# Unsupervised Pretraining, Autoencoder and Manifolds

Christian Herta

# Outline

- Autoencoders
- Unsupervised pretraining of deep networks with autoencoders
- Manifold-Hypotheses

# Autoencoder

# Problems of training of deep neural networks

- stochastic gradient descent + standard algorithm "Backpropagation":

  vanishing or exploding gradient: "Vanishing Gradient Problem" [Hochreiter 1991]
- only shallow nets are trainable

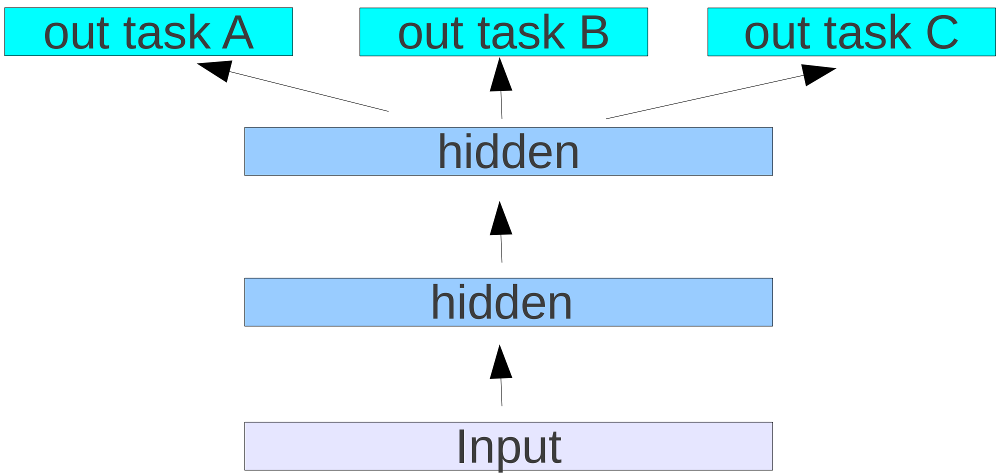
  => feature engineering
- for applications (in the past): most only one layer

# Solutions for training deep nets

- **layer wise pretraining** (first by [Hin06] with RBM)
  - with unlabeled data (unsupervised pretraining)
    - Restricted Boltzmann Machines (BM)
    - Stacked autoencoder
    - Contrastive estimation
- more effective optimization
  - second order methods, like "Hessian free Optimization"
- more carefully initialization + other neuron types (e.g. linear rectified/maxout) + dropout+ more sophisticated momentum (e.g. nesterov momentum); see e.g. [Glo11]

# Representation Learning

- "Feature Learning" statt "Feature Engineering"
- *Multi Task Learning*:
  - learned Features (distributed representations) can be used for different tasks
  - unsupervised pretraining + supervised finetuning



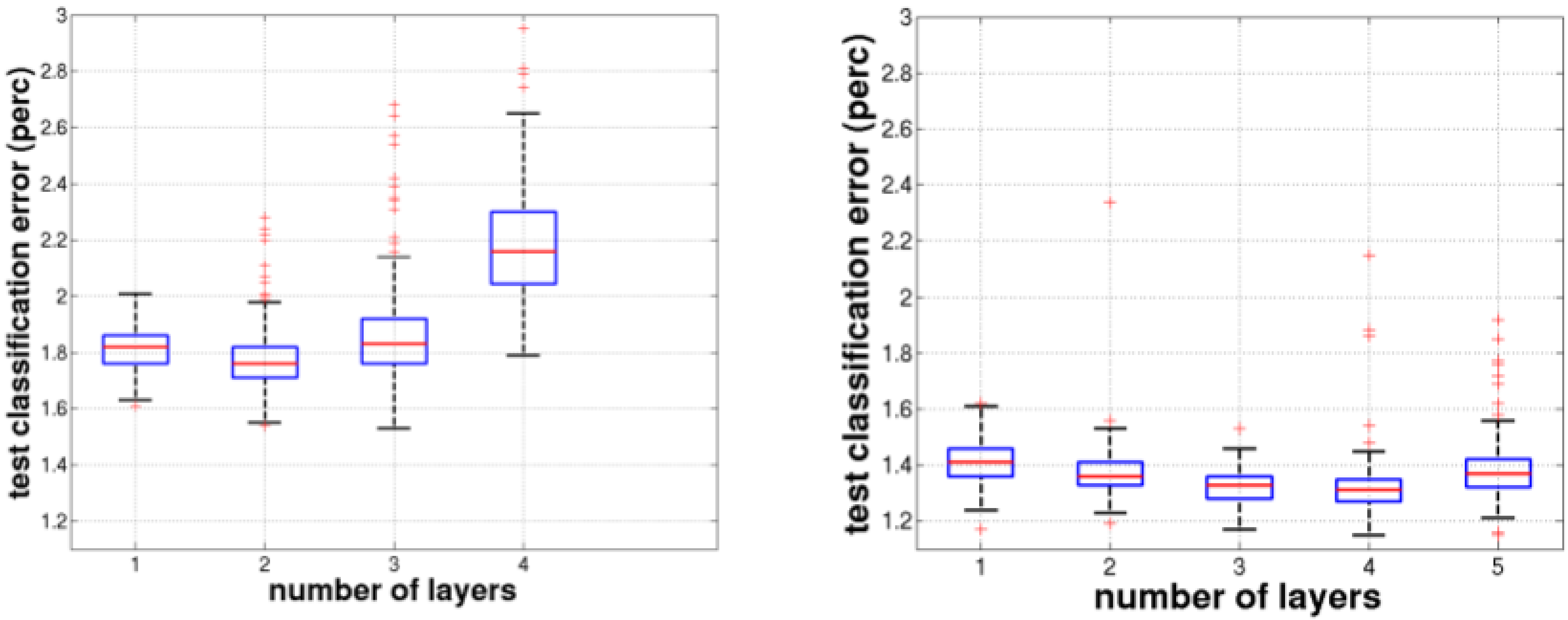



Figure 1: Effect of depth on performance for a model trained (**left**) without unsupervised pre-training and (**right**) with unsupervised pre-training, for 1 to 5 hidden layers (networks with 5 layers failed to converge to a solution, without the use of unsupervised pre-

