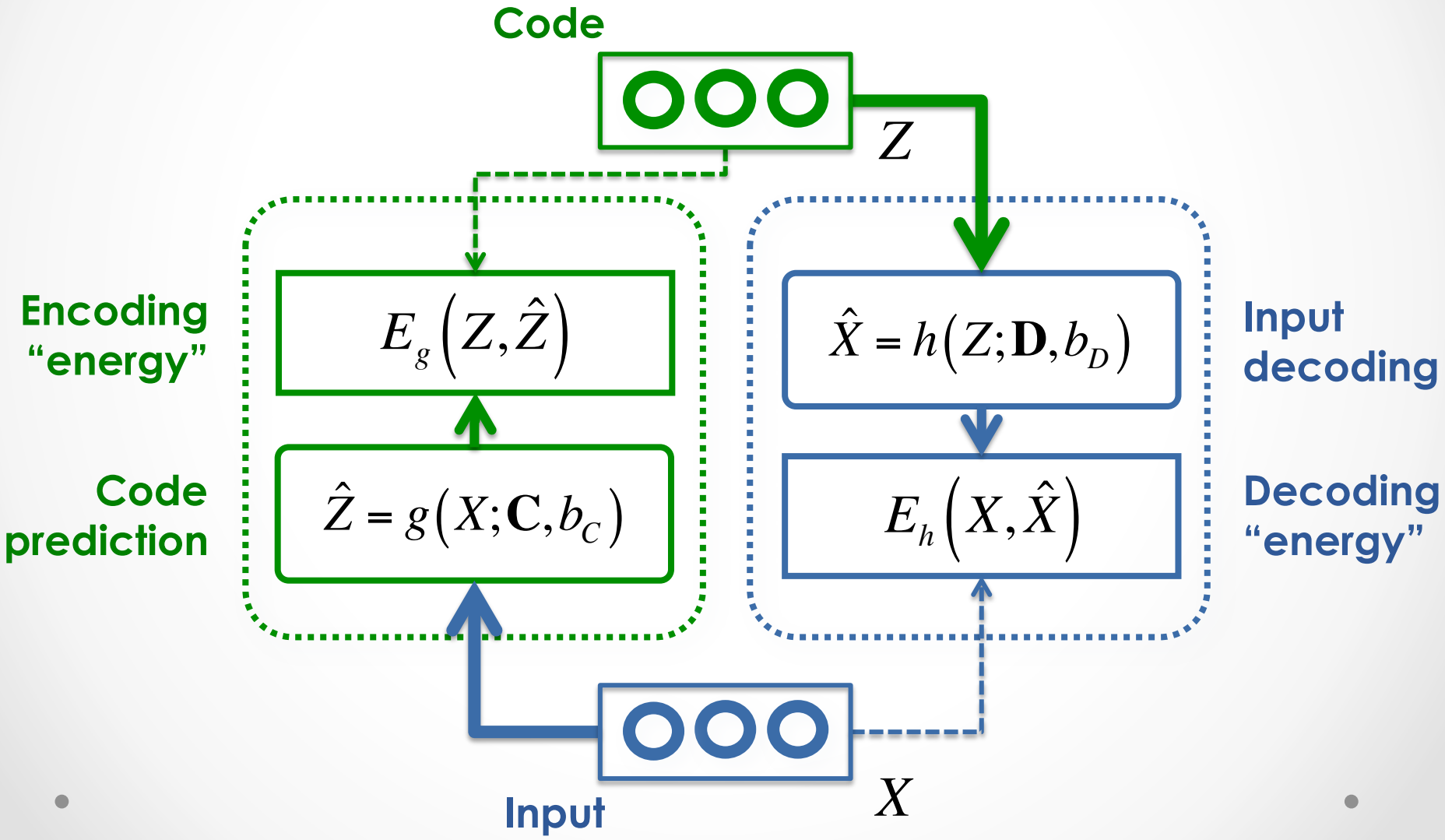
# Auto-encoder



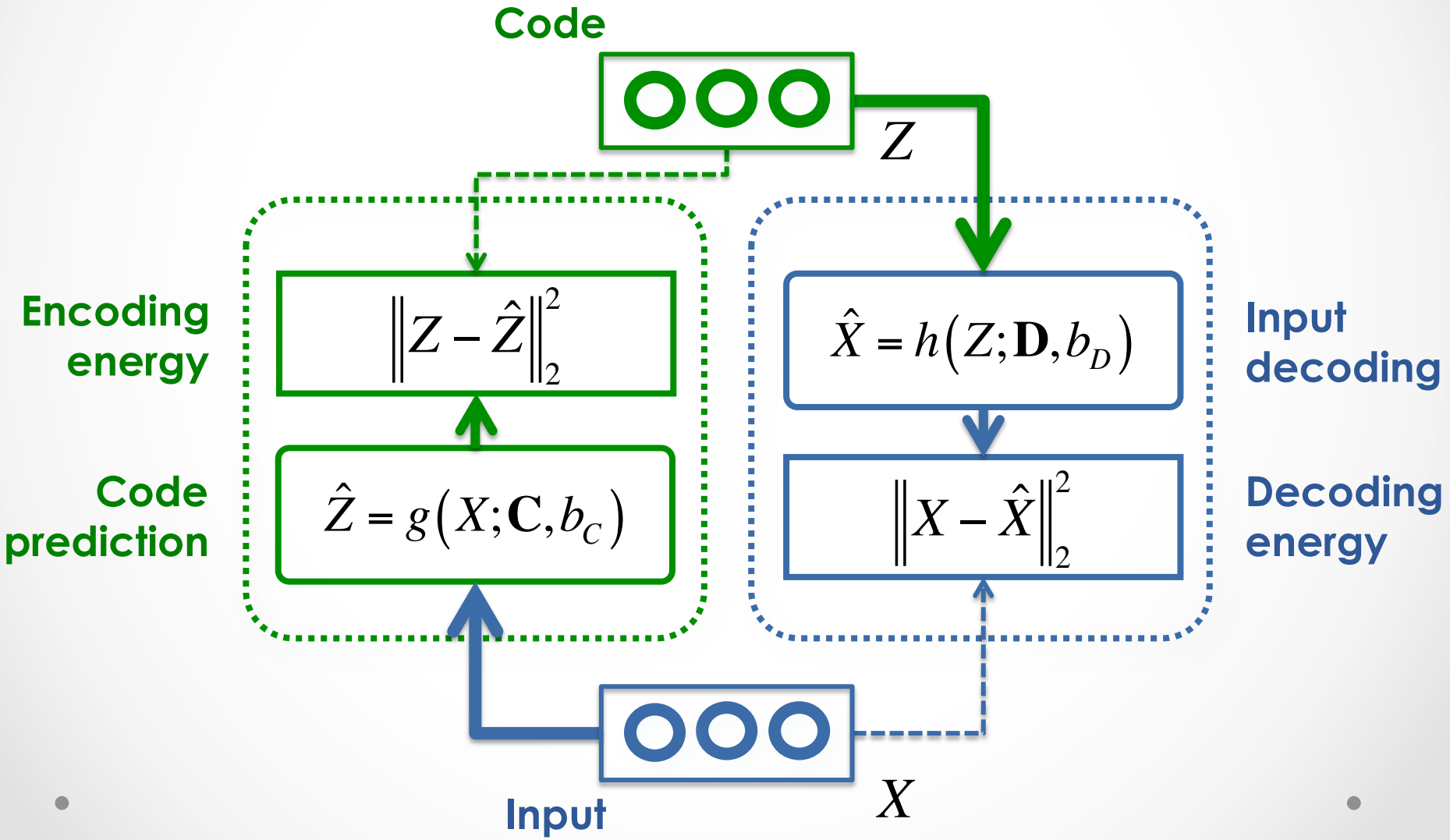

**"Bottleneck" code**
i.e., low-dimensional, typically dense, distributed representation

**"Overcomplete" code**
i.e., high-dimensional, always sparse, distributed representation

# Auto-encoder



# Auto-encoder



# Auto-encoder loss function

For one sample *t*

$$L\left(X(t),Z(t);\mathbf{W}\right)=\alpha\left\|Z(t)-g\left(X(t);\mathbf{C},b_C\right)\right\|_2^2+\left\|X(t)-h\left(Z(t);\mathbf{D},b_D\right)\right\|_2^2$$

coefficient of the encoder error

Encoding energy

Decoding energy

For all *T* samples

$$L\left(\mathbf{X},\mathbf{Z};\mathbf{W}\right)=\sum_{t=1}^{T}\alpha\left\|Z(t)-g\left(X(t);\mathbf{C},b_C\right)\right\|_2^2+\sum_{t=1}^{T}\left\|X(t)-h\left(Z(t);\mathbf{D},b_D\right)\right\|_2^2$$

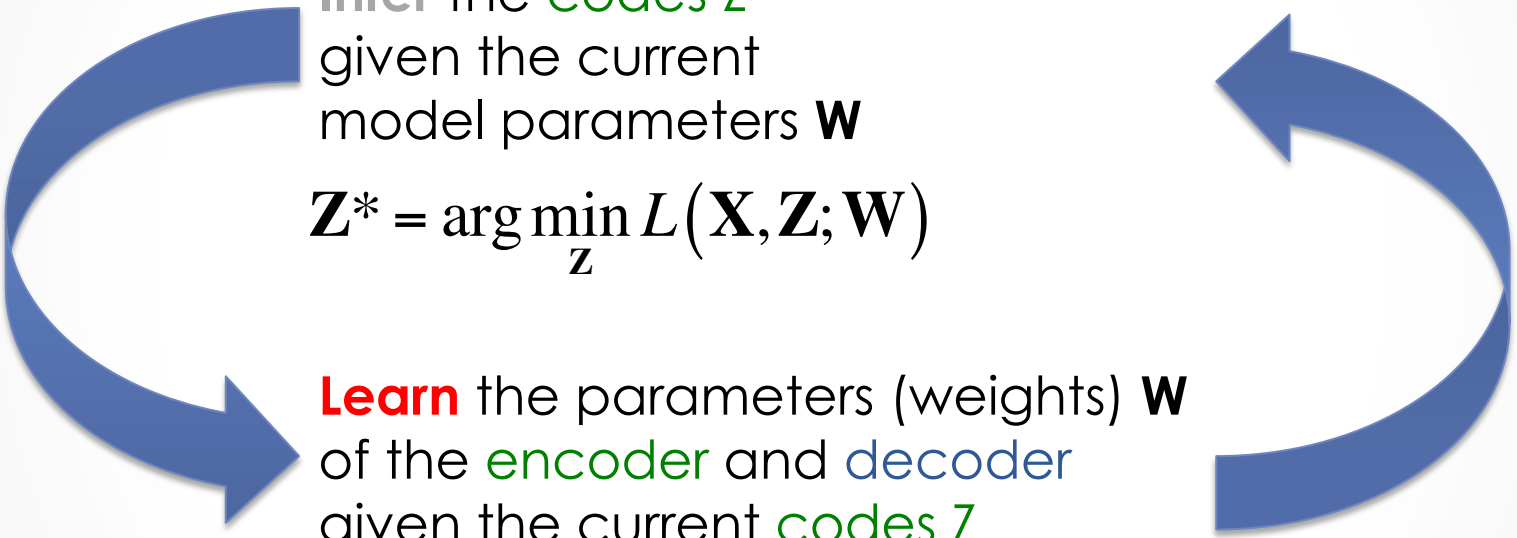
Encoding energy

Decoding energy

**How do we get the codes Z?**

We note **W**={**C**, $b_C$, **D**, $b_D$}

# Learning and inference in auto-encoders



**Infer** the codes Z
given the current
model parameters **W**

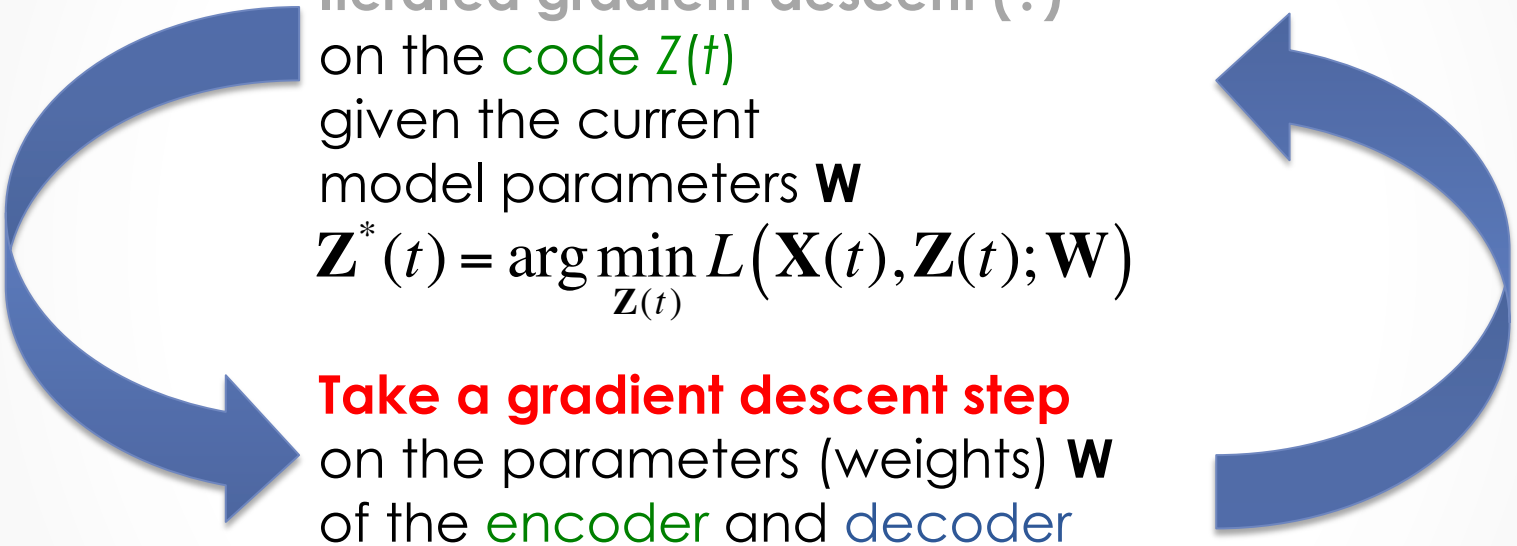
$$\mathbf{Z}^* = \arg\min_{\mathbf{Z}} L(\mathbf{X}, \mathbf{Z}; \mathbf{W})$$

