Deep Learning Summer School 2015

On manifolds and autoencoders

by Pascal Vincent

Montreal Institute for Learning Algorithms

August 5, 2015
PLAN

Part I: Leveraging the manifold hypothesis
Part II: Regularizing Auto-Encoders

Will be largely about unsupervised learning
An unsupervised learning task: dimensionality reduction

\[ h \in \mathbb{R}^M \quad M < D \]

\[(0.32, -1.3, 1.2)\]

\[ f^\theta \]

\[(3.5, -1.7, 2.8, -3, 5, -1.4, 2.4, 2.7, 7.5)\]

\[ x \in \mathbb{R}^D \]

What is it useful for?
An unsupervised learning task:
dimensionality reduction

What is it useful for?

- Data compression (lossy)
- Dataset visualisation (in 2D or 3D)
- Discovering «most important» features.
A classic algorithm [Pearson 1901] [Hotelling 1933]

Principal Component Analysis

- Finds (learns) \( k \) directions (a subspace) in which data has highest variance
  \( \Rightarrow \) \textit{principal directions} (eigenvectors) \( W \)

- Projecting inputs \( x \) on these vectors yields reduced dimension \textit{representation} (decorrelated)
  \( \Rightarrow \) \textit{principal components}

\[
h = f_\theta(x) = W(x - \mu) \quad \text{with} \quad \theta = \{W, \mu\}
\]
A classic algorithm [Pearson 1901] [Hotelling 1933]
Principal Component Analysis

- Finds (learns) k directions (a subspace) in which data has highest variance
  => principal directions (eigenvectors) $W$

- Projecting inputs $x$ on these vectors yields reduced dimension representation (unrelated)
  => principal components
  \[ h = f_\theta(x) = W(x-\mu) \] with $\theta=\{W,\mu\}$

Why mention PCA?
- Prototypical unsupervised representation learning algorithm.
- Related to autoencoders
- Prototypical manifold modeling algorithm
Lower-dimensional manifolds embedded in high dimensional space

Linear 2D manifold in 3D space (ex: subspace found by PCA)

Non-linear 2D manifold in 3D input space

Principal components are coordinates in a coordinate-system on the manifold
The manifold hypothesis
(assumption)

Natural data in high dimensional spaces concentrates close to lower dimensional manifolds.

Probability density decreases very rapidly when moving away from the supporting manifold.
The curse of dimensionality

There are $10^{96329}$ possible 200x200 RGB images.
• Natural images occupy a tiny fraction of that space  
  => suggests peaked density

• Realistic smooth transformations from one image to another  
  => continuous path along manifold
• Natural images occupy a tiny fraction of that space
  => suggests peaked density

• Realistic smooth transformations from one image to another
  => continuous path along manifold

The manifold hypothesis
Data density concentrates near a lower dimensional manifold
Can shift the curse from high $d$ to $d_M \ll d$
Manifold follows naturally from continuous underlying factors
(≈ intrinsic manifold coordinates)

Ex: pose parameters of a face

Ex: rotation, size of digits (+ line thickness, ...)

Such continuous factors are (part of) a meaningful representation!

Image borrowed from University of Dayton Vision Lab website.
Modeling local tangent spaces

A non-linear manifold

- Can be represented by patchwork of tangent spaces
- Yields local linear coordinate systems (chart -> atlas)
Non-parametric density estimation
Non-parametric density estimation

\[ \hat{p}(x) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{N}(x; x_i, C_i) \]

Classical Parzen Windows

density estimator
Non-parametric density estimation

\[ \hat{p}(x) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{N}(x; x_i, C_i) \]

Classical Parzen Windows density estimator

- Archetypal «non-parametric» kernel density estimator
- Isotropic Gaussian centered on each training point
- No sense of manifold direction
- Probability mass allocated away from manifold

mercredi 5 août 2015
Non-parametric density estimation

\[ \hat{p}(x) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{N}(x; x_i, C_i) \]

Classical Parzen Windows
density estimator

- Archetypal «non-parametric» kernel density estimator
- Isotropic Gaussian centered on each training point
- No sense of manifold direction
- Probability mass allocated away from manifold

Manifold Parzen Windows
density estimator

(Vincent and Bengio, NIPS 2003)

- Oriented Gaussian «pancake» centered on each training point
- Uses low-rank parametrization of \( C_i \), learned from nearest neighbors (local PCA)
- «Parametric» cousins:
  Mixtures of Gaussian pancakes (Hinton et al. 95)
  Mixtures of Factor Analysers (Gharamani + Hinton 96)
  Mixtures of Probabilistic PCA (Tipping + Bishop 99)
Non-local manifold Parzen windows

(Isotropic Parzen:
\[ \hat{p}(x) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{N}(x; x_i, \sigma^2 I) \]

isotropic)