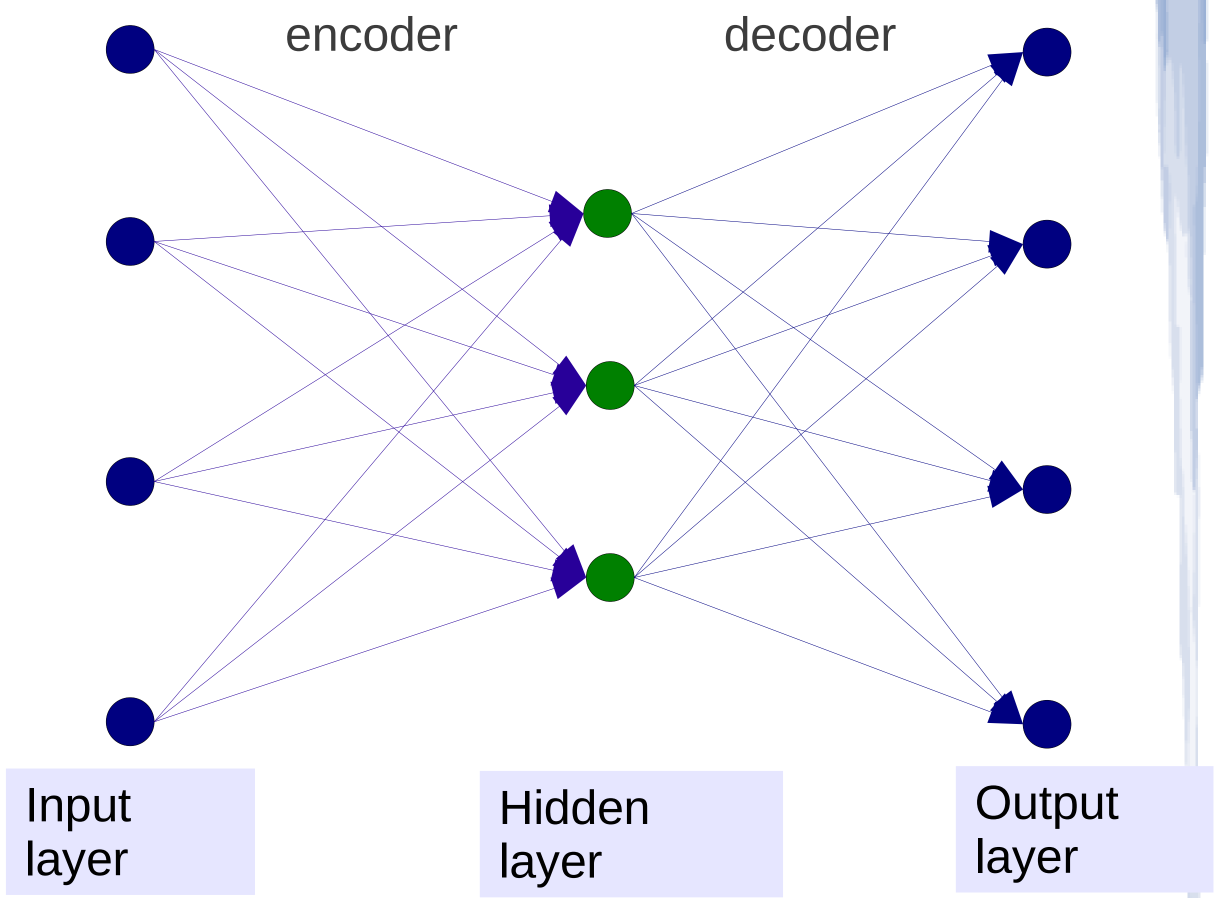
from
Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, Samy Bengio;
Why Does Unsupervised Pre-training Help Deep Learning?
JMLR2010

# Layer wise pretraining with autoencoders

# Autoencoder

- Goal:
  **reconstruction of the input**
  input = output

- different constraints on hidden layers
  - small number of neurons: compression of the input
  - other kinds of constraints, e.g. sparse autoencoder.



# Encoder-Decoder

- Encoder: $\vec{h}(\vec{x}) = s(W\vec{x} + \vec{b}_h)$
  - s: element wise sigmoid
  - Parameter: $W, \vec{b}_h$
- Decoder: $\vec{r} = \vec{g}(\vec{h}(\vec{x})) = s_2(W^T h(\vec{x}) + \vec{b}_r)$
  - Parameter: $W^T, \vec{b}_r$
  - Tied weights $W^T$ (shared with encoder)
  - activation function $s_2$:
    - logistic or identity

# Reconstruction Error

- Cost function: average reconstruction error

$$J_{AE}(\theta)=\sum_{\vec{x}\in D} L(\vec{x},\vec{r})$$

  - Reconstruction $\vec{r}=\vec{g}(\vec{h}(\vec{x}))$

- Loss function: reconstruction error
  - Squared error: $L(\vec{x},\vec{r})=\|\vec{x}-\vec{r}\|^2$
  - Bernoulli cross-entropy

$$L(\vec{x},\vec{r})=-\sum_{i=1}^{d} x_i \log(r_i)+(1-x_i)\log(1-r_i)$$

# Traditional Autoencoder

- Number of hidden units smaller than number of inputs/outputs
- Hidden state is a data driven compression of the input
- similar like (non-linear) PCA

# Sparse Autoencoder

- Sparsity Constraint
  - number of active hidden units should be small

$$J_{AE}(\theta)=\sum_{\vec{x}\in D}\left(L(\vec{x},\vec{r})+\lambda\sum_{j}|h_j(\vec{x})|\right)$$