**Learn** the parameters (weights) **W**
of the encoder and decoder
given the current codes Z

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} L(\mathbf{X}, \mathbf{Z}; \mathbf{W})$$

Relationship to Expectation-Maximization
in graphical models (backup slides)

# Learning and inference: stochastic gradient descent



**Iterated gradient descent (?)**
on the code Z(*t*)
given the current
model parameters **W**

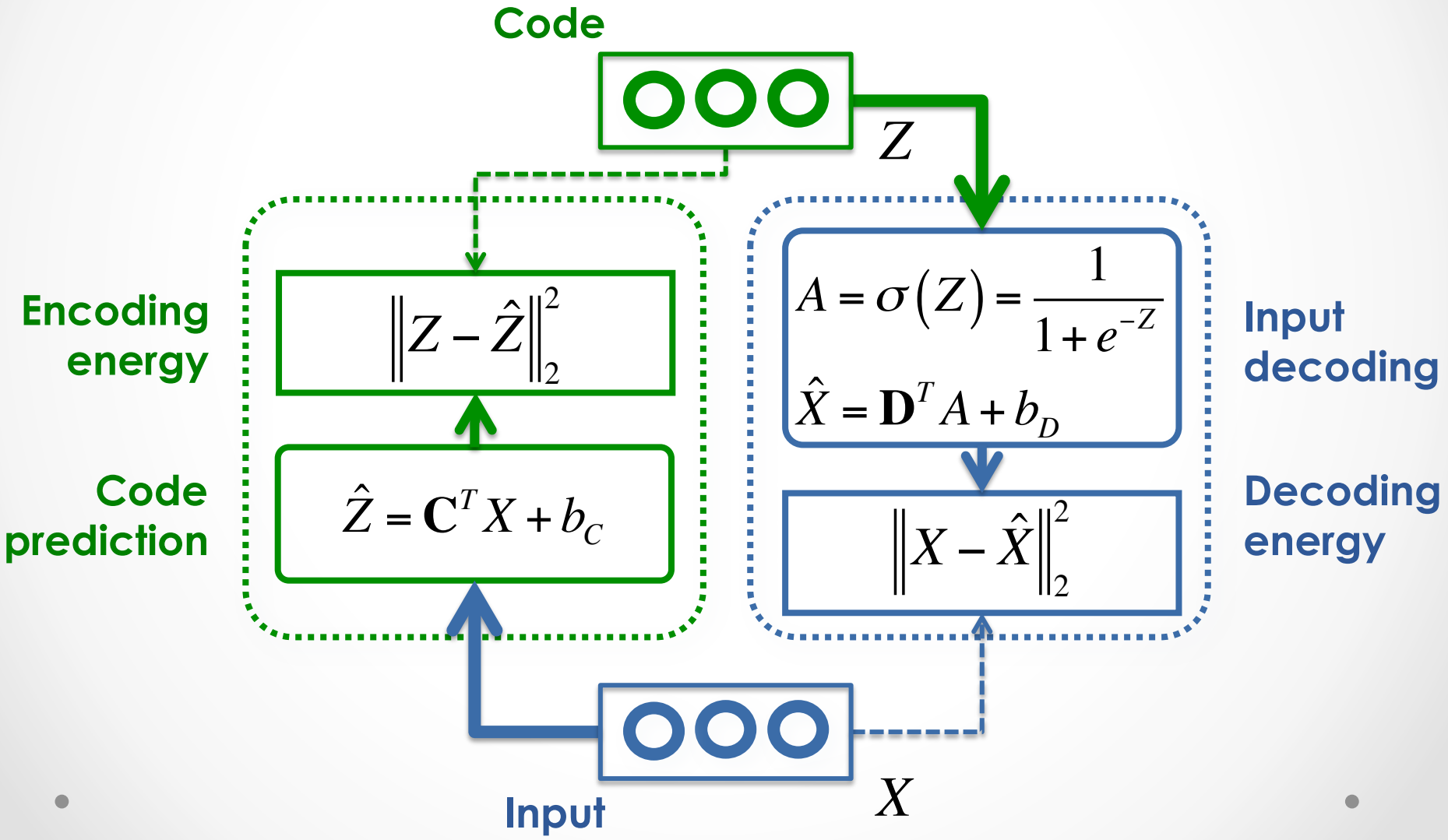
$$\mathbf{Z}^*(t) = \arg\min_{\mathbf{Z}(t)} L\left(\mathbf{X}(t), \mathbf{Z}(t); \mathbf{W}\right)$$

**Take a gradient descent step**
on the parameters (weights) **W**
of the encoder and decoder
given the current codes Z

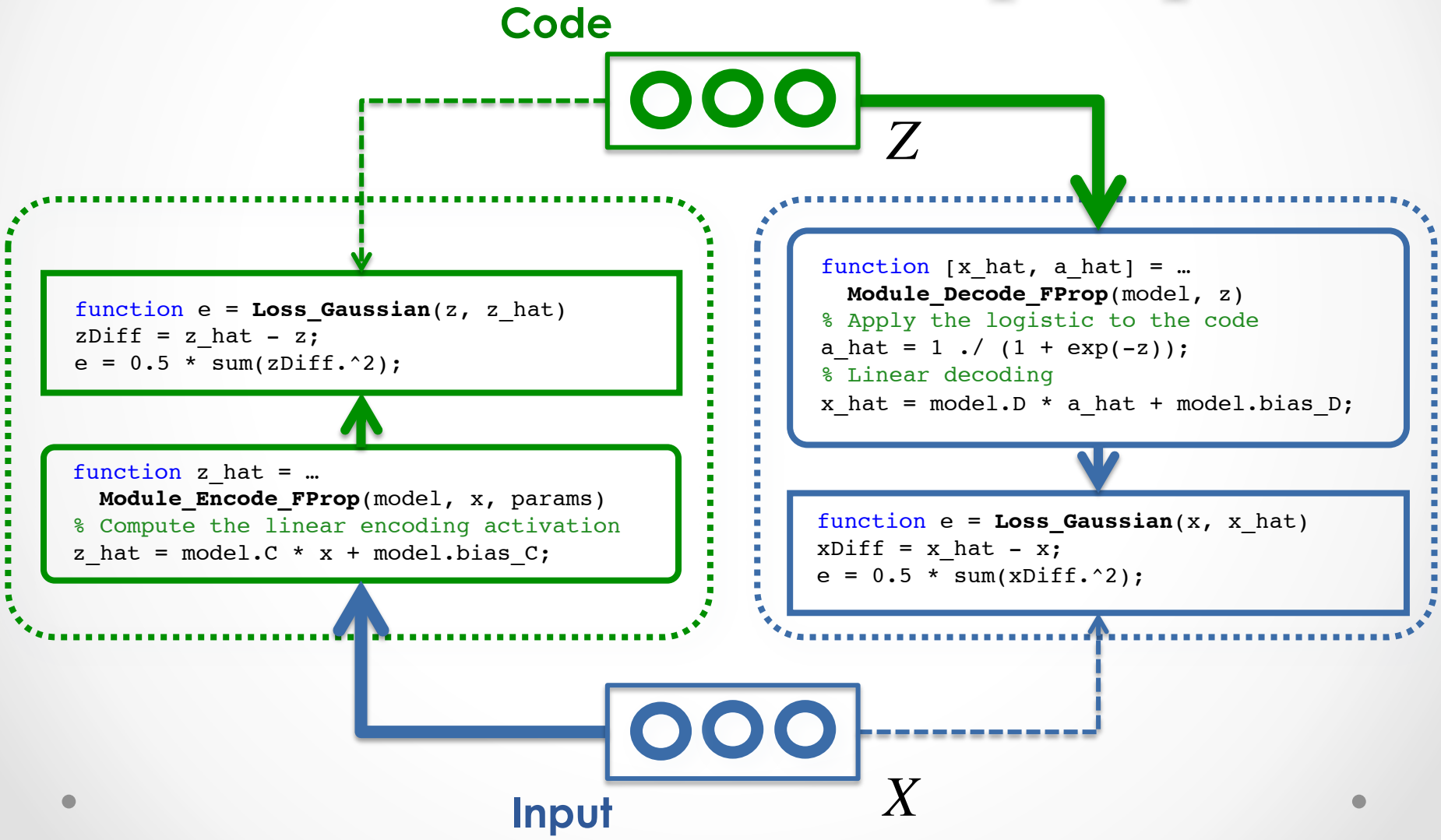
$$L\left(\mathbf{X}(t), \mathbf{Z}(t); \mathbf{W}^*\right) < L\left(\mathbf{X}(t), \mathbf{Z}(t); \mathbf{W}\right)$$

Relationship to Generalized EM
in graphical models (backup slides)

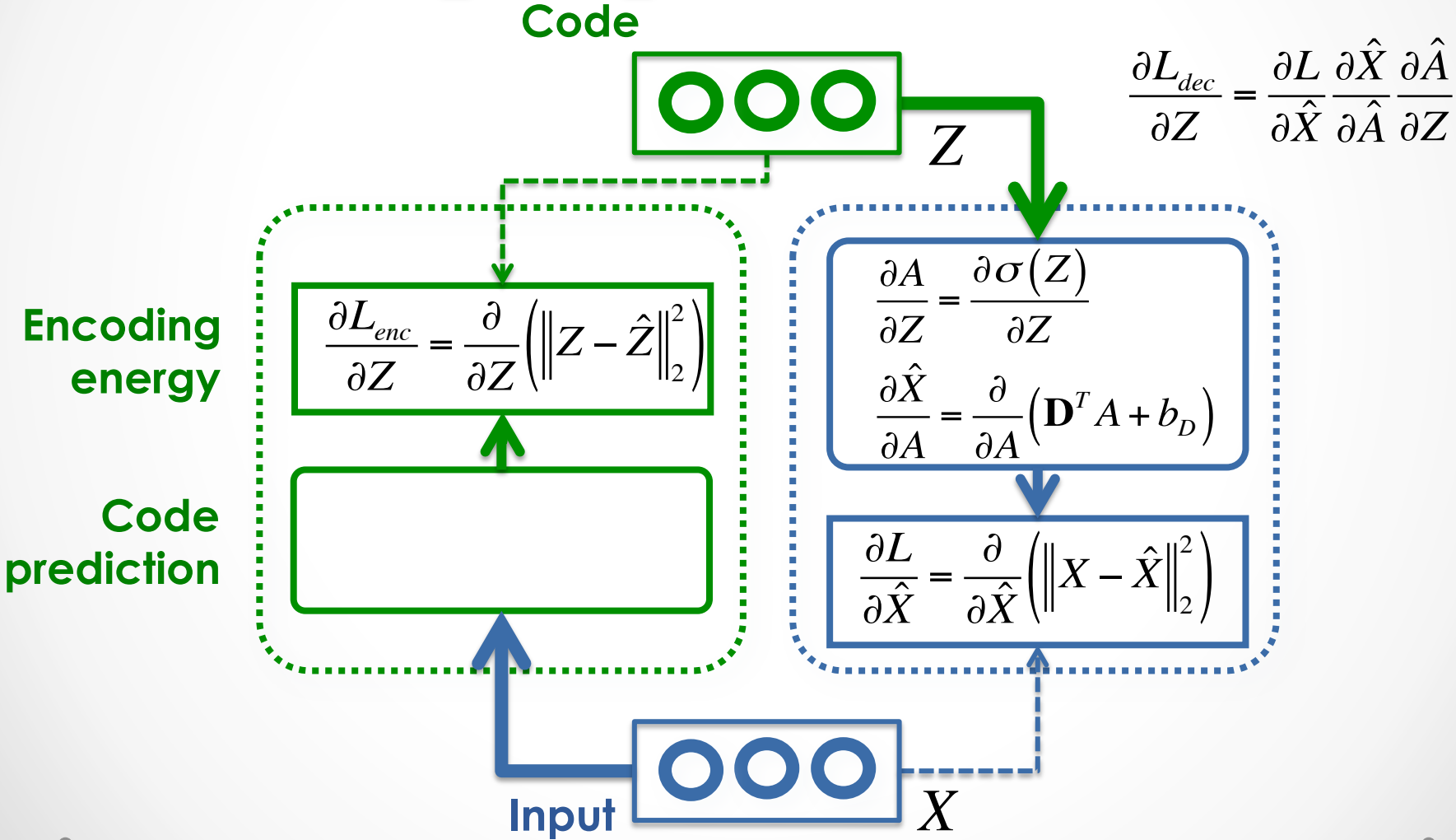
# Auto-encoder



# Auto-encoder: fprop



# Auto-encoder backprop w.r.t. codes




$$\frac{\partial L_{dec}}{\partial Z} = \frac{\partial L}{\partial \hat{X}} \frac{\partial \hat{X}}{\partial \hat{A}} \frac{\partial \hat{A}}{\partial Z}$$

$$\frac{\partial L_{enc}}{\partial Z} = \frac{\partial}{\partial Z}\left(\left\|Z - \hat{Z}\right\|_2^2\right)$$

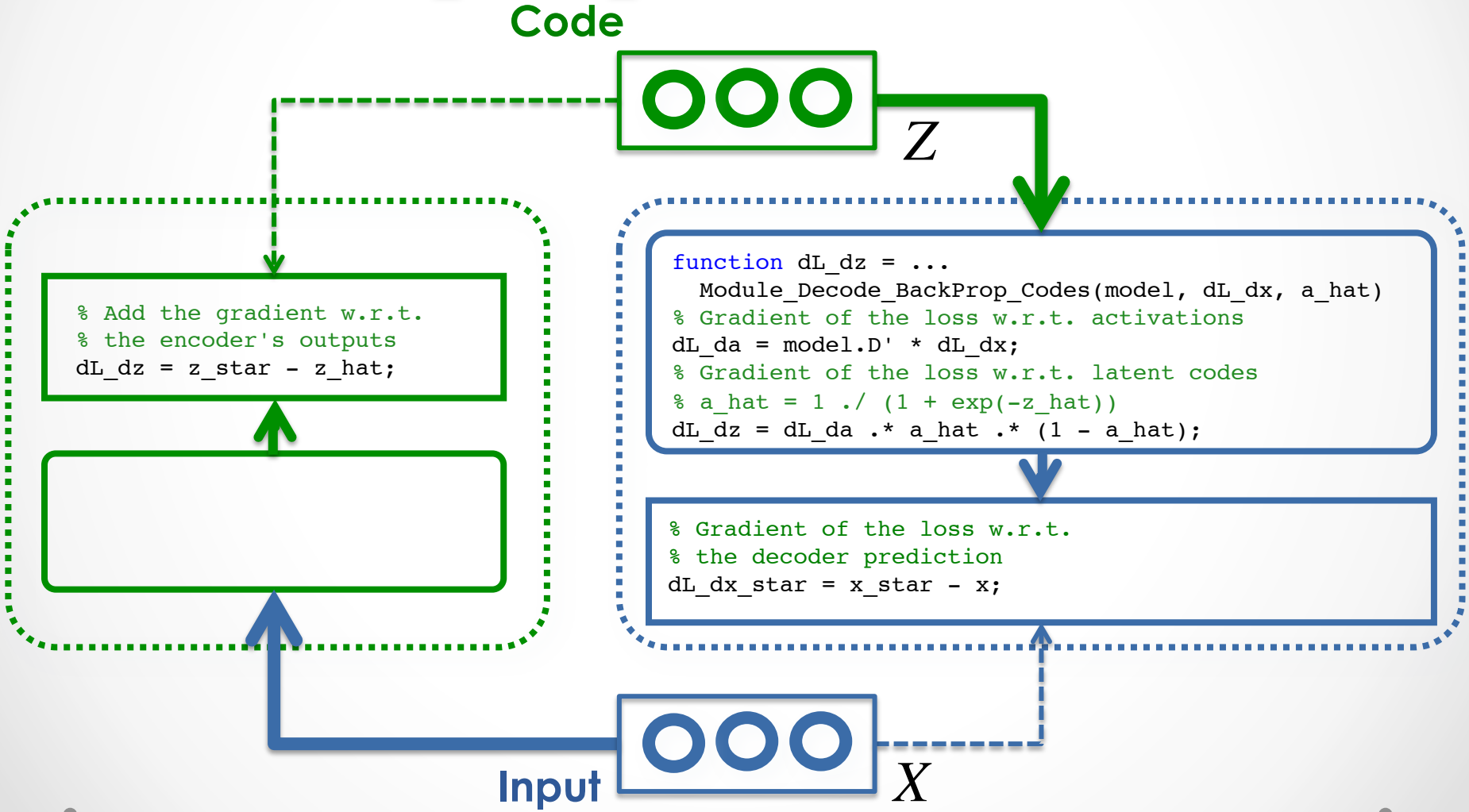
$$\frac{\partial A}{\partial Z} = \frac{\partial \sigma(Z)}{\partial Z}$$

$$\frac{\partial \hat{X}}{\partial A} = \frac{\partial}{\partial A}\left(\mathbf{D}^T A + b_D\right)$$

$$\frac{\partial L}{\partial \hat{X}} = \frac{\partial}{\partial \hat{X}}\left(\left\|X - \hat{X}\right\|_2^2\right)$$

[Ranzato, Boureau & LeCun, "Sparse Feature Learning for Deep Belief Networks ", *NIPS*, 2007]

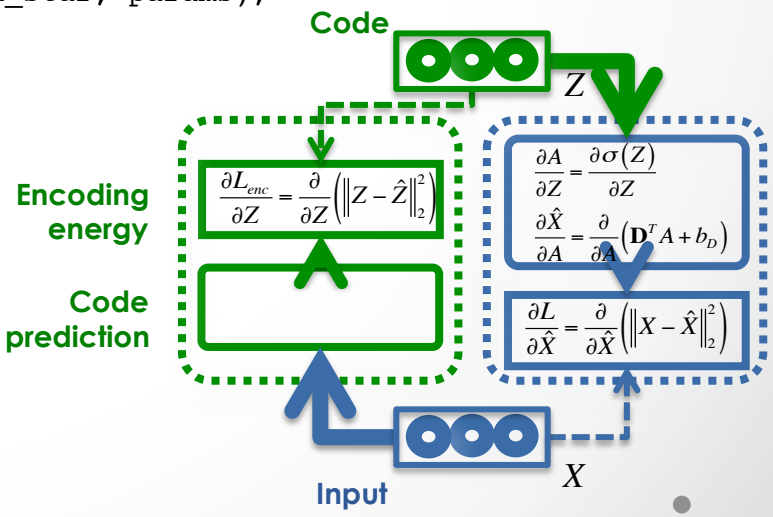
# Auto-encoder: backprop w.r.t. codes




[Ranzato, Boureau & LeCun, "Sparse Feature Learning for Deep Belief Networks ", *NIPS*, 2007]

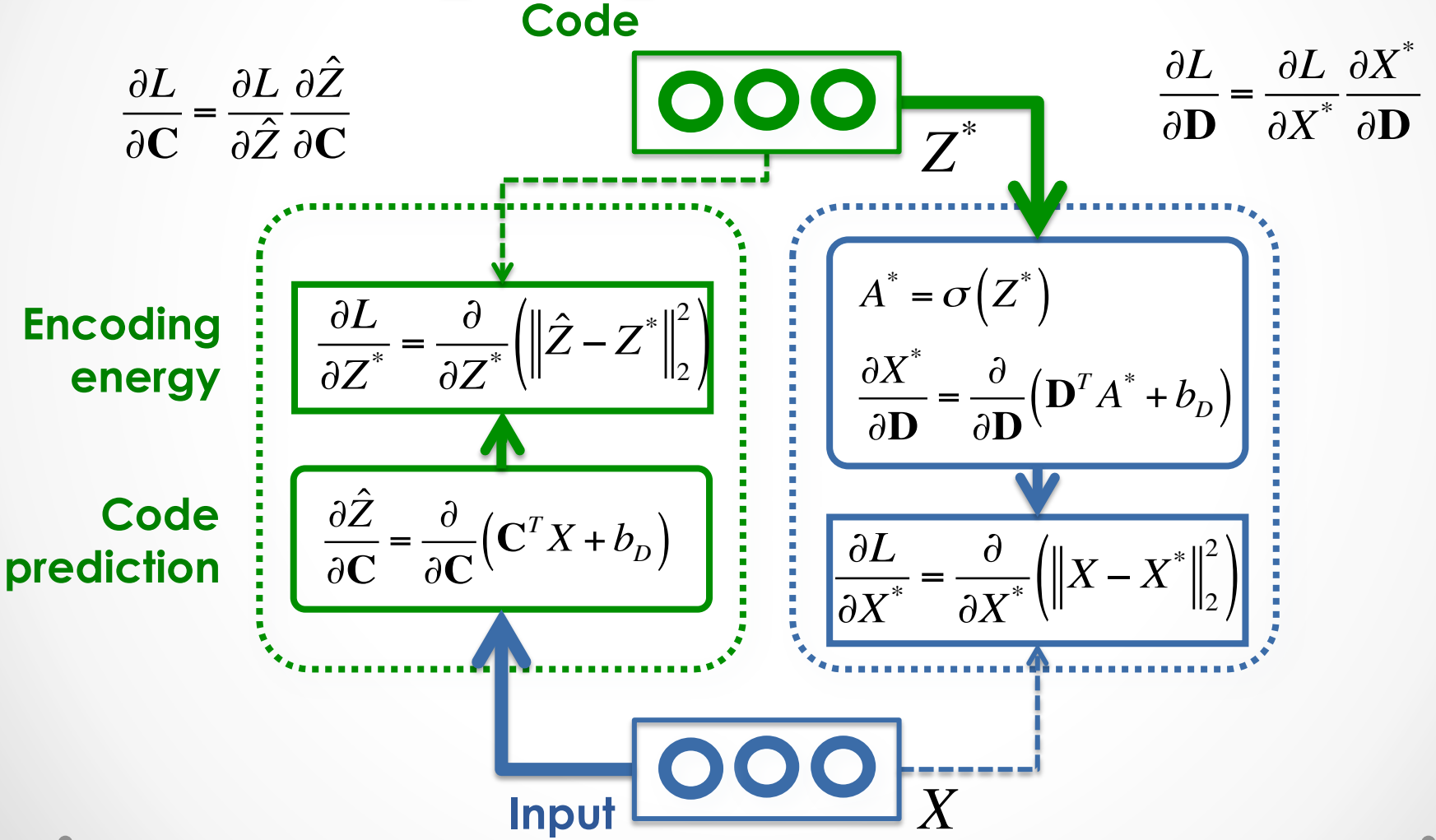
# Code inference in the auto-encoder

```
function [z_star, z_hat, loss_star, loss_hat] = Layer_Infer(model, x, params)
% Encode the current input and initialize the latent code
z_hat = Module_Encode_FProp(model, x, params);
% Decode the current latent code
[x_hat, a_hat] = Module_Decode_FProp(model, z_hat);
% Compute the current loss term due to decoding (encoding loss is 0)
loss_hat = Loss_Gaussian(x, x_hat);
% Relaxation on the latent code: loop until convergence
x_star = x_hat; a_star = a_hat; z_star = z_hat; loss_star = loss_hat;
while (true)
  % Gradient of the loss function w.r.t. decoder prediction
  dL_dx_star = x_star - x;
  % Back-propagate the gradient of the loss onto the codes
  dL_dz = Module_Decode_BackProp_Codes(model, dL_dx_star, a_star, params);
  % Add the gradient w.r.t. the encoder's outputs
  dL_dz = dL_dz + params.alpha_c * (z_star - z_hat);
  % Perform one step of gradient descent on the codes
  z_star = z_star - params.eta_z * dL_dz;
  % Decode the current latent code
  [x_star, a_star] = Module_Decode_FProp(model, z_star);
  % Compute the current loss and convergence criteria
  loss_star = Loss_Gaussian(x, x_star) + ...
    params.alpha_c * Loss_Gaussian(z_star, z_hat);
  % Stopping criteria
  [...]
end
```



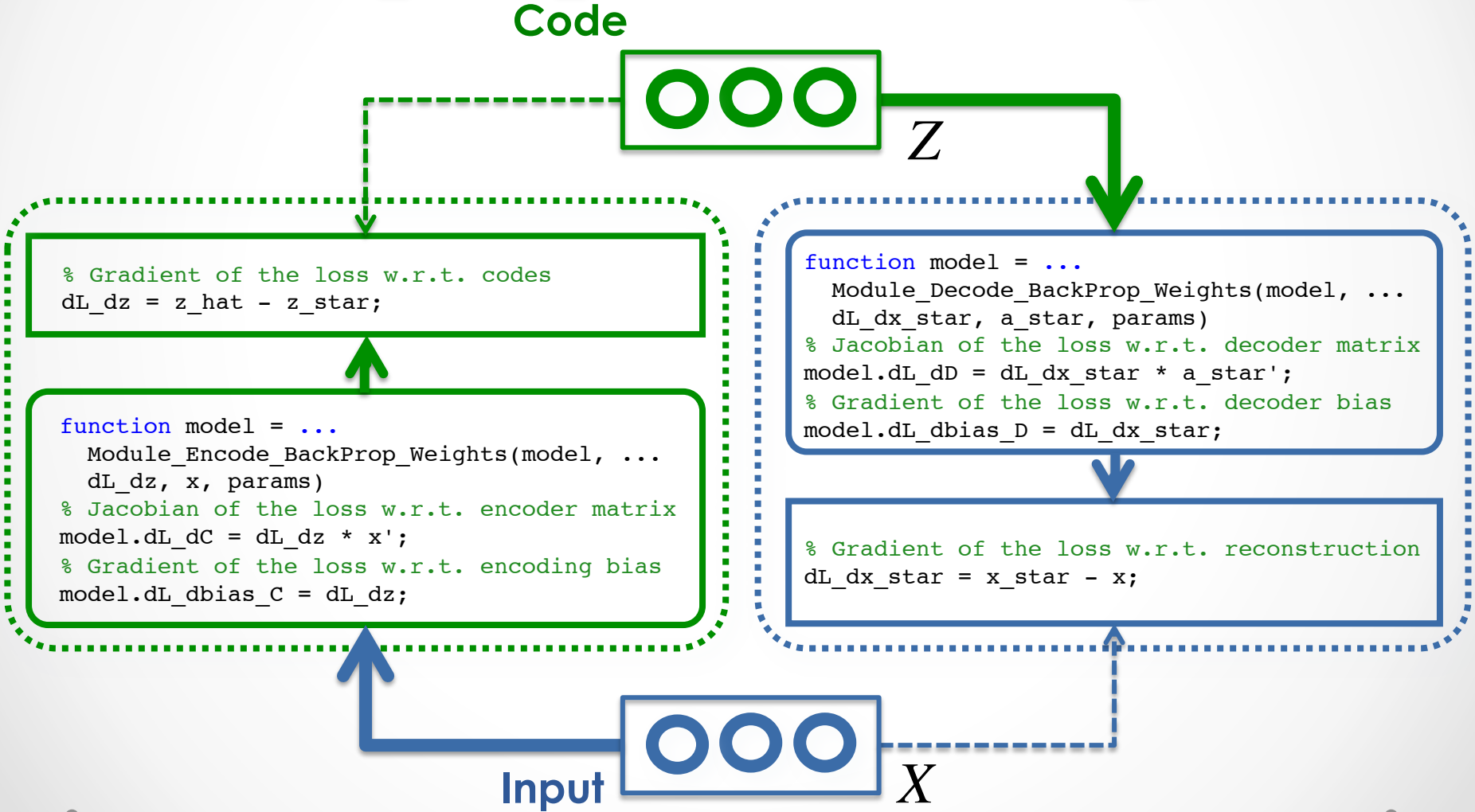

[Ranzato, Boureau & LeCun, "Sparse Feature Learning for Deep Belief Networks ", *NIPS*, 2007]

# Auto-encoder backprop w.r.t. codes




[Ranzato, Boureau & LeCun, "Sparse Feature Learning for Deep Belief Networks ", *NIPS*, 2007]