(this sparsity constraint corresponds to a Lapacian prior from a probabilistic point of view)

  - other kinds of penalties are possible

# Contractive Autoencoder (CAE)
[Rif11]

- *Penalization* of the sensitivity on the input

$$J_{CAE}(\theta) = \sum_{\vec{x} \in D} \Big( \underbrace{L(\vec{x}, \vec{r})}_{\text{reconstruction}} + \underbrace{\lambda \, \| Jac(\vec{x}) \|^2}_{\text{contraction}} \Big)$$

  – with the Jaccobian of the encoder

$$Jac(\vec{x}) = \frac{\partial \vec{h}(\vec{x})}{\partial \vec{x}}$$

Intuition: hidden state not sensitive to input (but reconstruction should be performed)

  – and the hyperparameter $\lambda$

- also possible additionally for higher order derivatives (e.g. Hessian)(CAE+H)

# Denoising Auto-Encoder (DAE)

[Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. J. Machine Learning Res., 11]

- Corruption of the input $C(\tilde{x}|x)$
  - corrupted input $\tilde{x}$
  - original input $x$
- Reconstruction of the corrupted input with the autoencoder
  - DAE learns a reconstruction distribution $P(x|\tilde{x})$
  - by the minimization of $-\log P(x|\tilde{x})$

- also sampling from the estimated distribution possible: Bengio, Y., Yao, L., Alain, G., and Vincent, P. (2013a). Generalized denoising auto-encoders as generative models. In Advances in Neural Information Processing Systems 26 (NIPS'13)

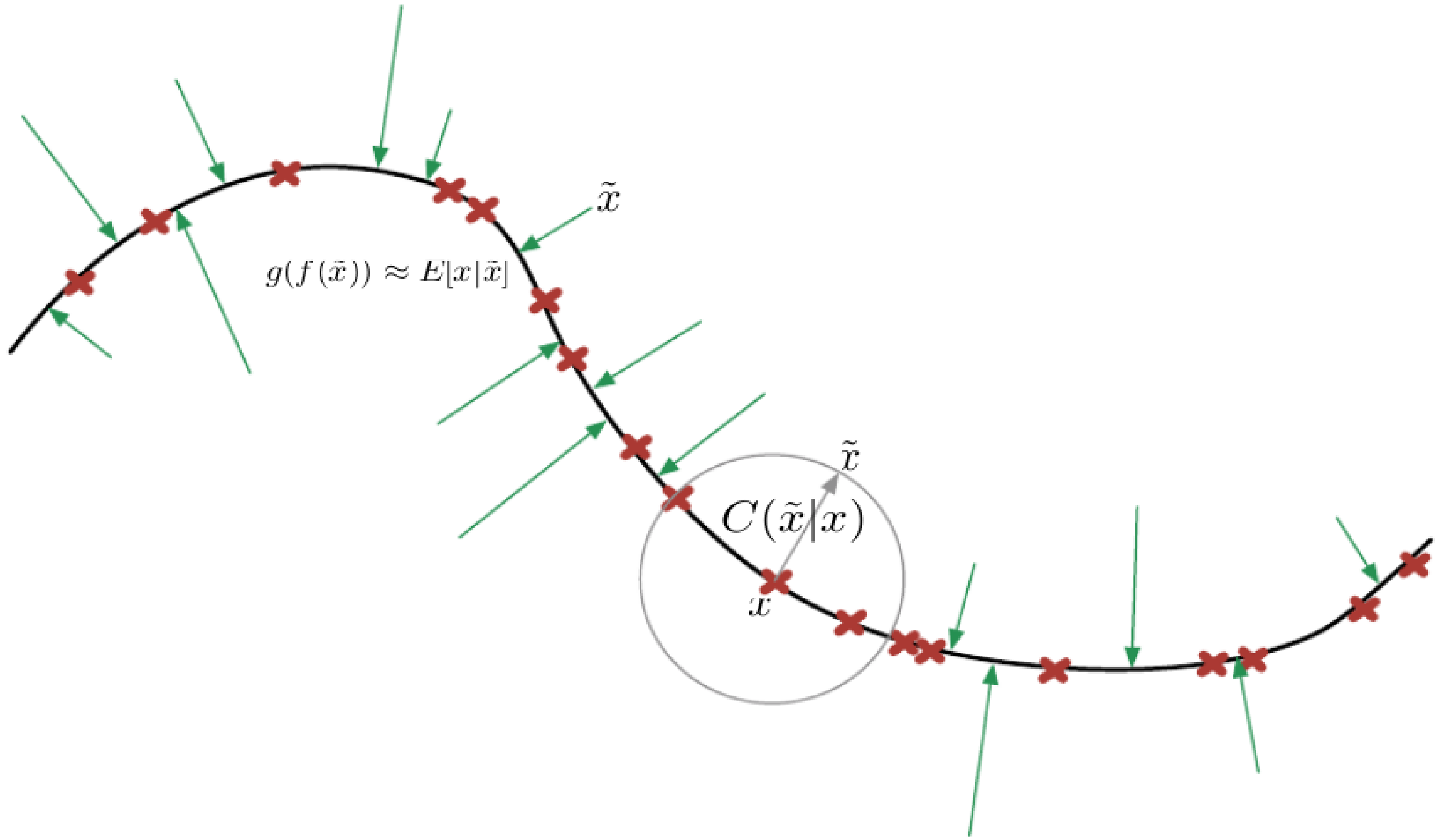



Figure 13.14: A denoising auto-encoder is trained to reconstruct the clean data point $\boldsymbol{x}$