# Auto-encoder: backprop w.r.t. weights

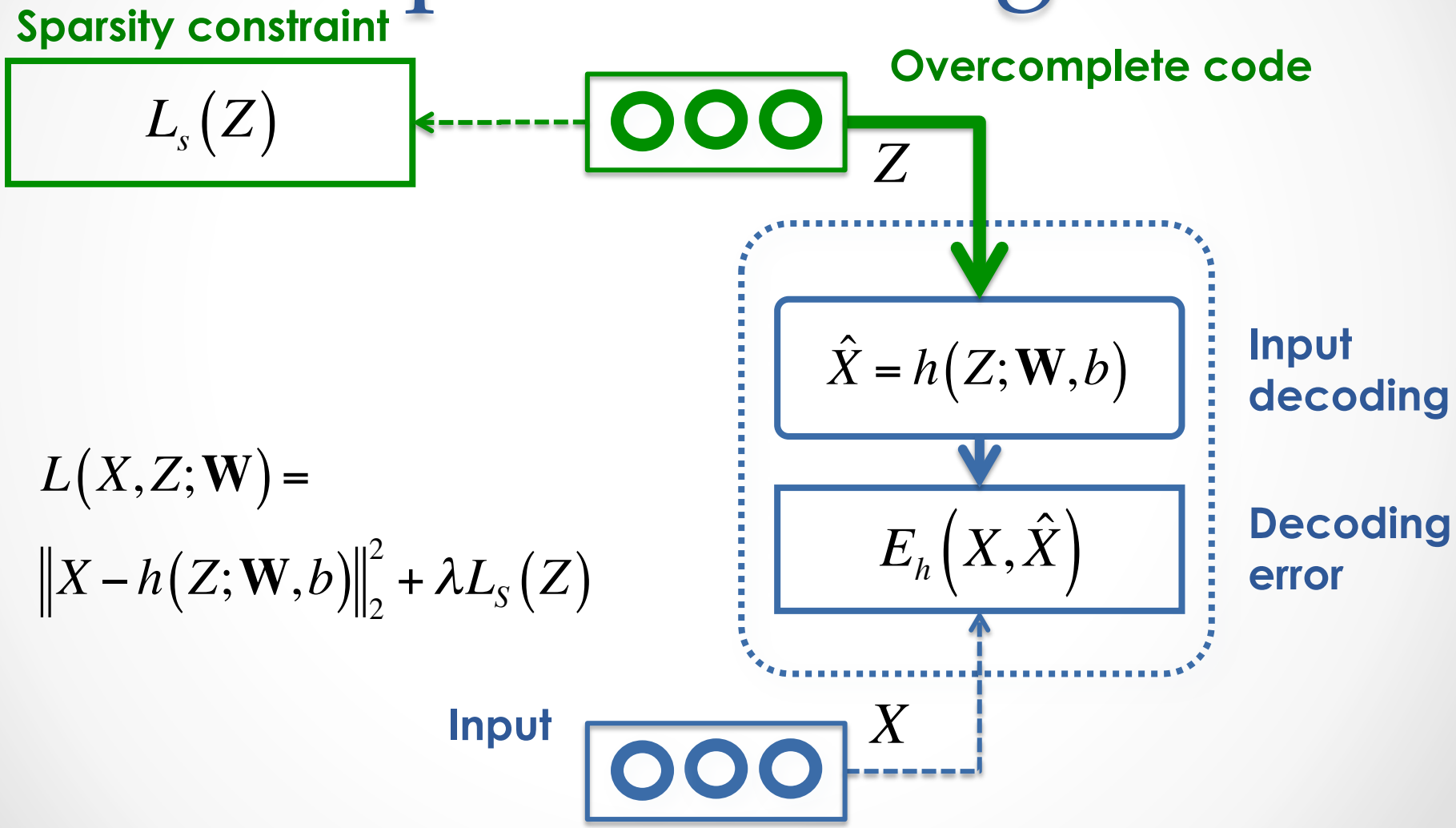



[Ranzato, Boureau & LeCun, "Sparse Feature Learning for Deep Belief Networks ", *NIPS*, 2007]

# Usual tricks about classical SGD

- Regularization (L1-norm or L2-norm) of the parameters?
- Learning rate?
- Learning rate decay?
- Momentum term on the parameters?
- Choice of the learning hyperparameters
  - Cross-validation?

# Sparse coding




$$L(X,Z;\mathbf{W}) = \left\|X - h(Z;\mathbf{W},b)\right\|_2^2 + \lambda L_S(Z)$$

[Olshausen & Field, "Sparse coding with an overcomplete basis set: a strategy employed by V1? ", *Vision Research*, 1997]

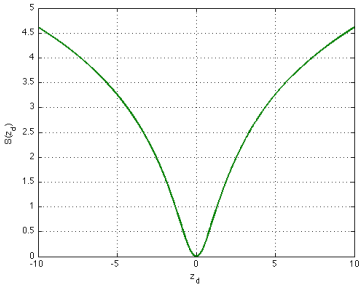
# Sparse coding

**Sparsity constraint**

$$\sum_{d=1}^{M} \log\left(1 + z_d^2\right)$$

**Overcomplete code**

$Z$



$$\hat{X} = \mathbf{W}^T Z + b$$

**Input decoding**

$$\left\| X - \hat{X} \right\|_2^2$$

**Decoding error**

$$L(X, Z; \mathbf{W}) =$$

$$\left\| X - \left(\mathbf{W}^T Z + b\right) \right\|_2^2 + \lambda L_S(Z)$$

**Input**

$X$

[Olshausen & Field, "Sparse coding with an overcomplete basis set: a strategy employed by V1? ", *Vision Research*, 1997]

# Limitations of sparse coding

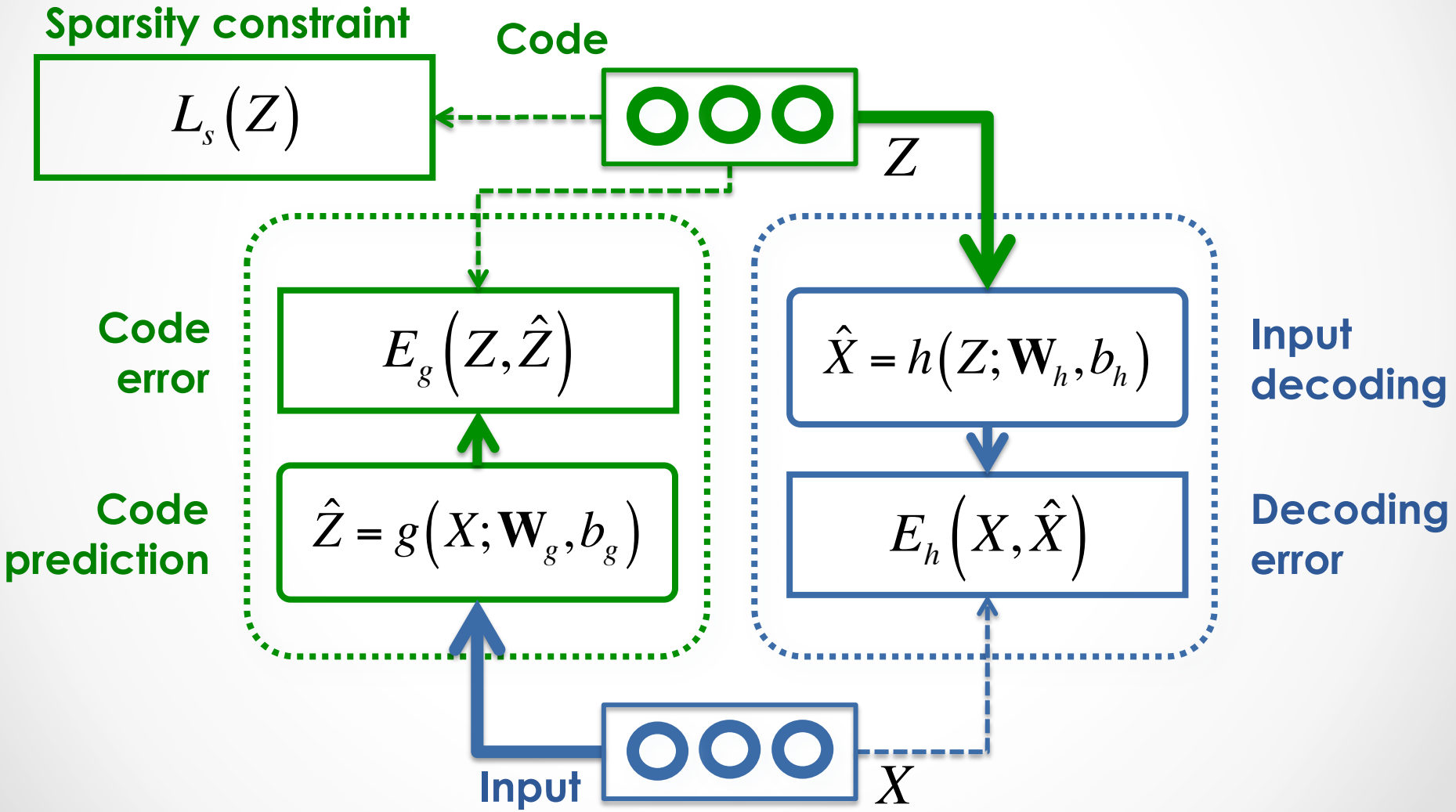
- At runtime, assuming a trained model W, **inferring the code Z** given an input sample X is **expensive**
- Need a **tweak** on the **model weights W**: normalize the columns of W to unit length after each learning step
- Otherwise:

code pulled to 0
by sparsity constraint

$$\hat{X} = \mathbf{W}^T Z + b$$

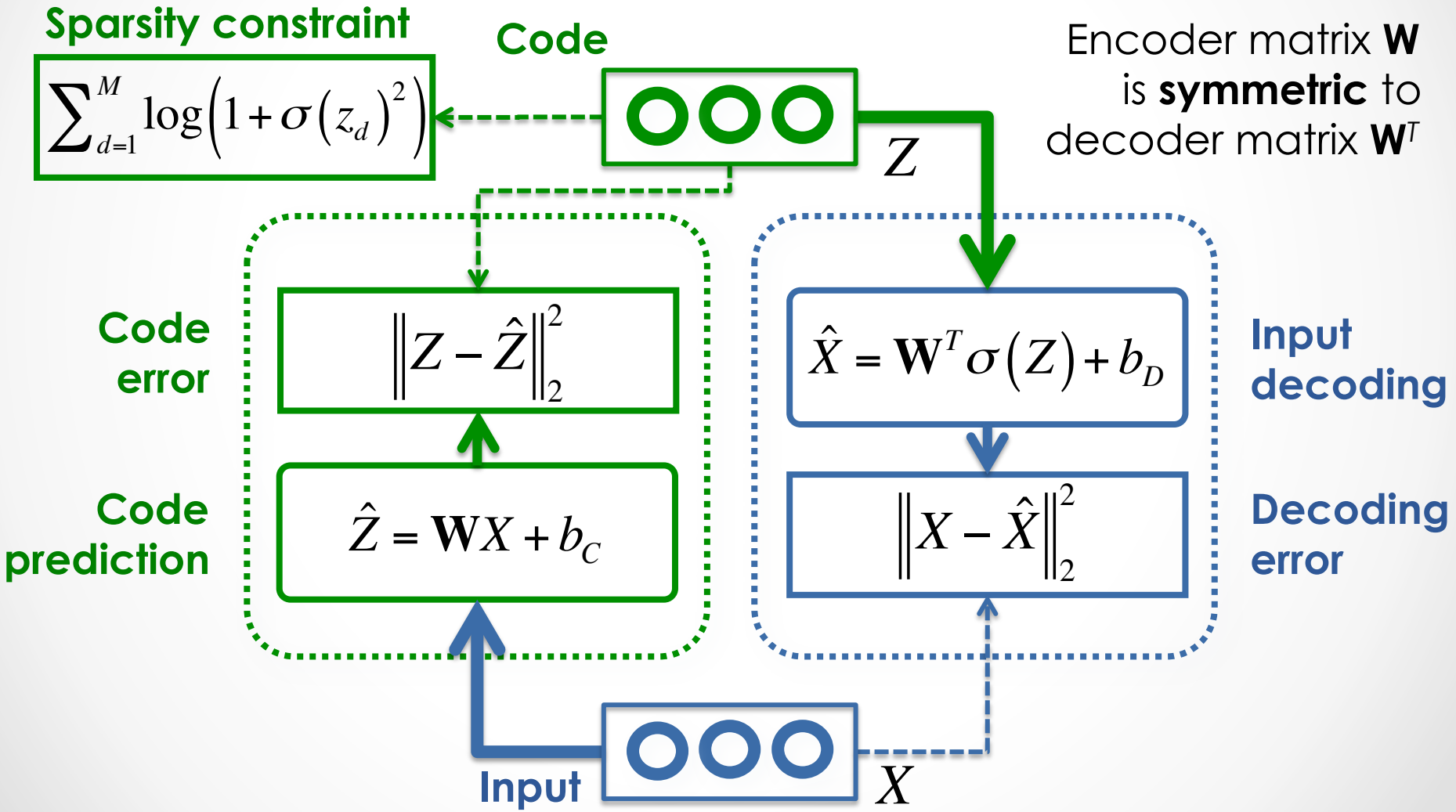
weights go to
infinity to compensate

# Sparse auto-encoder



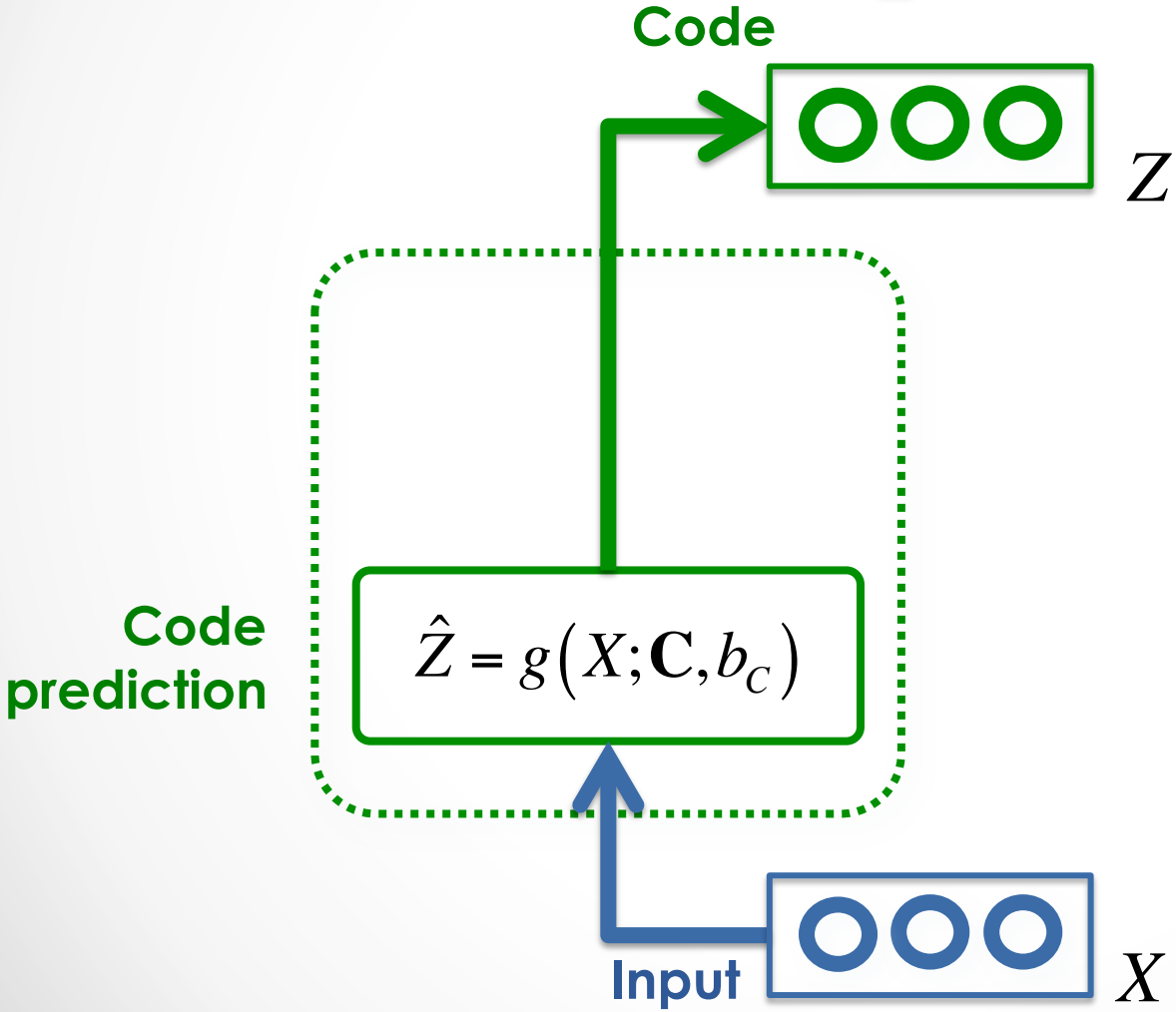

[Ranzato, Poultney, Chopra & LeCun, "Efficient Learning of Sparse Representations with an Energy-Based Model ", *NIPS*, 2006]
[Ranzato, Boureau & LeCun, "Sparse Feature Learning for Deep Belief Networks ", *NIPS*, 2007]

# Symmetric sparse auto-encoder




[Ranzato, Poultney, Chopra & LeCun, "Efficient Learning of Sparse Representations with an Energy-Based Model ", *NIPS*, 2006, Ranzato, Boureau & LeCun, "Sparse Feature Learning for Deep Belief Networks ", *NIPS*, 2007]

# Predictive Sparse Decomposition

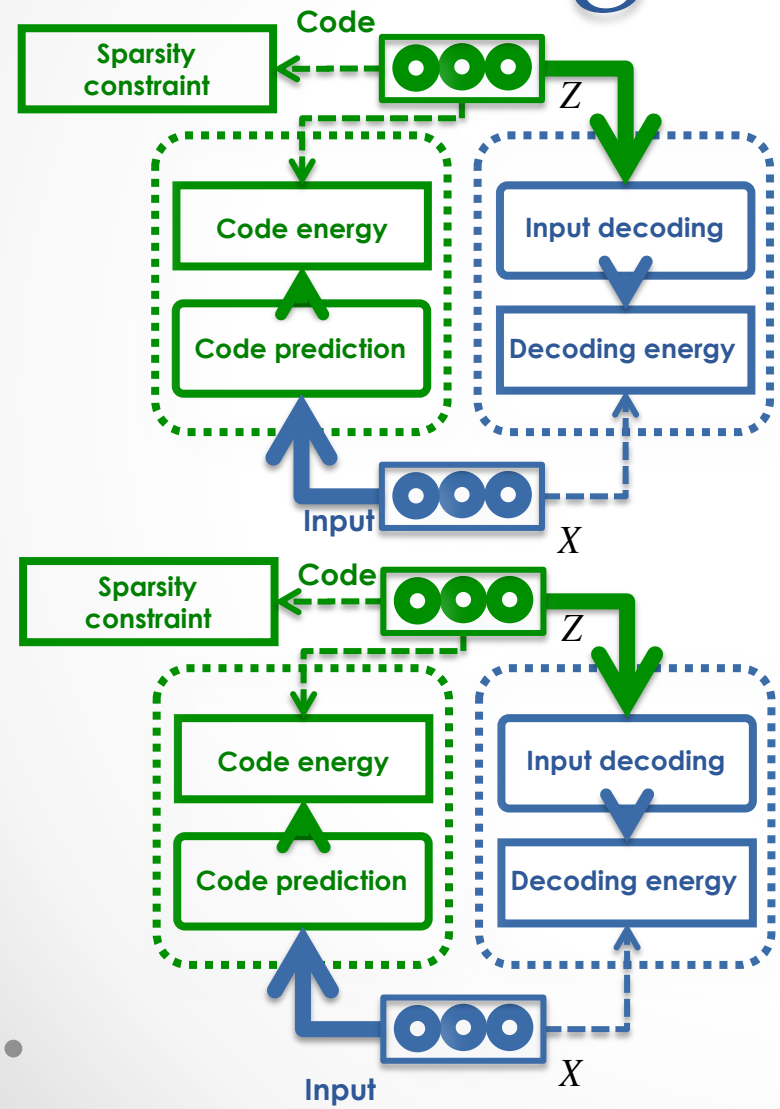



Once the encoder g is properly trained, the code Z can be **directly predicted** from input X

# Outline

- Deep learning concepts covered
  - **Hierarchical** representations
  - **Sparse** and/or **distributed** representations
  - Supervised vs. **unsupervised** learning
- Auto-encoder
  - Architecture
  - **Inference** and **learning**
  - Sparse coding
  - Sparse auto-encoders
- Illustration: handwritten digits
  - **Stacking** auto-encoders
  - Learning representations of digits
  - Impact on **classification**
- Applications to text
  - Semantic hashing
  - Semi-supervised learning
  - Moving away from auto-encoders
- Topics not covered in this talk

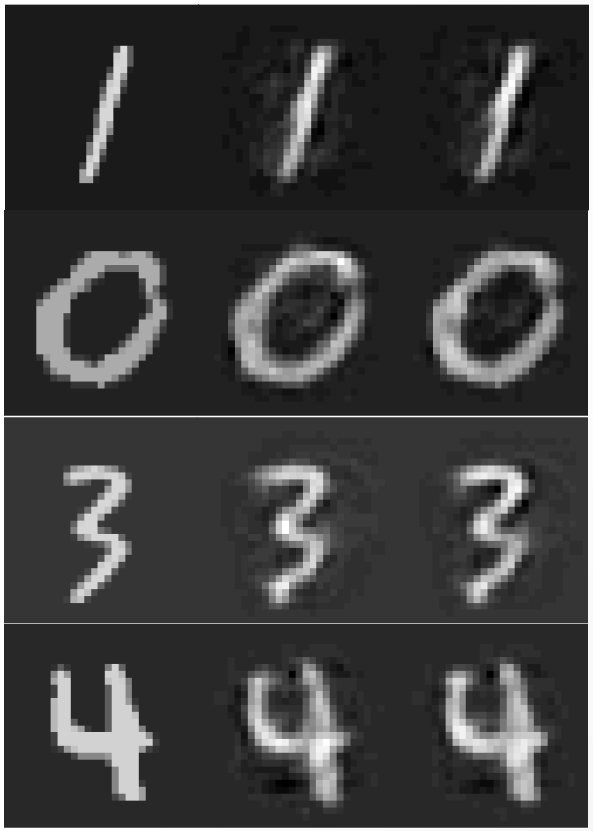
# Stacking auto-encoders




[Ranzato, Boureau & LeCun, "Sparse Feature Learning for Deep Belief Networks", *NIPS*, 2007]

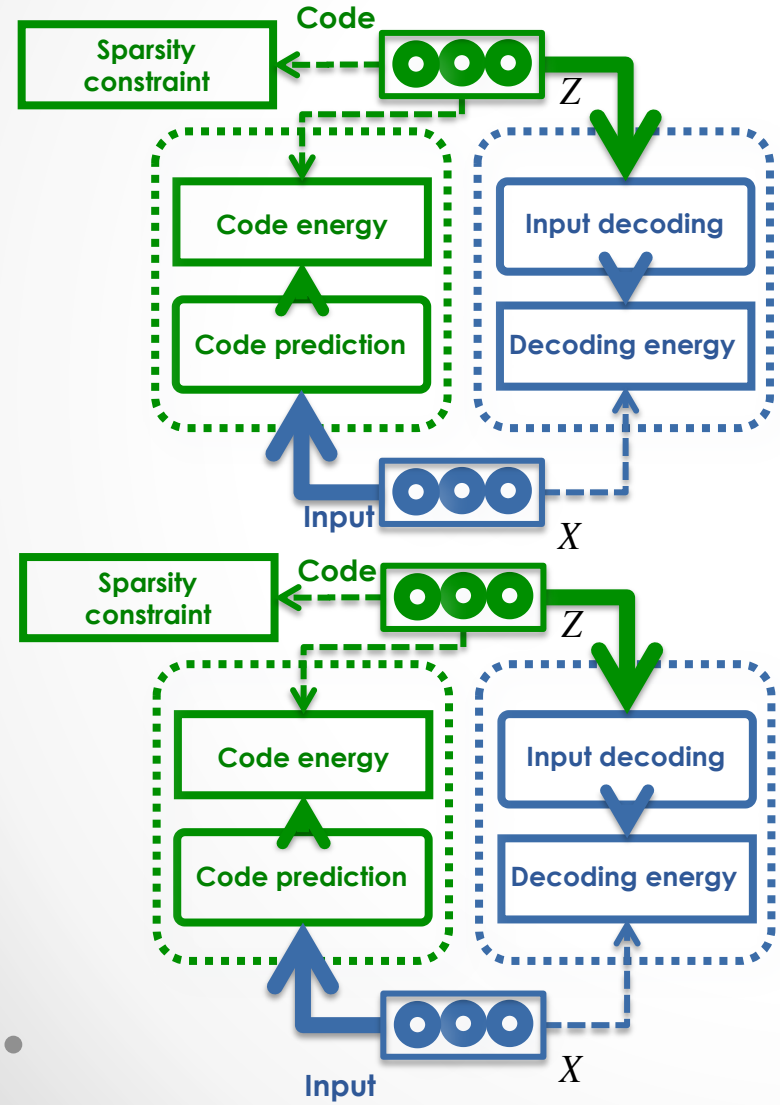
# MNIST handwritten digits

- Database of 70k handwritten digits
  - Training set: 60k
  - Test set: 10k
- 28 x 28 pixels
- Best performing classifiers:
  - Linear classifier: 12% error
  - Gaussian SVM 1.4% error
  - ConvNets <1% error



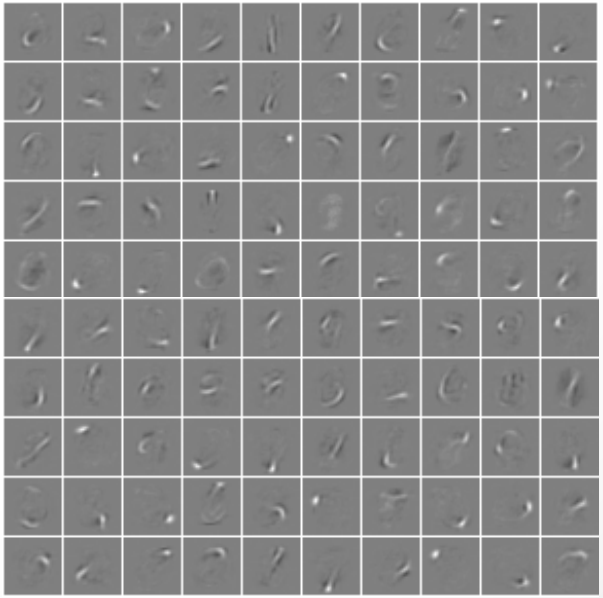
[http://yann.lecun.com/exdb/mnist/]

# Stacked auto-encoders



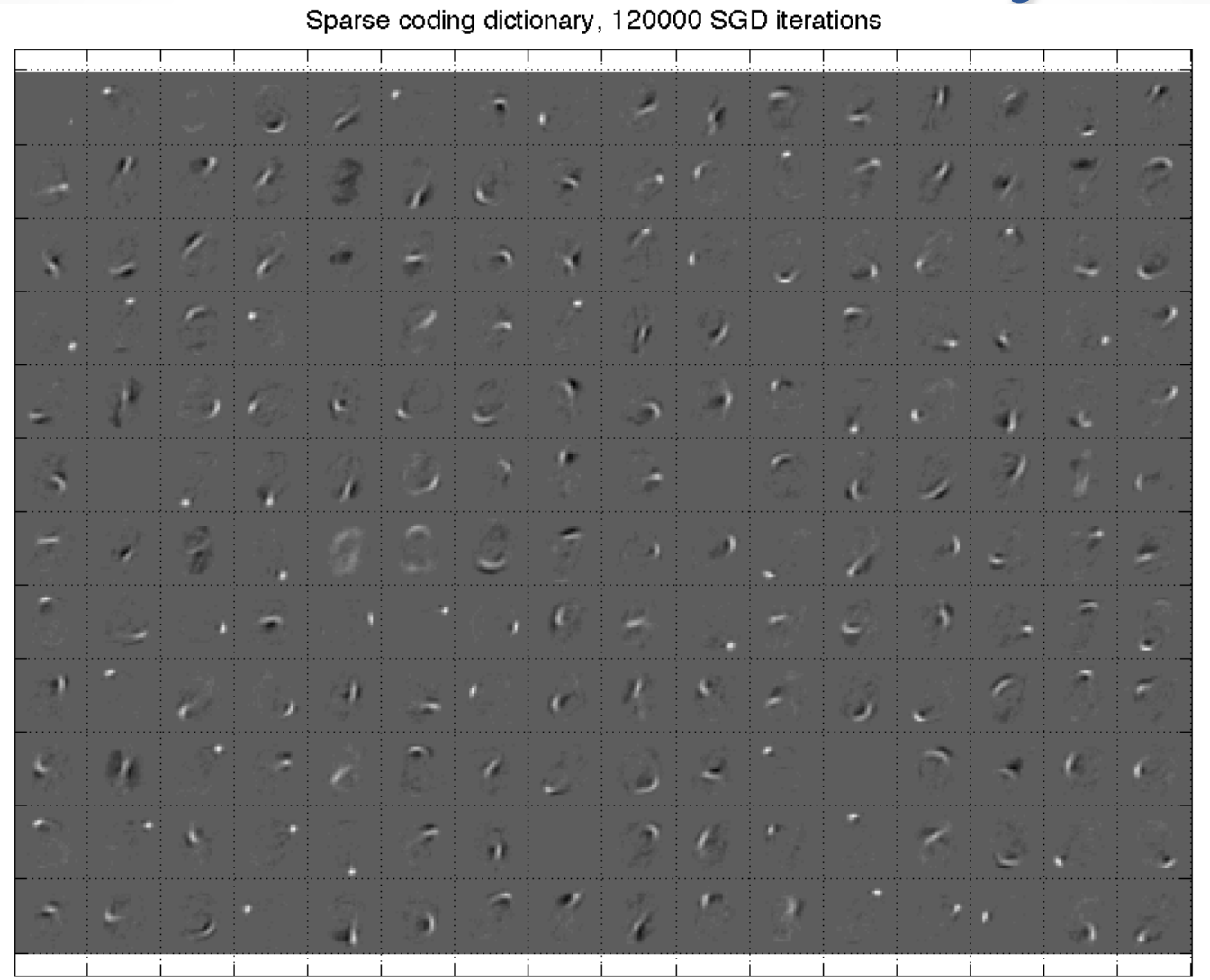

Layer 2: Matrix $\mathbf{W_2}$ of size 10 x 192
**10** sparse bases of 192 units

Layer 1: Matrix $\mathbf{W_1}$ of size 192 x 784
**192** sparse bases of 28 x 28 pixels

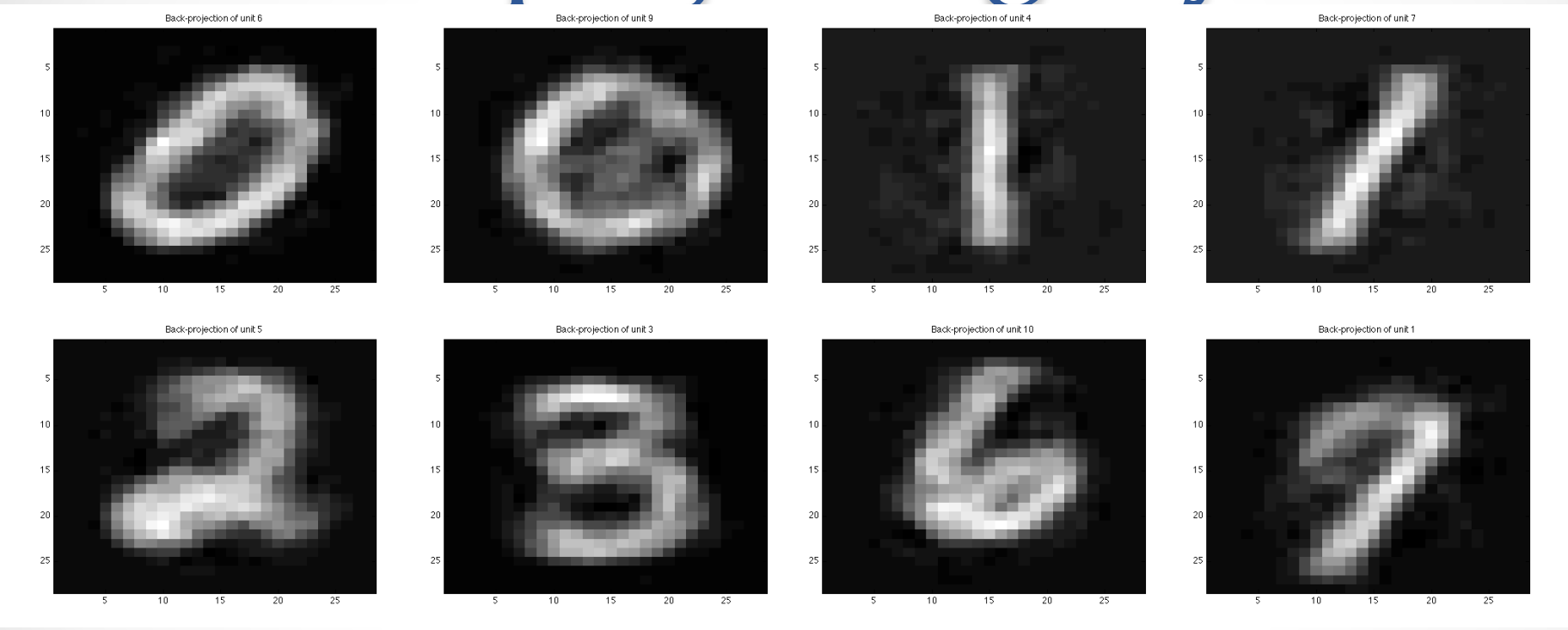


[Ranzato, Boureau & LeCun, "Sparse Feature Learning for Deep Belief Networks", *NIPS*, 2007]