from Bengio et. al. "Deep Learning", Book for MIT press in preparation

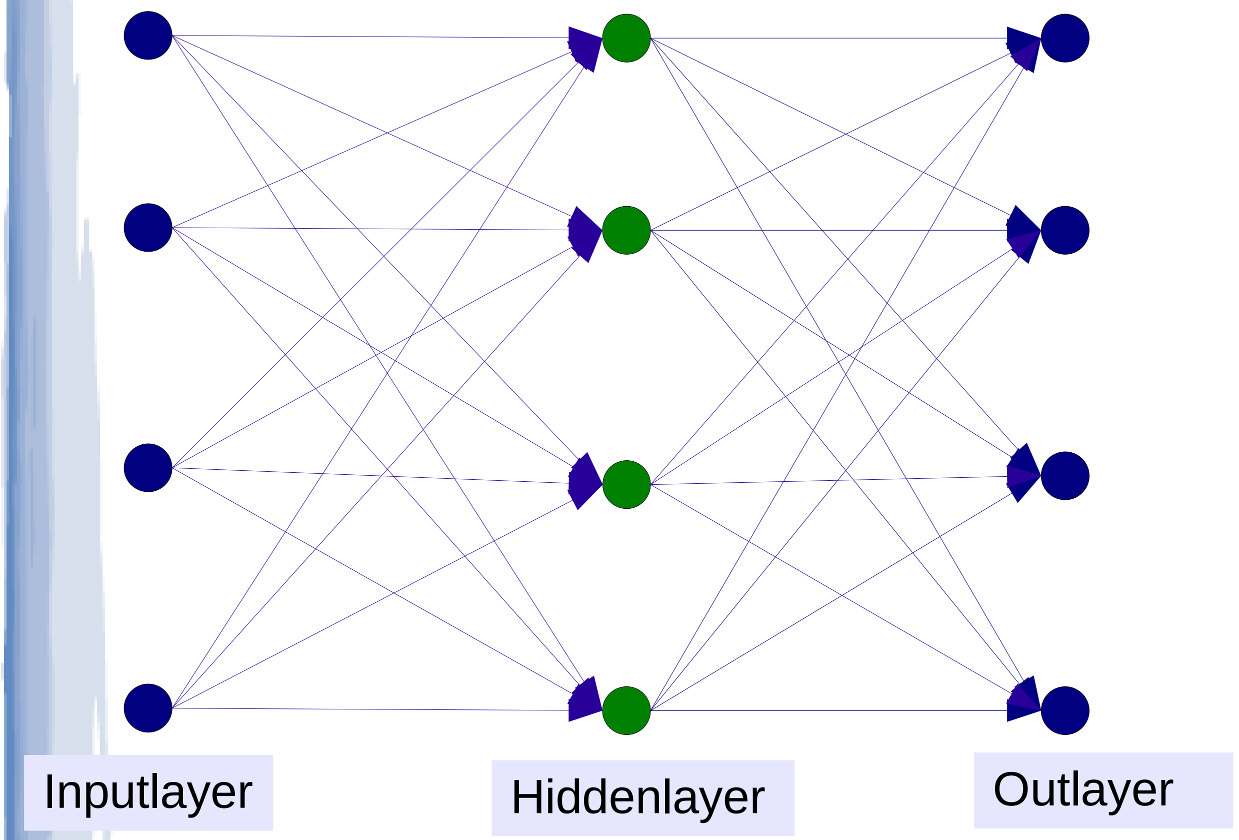
DAE learns a vector field (green arrows) which is a estimation of the gradient field $\nabla \log Q(x)$

Q(x) is the unknown data generating distribution

see [Alain and Bengio, 2012] [Alain and Bengio 2013]