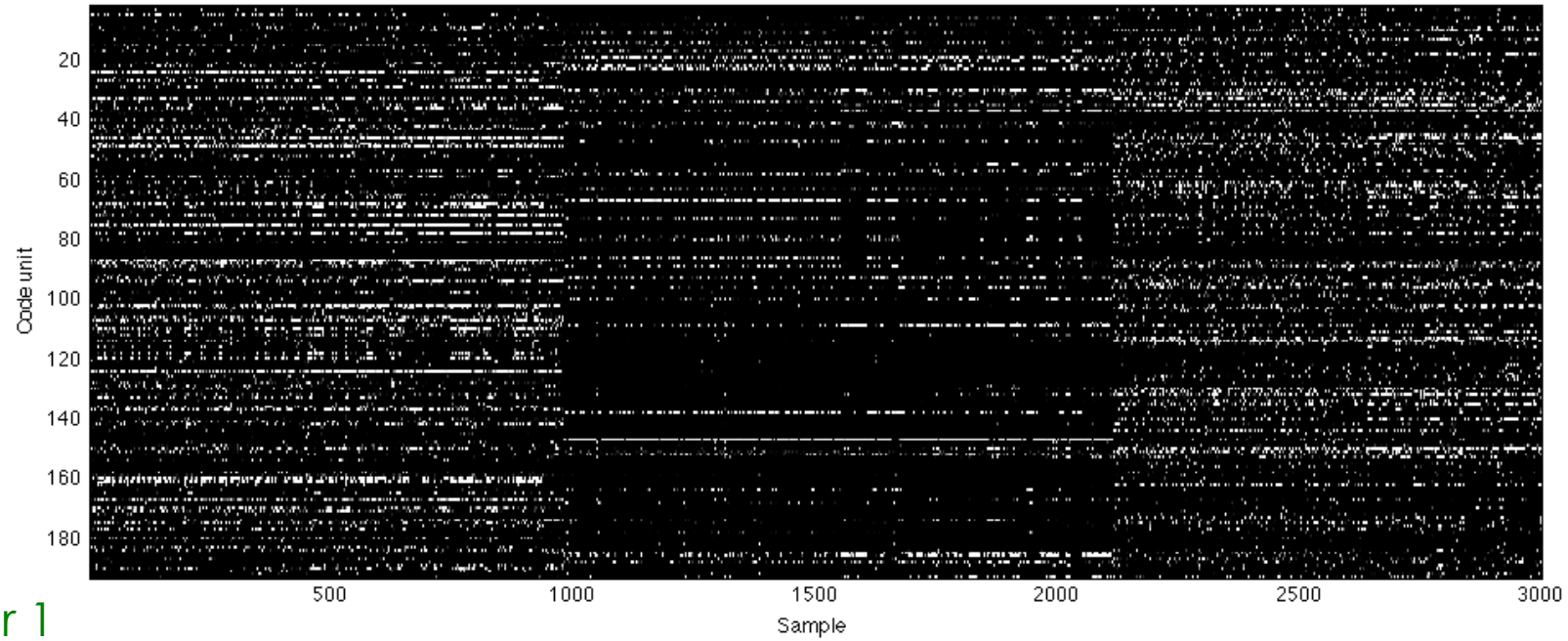
# Our results: bases learned on layer 1



# Our results: back-projecting layer 2



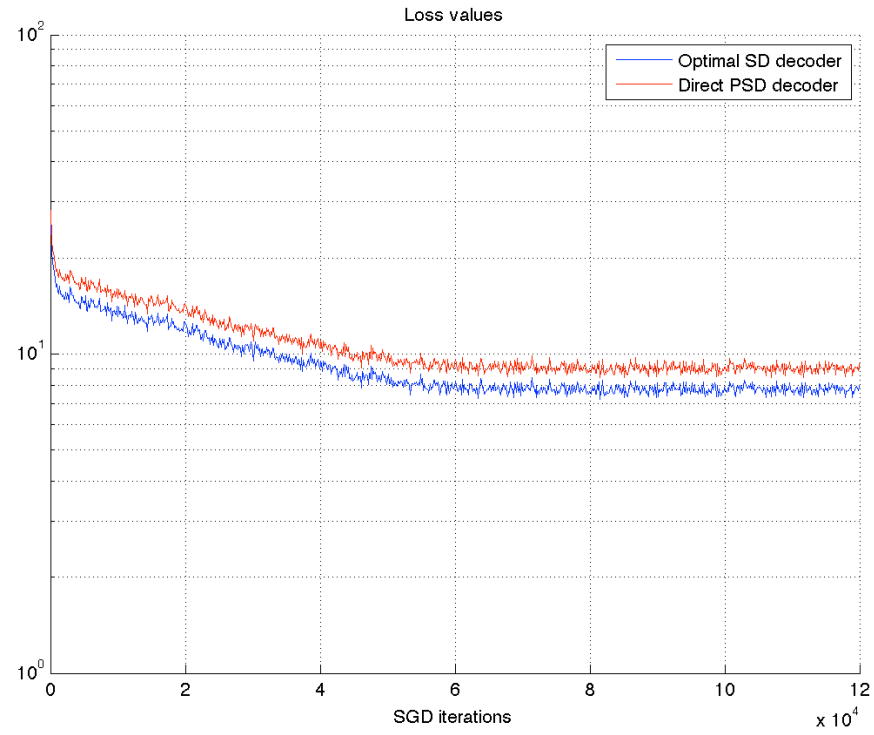
# Sparse representations



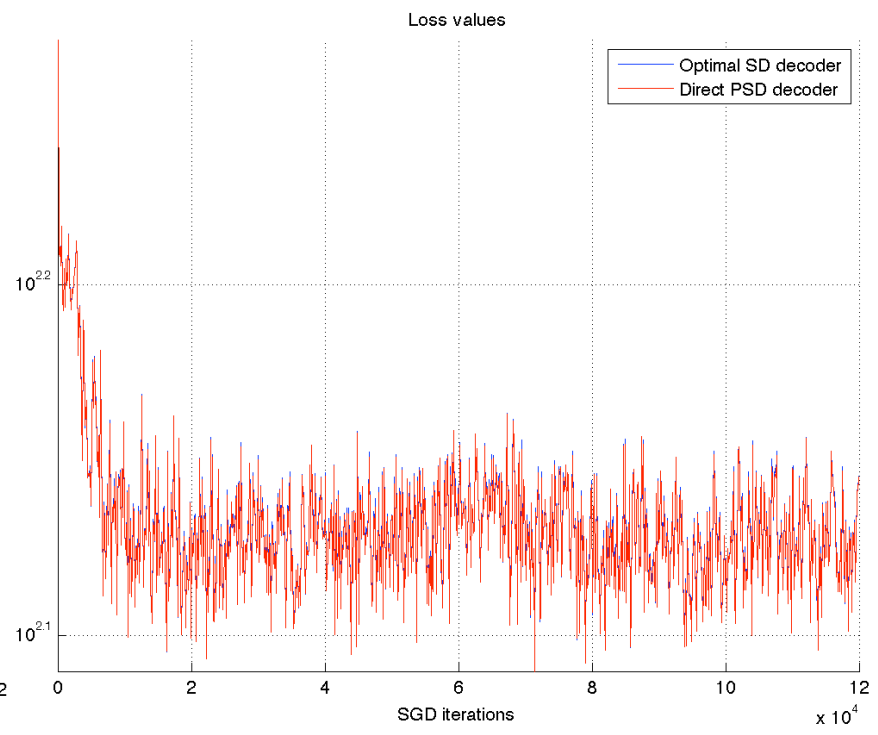

Layer 1


Layer 2

# Training "converges" in one pass over data




Layer 1

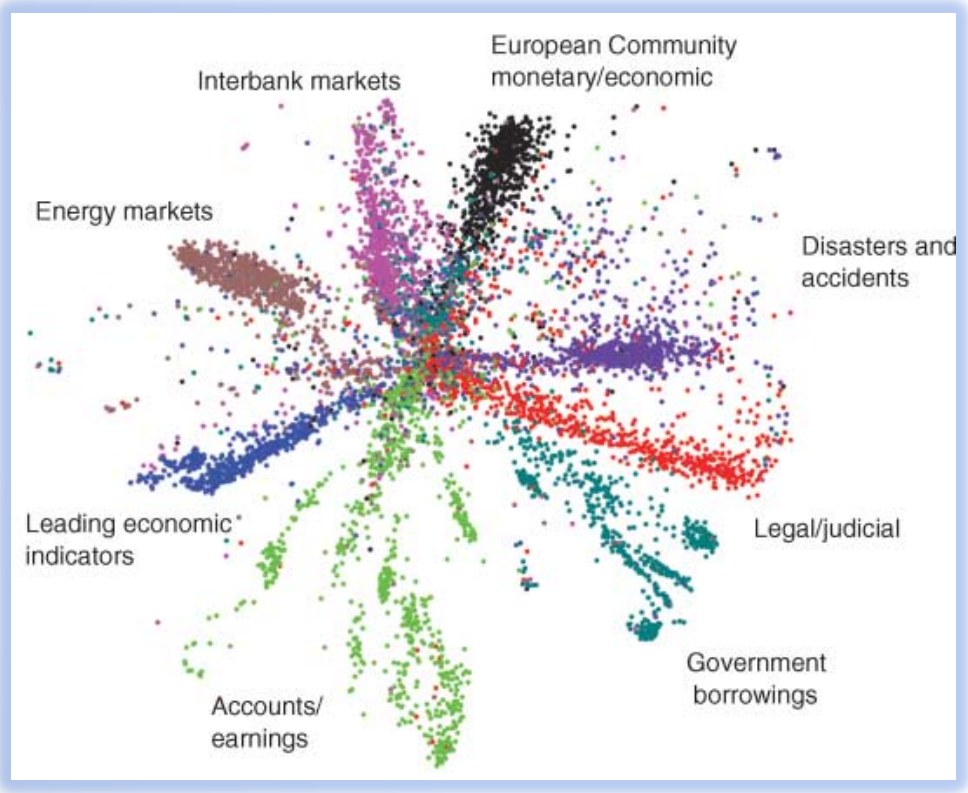


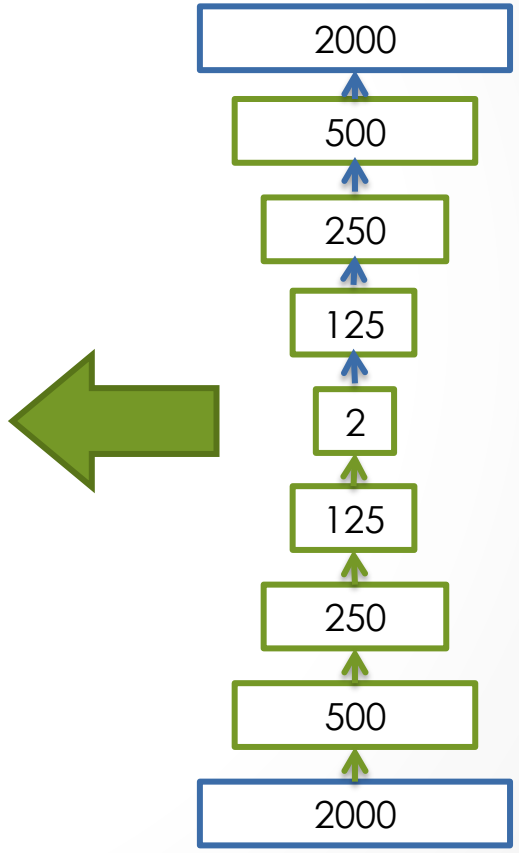

Layer 2

# Outline

- Deep learning concepts covered
  - **Hierarchical** representations
  - **Sparse** and/or **distributed** representations
  - Supervised vs. **unsupervised** learning
- Auto-encoder
  - Architecture
  - **Inference** and **learning**
  - Sparse coding
  - Sparse auto-encoders
- Illustration: handwritten digits
  - **Stacking** auto-encoders
  - Learning representations of digits
  - Impact on **classification**
- Applications to text
  - Semantic hashing
  - Semi-supervised learning
  - Moving away from auto-encoders
- Topics not covered in this talk

# Semantic Hashing

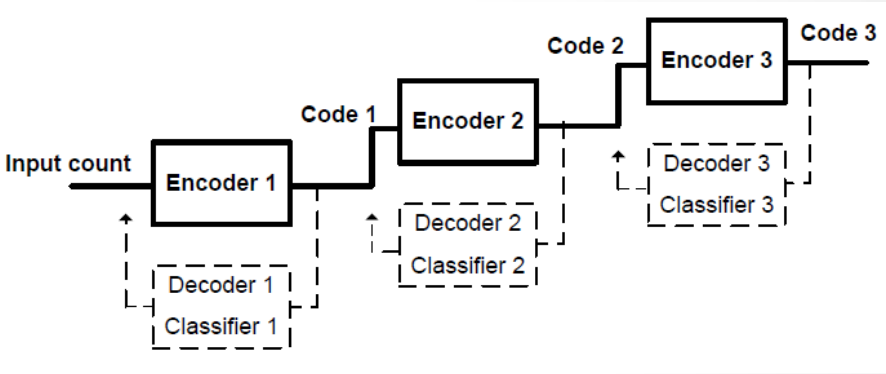


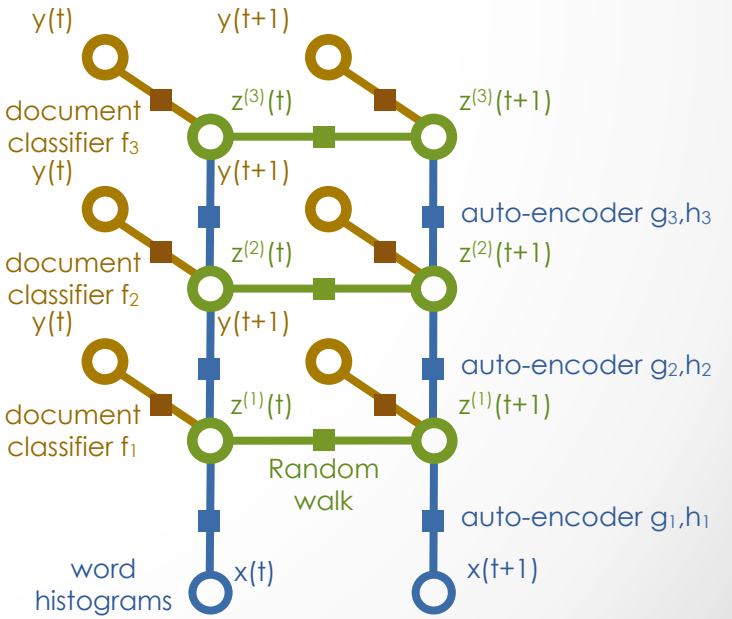




[Hinton & Salakhutdinov, "Reducing the dimensionality of data with neural networks, *Science*, 2006; Salakhutdinov & Hinton, "Semantic Hashing", *Int J Approx Reason*, 2007]

# Semi-supervised learning of auto-encoders

- Add classifier module to the codes
- When a **input** $X(t)$ has a **label** $Y(t)$, back-propagate the prediction error on $Y(t)$ to the **code** $Z(t)$
- Stack the encoders
- Train layer-wise



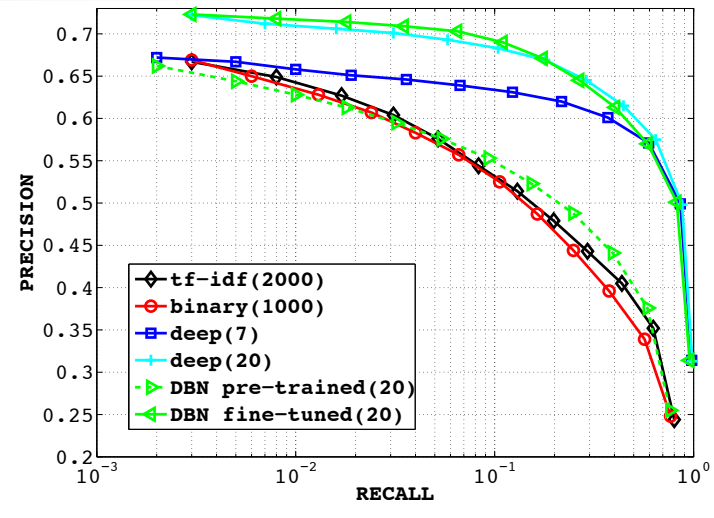




[Ranzato & Szummer, "Semi-supervised learning of compact document representations with deep networks", *ICML*, 2008; Mirowski, Ranzato & LeCun, "Dynamic auto-encoders for semantic indexing", *NIPS Deep Learning Workshop*, 2010]

# Semi-supervised learning of auto-encoders

2000w-TFIDF logistic regression $F_1$=0.83
2000w-TFIDF SVM $F_1$=0.84

| K | LSA+ICA | [Mirowski et al, 2010] no dynamics | [Mirowski et al, 2010] $L_1$ dynamics | [Ranzato et al, 2008] |
|---|---|---|---|---|
| 100 | 0.81 | **0.86** | **0.85** | **0.85** |
| 30 | 0.70 | **0.86** | **0.85** | **0.85** |
| 10 | 0.40 | **0.86** | **0.85** | 0.84 |
| 2 | 0.09 | **0.54** | **0.51** | 0.19 |



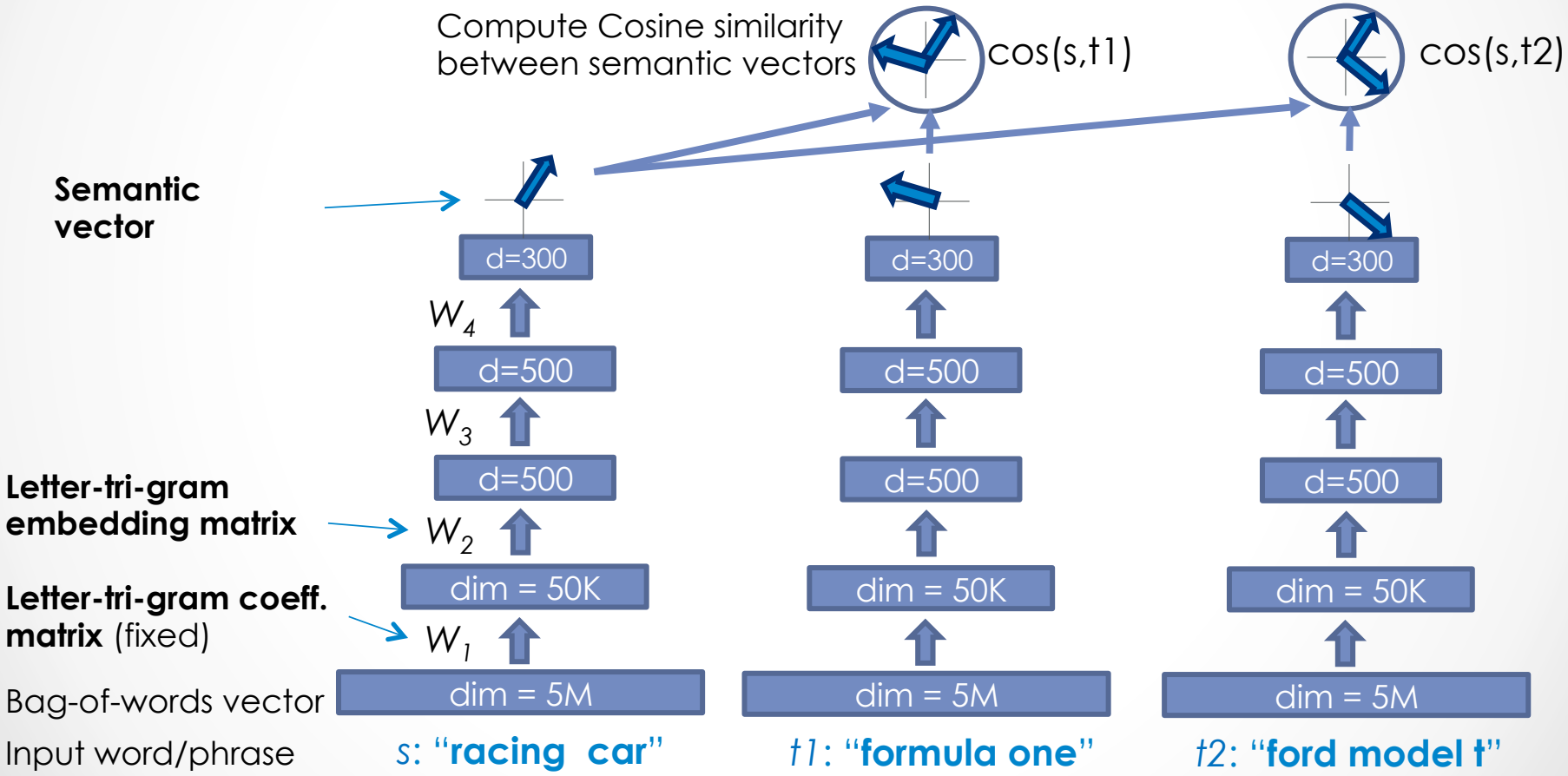

Performance on document retrieval task:
**Reuters-21k** dataset (9.6k training, 4k test), vocabulary 2k words, 10-class classification

Comparison with:

- unsupervised techniques (DBN: Semantic Hashing, LSA) + SVM
- traditional technique: word TF-IDF + SVM

[Ranzato & Szummer, "Semi-supervised learning of compact document representations with deep networks", *ICML*, 2008; Mirowski, Ranzato & LeCun, "Dynamic auto-encoders for semantic indexing", *NIPS Deep Learning Workshop*, 2010]

# Beyond auto-encoders for web search (MSR)




[Huang, He, Gao, Deng et al, "Learning Deep Structured Semantic Models for Web Search using Clickthrough Data", *CIKM*, 2013]

# Beyond auto-encoders for web search (MSR)

Results on a web ranking task (16k queries)
Normalized discounted cumulative gains

| # | Models | NDCG@1 | NDCG@3 | NDCG@10 |
|---|---|---|---|---|
| 1 | TF-IDF | 0.319 | 0.382 | 0.462 |
| 2 | BM25 | 0.308 | 0.373 | 0.455 |
| 3 | WTM | 0.332 | 0.400 | 0.478 |
| 4 | LSA | 0.298 | 0.372 | 0.455 |
| 5 | PLSA | 0.295 | 0.371 | 0.456 |
| 6 | DAE | 0.310 | 0.377 | 0.459 |
| 7 | BLTM-PR | 0.337 | 0.403 | 0.480 |
| 8 | DPM | 0.329 | 0.401 | 0.479 |
| 9 | DNN | 0.342 | 0.410 | 0.486 |
| 10 | L-WH linear | 0.357 | 0.422 | 0.495 |
| 11 | L-WH non-linear | 0.357 | 0.421 | 0.494 |
| 12 | L-WH DNN | 0.362 | 0.425 | 0.498 |

**Semantic hashing**
[Salakhutdinov & Hinton, 2007]

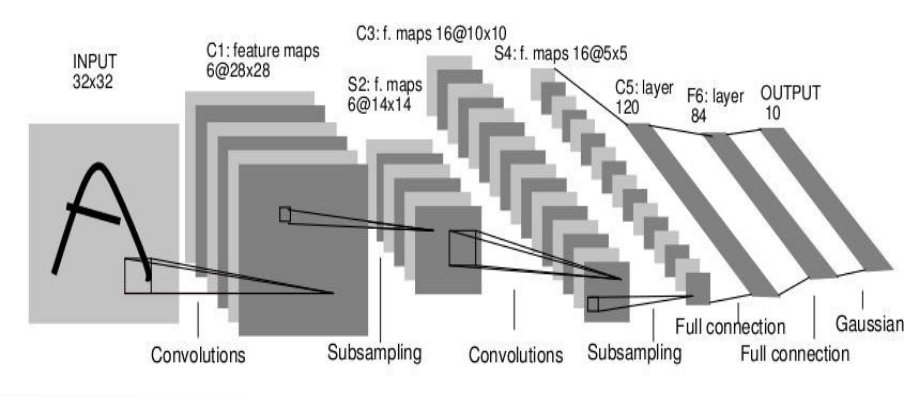
**Deep Structured Semantic Model**
[Huang, He, Gao et al, 2013]

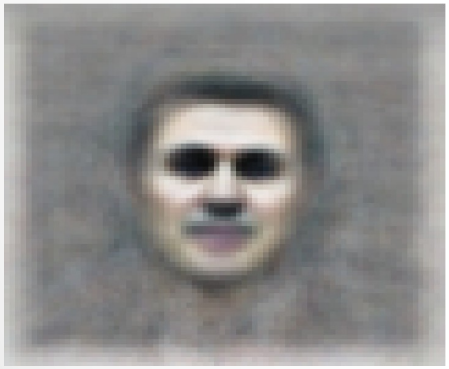
[Huang, He, Gao, Deng et al, "Learning Deep Structured Semantic Models for Web Search using Clickthrough Data", *CIKM*, 2013]

# Outline

- Deep learning concepts covered
  - **Hierarchical** representations
  - **Sparse** and/or **distributed** representations
  - Supervised vs. **unsupervised** learning
- Auto-encoder
  - Architecture
  - **Inference** and **learning**
  - Sparse coding
  - Sparse auto-encoders
- Illustration: handwritten digits
  - **Stacking** auto-encoders
  - Learning representations of digits
  - Impact on **classification**
- Applications to text
  - Semantic hashing
  - Semi-supervised learning
  - Moving away from auto-encoders
- Topics not covered in this talk

# Topics not covered in this talk



- Other variations of auto-encoders
  - **Restricted Boltzmann Machines** (work in Geoff Hinton's lab)
  - **Denoising Auto-Encoders** (work in Yoshua Bengio's lab)
- Invariance to shifts in input and feature space
  - **Convolutional** kernels
  - Sliding **windows** over input
  - **Max-pooling** over codes

[LeCun, Bottou, Bengio & Haffner, "Gradient-based learning applied to document recognition", Proceedings of IEEE,1998;
Le, Ranzato et al. "Building high-level features using large-scale unsupervised learning" *ICML* 2012;
Sermanet et al, "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks", *ICLR* 2014]

# Thank you!

- Tutorial code:
  https://github.com/piotrmirowski
  http://piotrmirowski.wordpress.com
- Contact:
  piotr.mirowski@computer.org
- Acknowledgements:
  Marc'Aurelio Ranzato (FB)
  Yann LeCun (FB/NYU)

# Auto-encoders and Expectation-Maximization

Energy of inputs and codes

$$E(\mathbf{X},\mathbf{Z};\mathbf{W}) = \sum_{t=1}^{T} \alpha \left\| Z_t - g(X_t;\mathbf{C},b_C) \right\|_2^2 + \sum_{t=1}^{T} \left\| X_t - h(Z_t;\mathbf{D},b_D) \right\|_2^2$$

Input data likelihood

$$P(\mathbf{X}|\mathbf{W}) = \int_{\mathbf{Z}} P(\mathbf{X},\mathbf{Z}|\mathbf{W}) = \frac{\int_{\mathbf{Z}} e^{-\beta E(\mathbf{X},\mathbf{Z};\mathbf{W})}}{\int_{\mathbf{X},\mathbf{Z}} e^{-\beta E(\mathbf{X},\mathbf{Z};\mathbf{W})}}$$

Do not marginalize over: take maximum likelihood latent code instead

$$-\log P(\mathbf{X}|\mathbf{W}) = -\frac{1}{\beta}\log \int_{Z} e^{-\beta E(\mathbf{X},\mathbf{Z};\mathbf{W})} + \frac{1}{\beta}\log \int_{Y,Z} e^{-\beta E(\mathbf{X},\mathbf{Z};\mathbf{W})}$$

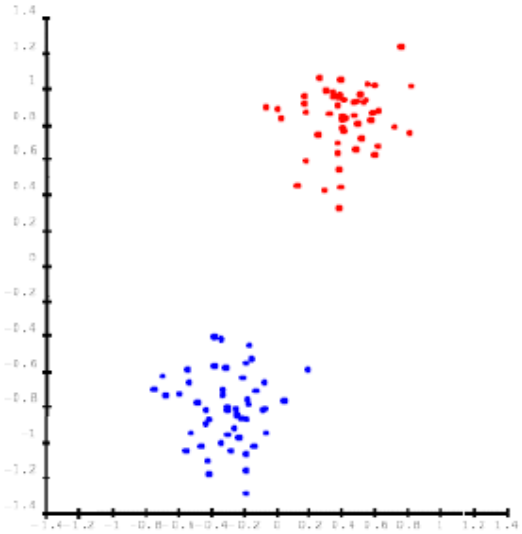
**Maximum A Posteriori**: take minimal energy code **Z**

$$-\log P(\mathbf{X}|\mathbf{W}) = E(\mathbf{X},\mathbf{Z}^*;\mathbf{W}) + \frac{1}{\beta}\log \int_{Y} e^{-\beta E(\mathbf{X},\mathbf{Z}^*;\mathbf{W})}$$

Enforce sparsity on **Z** to constrain **Z** and avoid computing partition function

# Stochastic gradient descent

**Dataset #1**



Examples of each class are drawn from a Gaussian distribution centered at (–0.4, –0.8), and (0.4, 0.8).

Eigenvalues of covariance matrix: 0.83 and 0.036

**Batch gradient descent**

data set: set–1 (100 examples, 2 gaussians)
network: 1 linear unit, 2 inputs, 1 output.
2 weights, 1 bias.

Learning rate:

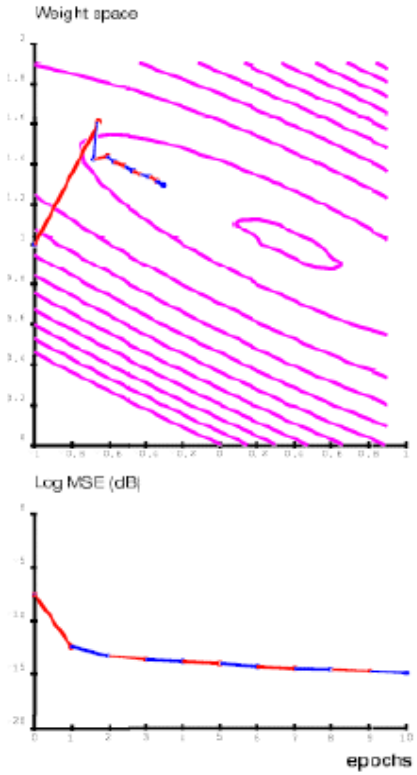
$\eta = 1.5$

Hessian largest eigenvalue:

$\lambda_{max} = 0.84$

Maximum admissible Learning rate:

$\eta_{max} = 2.38$




[LeCun et al, "Efficient BackProp", *Neural Networks: Tricks of the Trade*, 1998;
Bottou, "Stochastic Learning", *Slides from a talk in Tubingen*, 2003]

# Stochastic gradient descent

**Batch gradient descent**

data set: set–1 (100 examples, 2 gaussians)
network: 1 linear unit, 2 inputs, 1 output.
2 weights, 1 bias.

Learning rate:

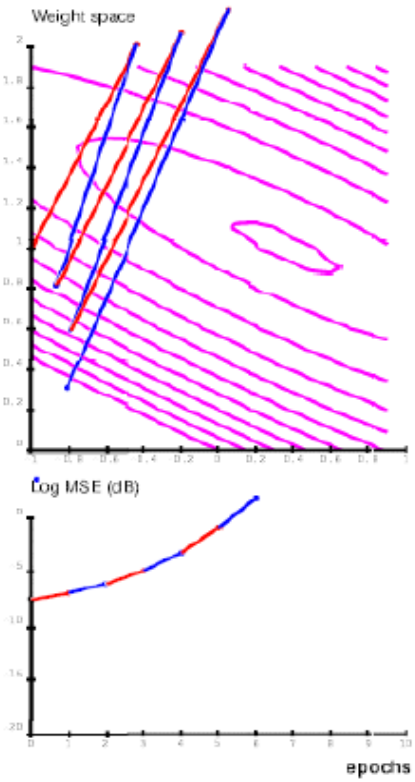
$\eta = 2.5$

Hessian largest eigenvalue:

$\lambda_{max} = 0.84$

Maximum admissible Learning rate:

$\eta_{max} = 2.38$




**Stochastic gradient descent**

data set: set–1 (100 examples, 2 gaussians)
network: 1 linear unit, 2 inputs, 1 output.
2 weights, 1 bias.

Learning rate:

$\eta = 0.2$

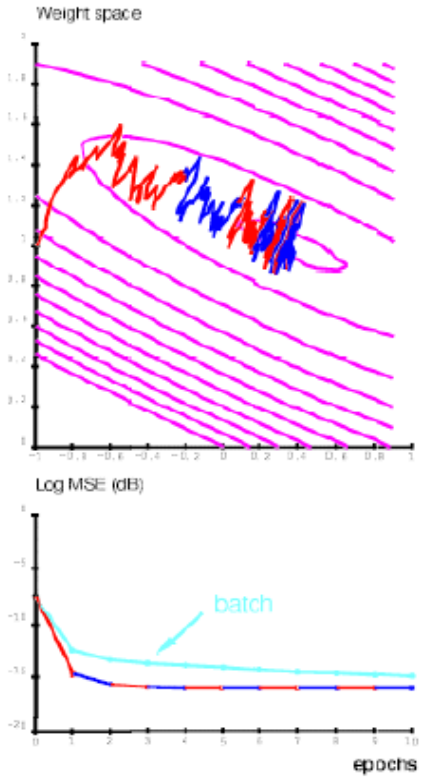
(equivalent to a batch learning rate of 20)

Hessian largest eigenvalue:

$\lambda_{max} = 0.84$

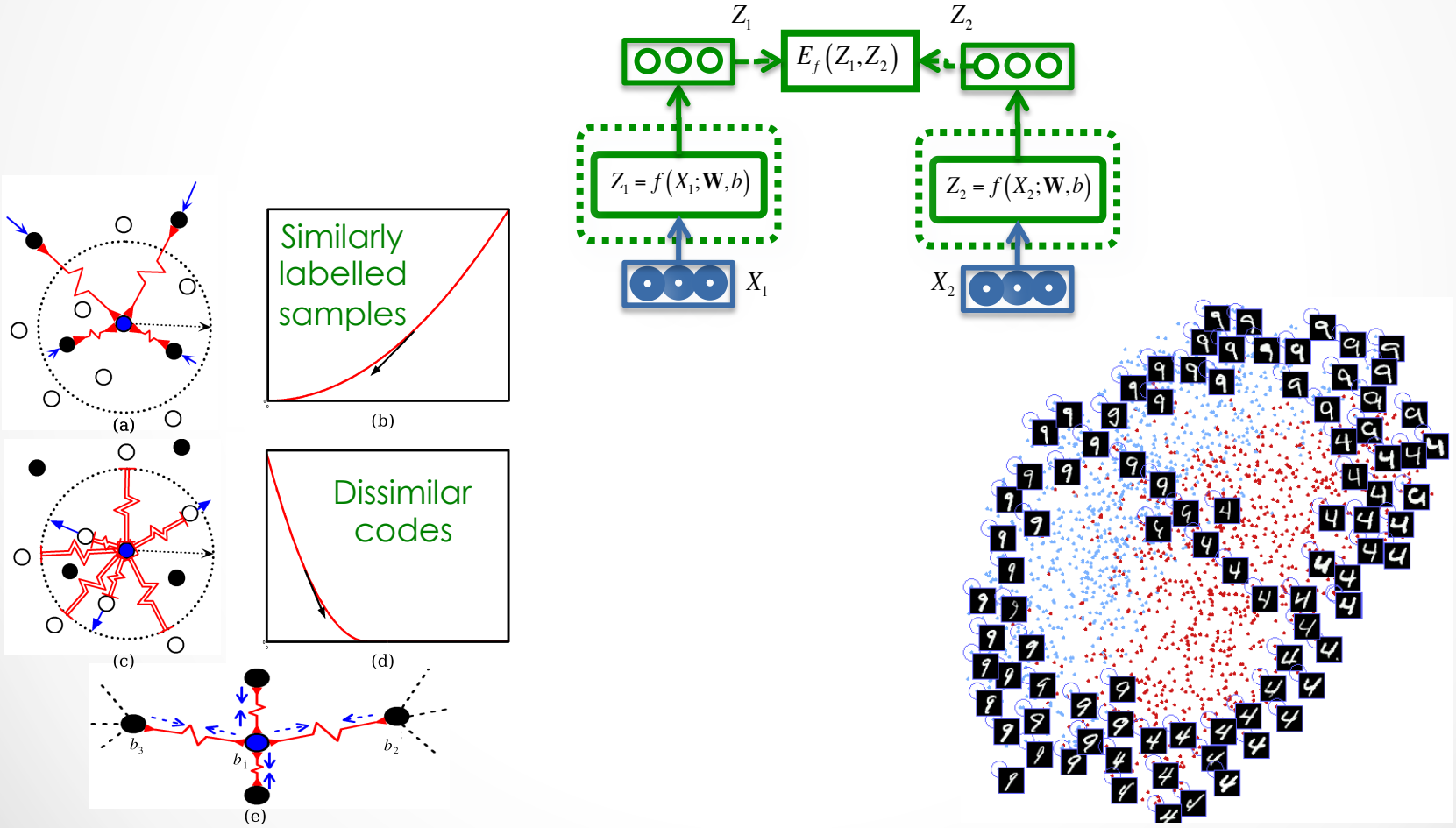
Maximum admissible Learning rate (for batch):

$\eta_{max} = 2.38$




[LeCun et al, "Efficient BackProp", *Neural Networks: Tricks of the Trade*, 1998;
Bottou, "Stochastic Learning", *Slides from a talk in Tubingen*, 2003]

# Dimensionality reduction and invariant mapping




[Hadsell, Chopra & LeCun, "Dimensionality Reduction by Learning an Invariant Mapping", *CVPR*, 2006]