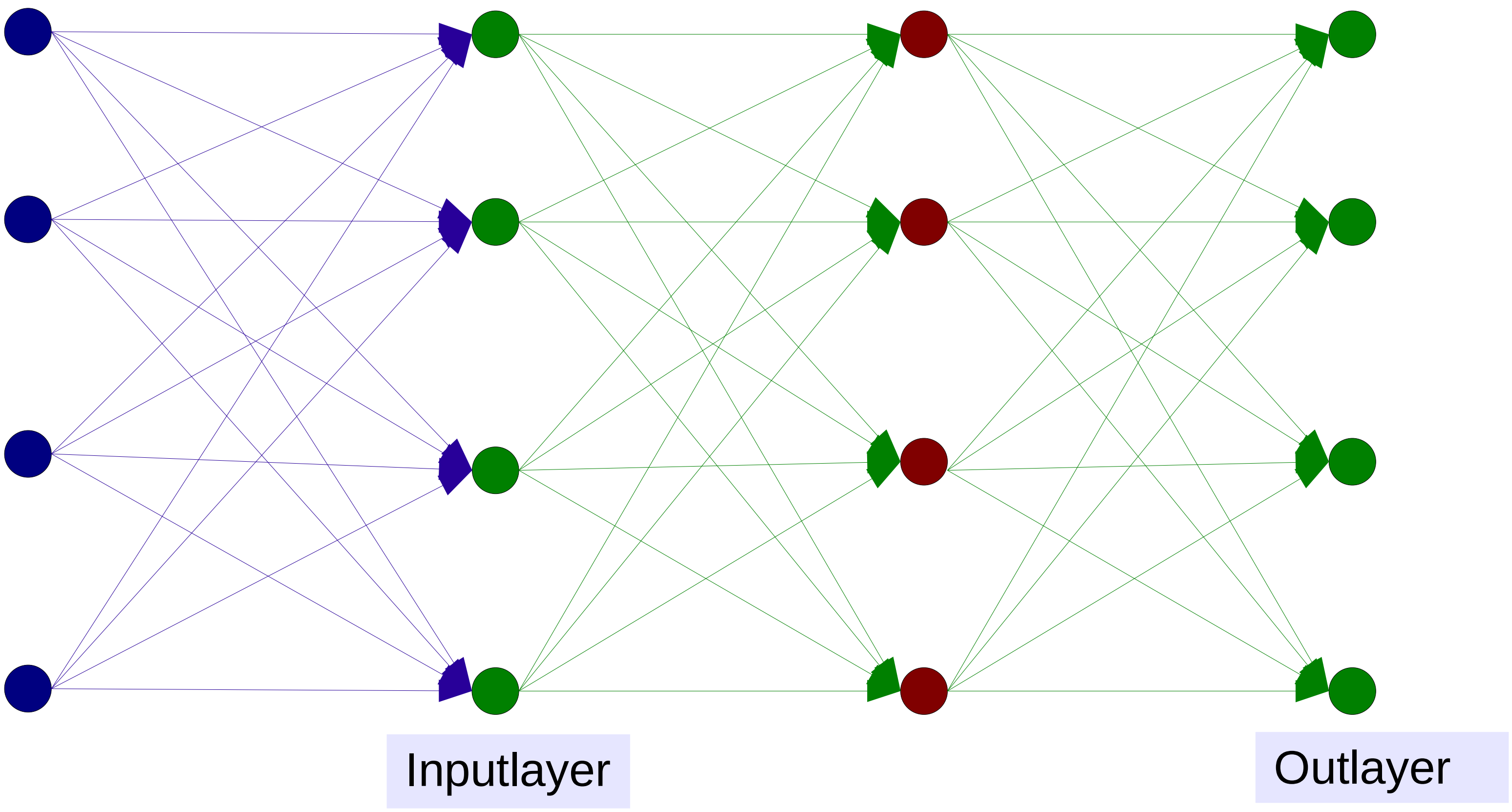
# Layer-wise pretraining

# Autoencoder



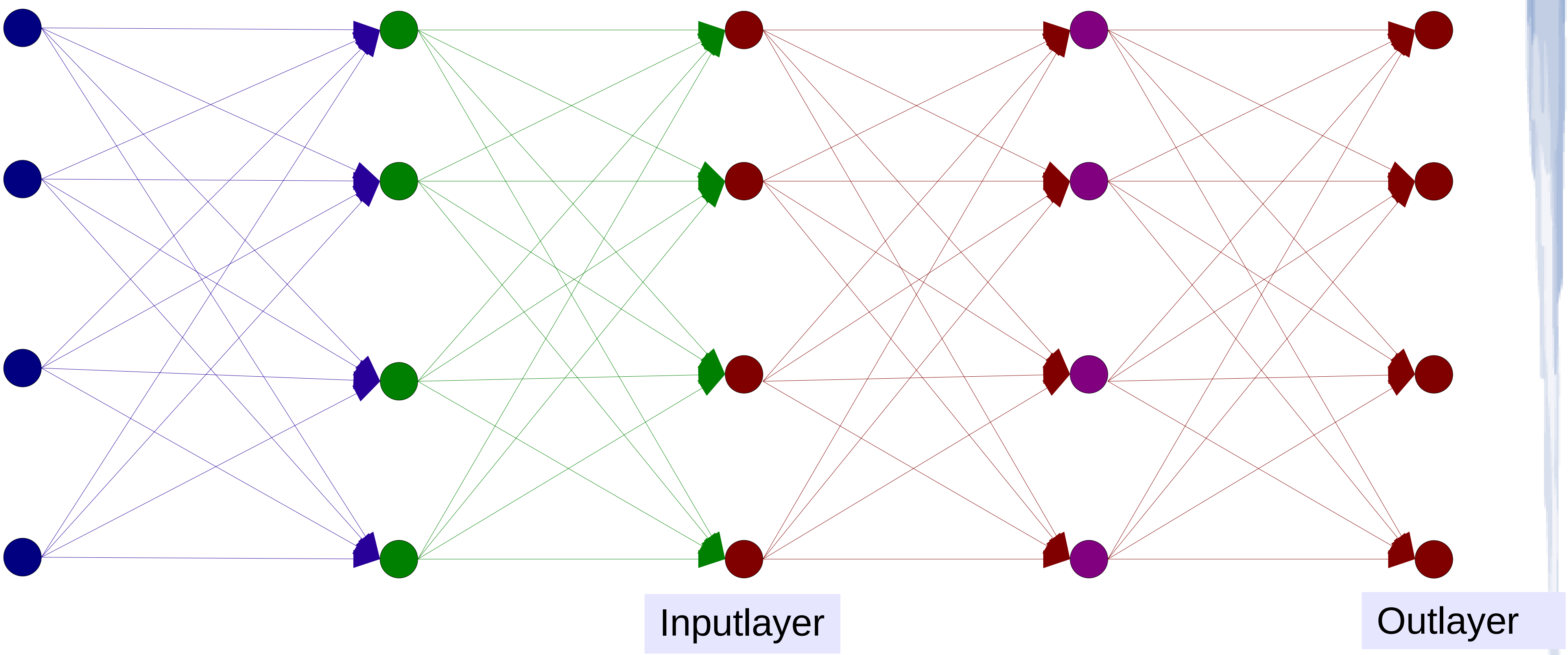

- unsupervised learning of the first layer

# Autoencoder



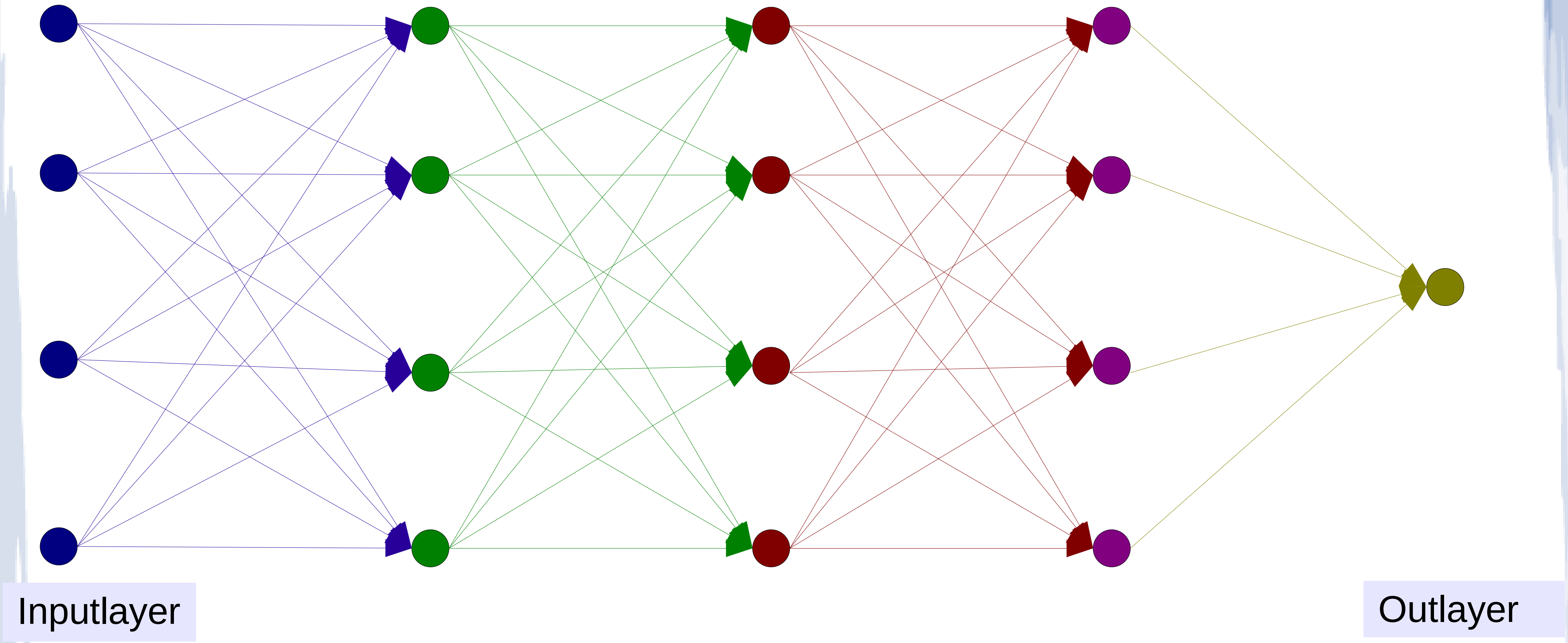

- unsupervised learning of the second layer

# Autoencoder



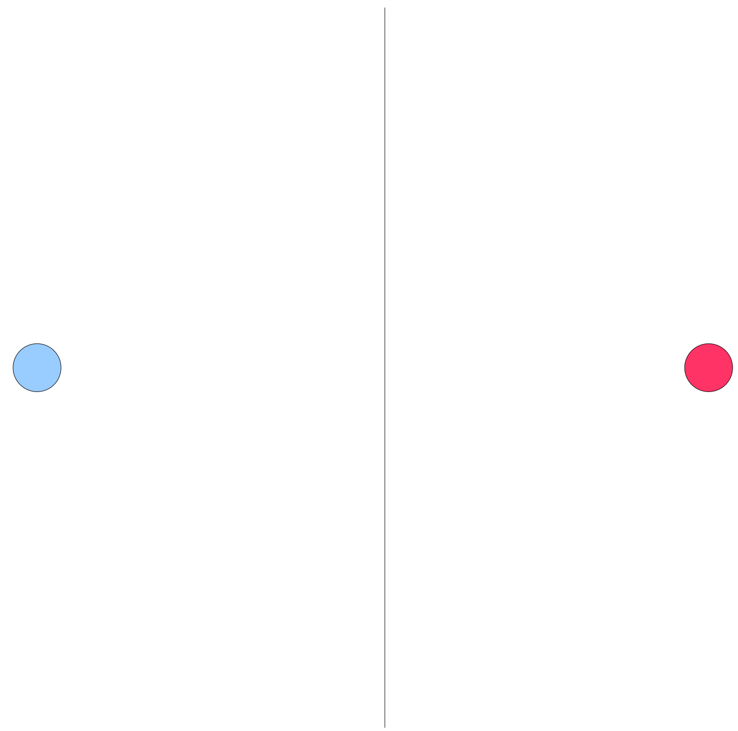

- unsupervised learning of the third layer

# Autoencoder

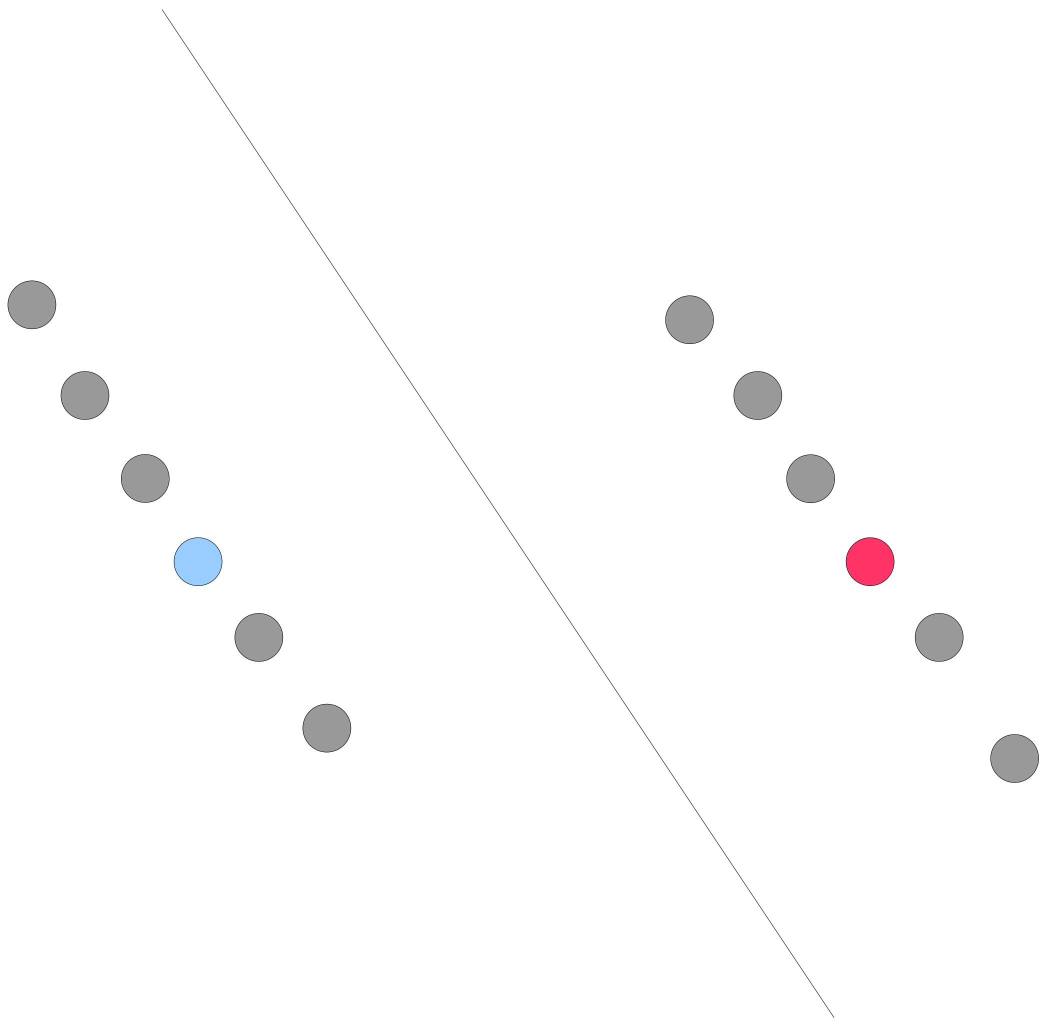



- supervised learning of the last layer

# purely supervised



# semi supervised



# Manifolds

# (Unsupervised) Manifold Hypothesis

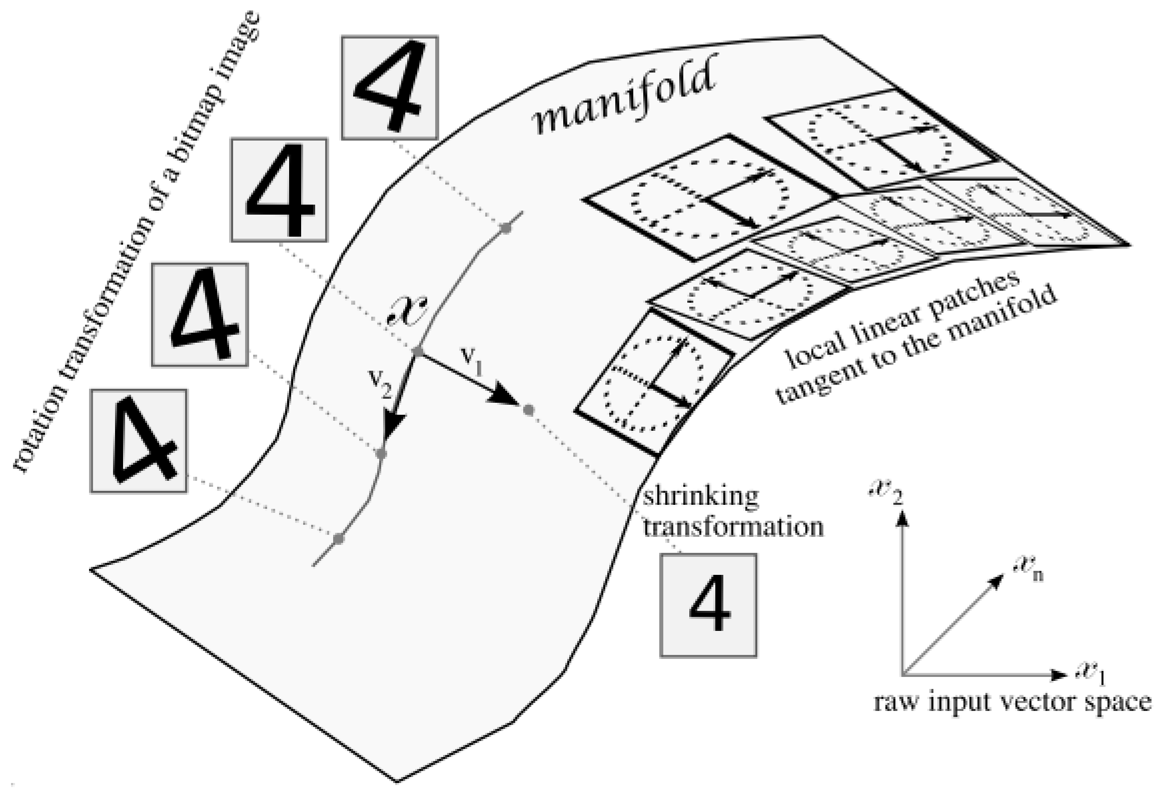
- data space extrem high dimensional
- natural data lives in a low-dimensional (non-linear) manifold, because variables in natural data are mutually dependent
- examples:
  - images vs. random pixels
  - different pictures of a face: dimension of the manifold smaller as:

    number of muscles + rotations- and translations degrees of freedom

# Manifold

- behaves locally like a Euclidean space
- definition in machine learning not so strict as in mathematics:
  - data is in the neighborhood of the manifold - not strictly on the manifold
  - dimensionality can vary for different regions in the embedding data space
  - also for discrete spaces (text processing)

# Manifold




from [Be09]

# manifold learning with regularized autoencoders

- two forces:
  - a) reduction of the reconstruction error
  - b) pressure to be insensitive to variations of the input space (due to additional regularization constraint)
- results in:
  - because of b): data points are mapped by the reconstruction (encoder-decoder) on the manifold in data space
  - because of a): different points are mapped to different locations on the manifold – they should be discriminable

# Explicit use of manifold hypotheses and tangent directions by the manifold tangent classifier [Rif11a]

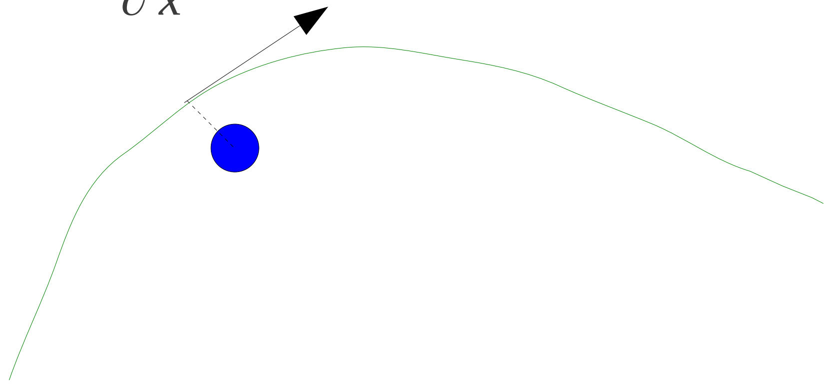
- Three Hypothesis:
  - semi-supervised learning hypothesis: learning of p(x) helps for models p(y|x)
  - unsupervised manifold hypothesis (also see slides above): data is concentrated on small sub-regions (sub-manifolds)
  - manifold hypothesis for classification: different classes concentrate along different sub-manifolds

# Learning of tanget directions with CAE(+H)

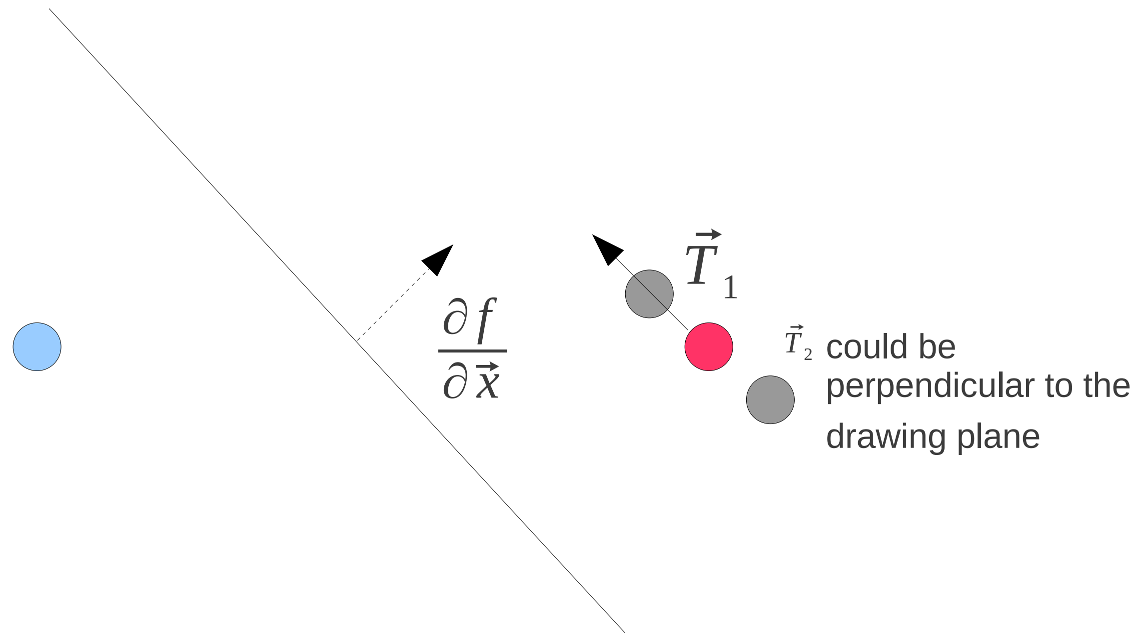
- the penalty of the CAE(+H) enforces that the encoder is only sensitive to "important" directions – directions on the manifold

$$T = \frac{\partial \vec{h}(\vec{x})}{\partial \vec{x}}$$

T = span (of the x first singular vectors) of the Jacobian at the data point



# Tangent Propagation Penalty




- Penalty $\sum_{T \in B_x} \left| \frac{\partial f(\vec{x})}{\partial \vec{x}} \cdot \vec{T} \right|^2$ forces that the gradient of the function (e.g. the nearby decision boundary for classification) is perpendicular to the tangent direction (local manifold patch) of the current data point x [Sim98]
- $f(\vec{x})$ is the output of the neural network
- Tangent directions $[\vec{T}_1, \vec{T}_2, \ldots \vec{T}_k]$ at each data point are computed from the Jacobian of the last layer representation of a CAE+H and its SVD (Singular Value decomposition) [Rif11a]

# Literature

General reference: Chaper "The Manifold Perspective on Autoencoder" of Deep Learning Book (in preparation for MIT Press) 2014; Yoshua Bengio and Ian J. Goodfellow and Aaron Courville

Ng's lecture notes to Sparse Autoencoder


- [Be09] Yoshua Bengio, Learning Deep Architectures for AI, Foundations and Trends in Machine Learning, 2(1), pp.1-127, 2009.
- [Glo11] Xavier Glorot, Antoine Bordes and Yoshua Bengio, Deep Sparse Rectifier Neural Networks, Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, 2011
- [Rif11] S. Rifal, P. Vincent, X. Muller, Y. Bengio; Contractive auto-encoders: explicit invariance during feature extraction. ICML 2011
- [Rif11a] S. Rifal, Y. Dauphin, P. Vincent, Y. Bengio, X. Muller; The Manifold Tangent Classifier, NIPS 2011
- [Vin10] Vincent, Pascal and Larochelle, Hugo and Lajoie, Isabelle and Bengio, Yoshua and Manzagol, Pierre-Antoine, Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion, J. Mach. Learn. Res., 2010

# Autoencoders with Theano

- Denoising Autoencoder:
  - http://deeplearning.net//tutorial/dA.html
  - http://deeplearning.net/tutorial/SdA.html
- Contractive Autoencoder
  - https://github.com/lisa-lab/DeepLearningTutorials/b