(Bengio, Larochelle, Vincent, NIPS 2006)
Non-local manifold Parzen windows
(Bengio, Larochelle, Vincent, NIPS 2006)

Isotropic Parzen:
\[ \hat{p}(x) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{N}(x; x_i, \sigma^2 I) \]
\( n \) isotropic

Manifold Parzen:
(Vincent and Bengio, NIPS 2003)
\[ \hat{p}(x) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{N}(x; x_i, C_i) \]
\( d_M \) high variance directions from PCA on \( k \) nearest neighbors

mercredi 5 août 2015
Non-local manifold Parzen windows
(Bengio, Larochelle, Vincent, NIPS 2006)

Isotropic Parzen:
\[ \hat{p}(x) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{N}(x; x_i, \sigma^2 I) \]

Manifold Parzen:
(Vincent and Bengio, NIPS 2003)
\[ \hat{p}(x) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{N}(x; x_i, C_i) \]
\(d_M\) high variance directions from PCA on \(k\) nearest neighbors

Non-local manifold Parzen:
(Bengio, Larochelle, Vincent, NIPS 2006)
\[ \hat{p}(x) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{N}(x; \mu(x_i), C(x_i)) \]
\(d_M\) high variance directions output by neural network
trained to maximize likelihood of \(k\) nearest neighbors

mercredi 5 août 2015
Use in Bayes classifier on USPS

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Valid.</th>
<th>Test</th>
<th>Hyper-Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>1.2%</td>
<td>4.68%</td>
<td>$C = 100, \sigma = 8$</td>
</tr>
<tr>
<td>Parzen Windows</td>
<td>1.8%</td>
<td>5.08%</td>
<td>$\sigma = 0.8$</td>
</tr>
<tr>
<td>Manifold Parzen</td>
<td>0.9%</td>
<td>4.08%</td>
<td>$d = 11, k = 11, \sigma_0^2 = 0.1$</td>
</tr>
<tr>
<td>Non-local MP</td>
<td>0.6%</td>
<td><strong>3.64% (-1.5218)</strong></td>
<td>$d = 7, k = 10, k_\mu = 10, \sigma_0^2 = 0.05, n_{hid} = 70$</td>
</tr>
<tr>
<td>Non-local MP*</td>
<td>0.6%</td>
<td><strong>3.54% (-1.9771)</strong></td>
<td>$d = 7, k = 10, k_\mu = 4, \sigma_0^2 = 0.05, n_{hid} = 30$</td>
</tr>
</tbody>
</table>
Manifold learning is a rich subfield

**Purely non-parametric:**
- Manifold Parzen, LLE, Isomap, Laplacian eigenmaps, t-SNE, ... 

**Learned parametrized function:**
- Parametric t-SNE, semi-supervised embedding, non-local manifold Parzen, ...

What do all these approaches have in common?
Neighborhood-based training!

- They explicitly use distance-based neighborhoods.
- Training with k-nearest neighbors, or pairs of points.
- Typically Euclidean neighbors
- But in high $d$, your nearest Euclidean neighbor can be very different from you...
Neighborhood-based training!

- They explicitly use distance-based neighborhoods.
- Training with k-nearest neighbors, or pairs of points.
- Typically Euclidean neighbors.
- But in high \(d\), your nearest Euclidean neighbor can be very different from you...
Neighborhood-based training!

- They explicitly use distance-based neighborhoods.
- Training with k nearest neighbors, or pairs of points.
- Typically Euclidean neighbors
- But in high $d$, your nearest Euclidean neighbor can be very different from you...
PART II

On Auto-Encoders and their regularization,
Multi-Layer Perceptron (MLP) with one hidden layer of size \( d' \) neurons

Functional form (parametric):

\[
y = f_\theta(x) = \text{sigmoid} \left( \langle w, h \rangle + b \right)
\]

\[
h = \text{sigmoid} \left( W^{\text{hidden}} x + b^{\text{hidden}} \right)
\]

Parameters:

\[
\theta = \{ W^{\text{hidden}}, b^{\text{hidden}}, w, b \}
\]

Optimizing parameters on training set (training the network):

\[
\theta^* = \arg \min_{\theta} \hat{R}_\lambda(f_\theta, D_n)
\]

\[
\mathcal{J}_{\text{MLP}}(\theta) = \left( \sum_{(x,t) \in D} L(t, f_\theta(x)) \right) + \lambda \Omega(\theta)
\]

\[
\text{empirical risk} \quad \text{regularization term (weight decay)}
\]

mercredi 5 août 2015
Autoencoders: MLPs used for «unsupervised» representation learning

- Make output layer same size as input layer
- Have target = input
- Loss encourages output (reconstruction) to be close to input.

Autoencoders are also called
- Autoencoders
- Auto-associators
- Diabolo networks
- Sandglass-shaped net

The Diabolo

\[ L(x, r) \]

reconstruction \( r \)
hidden \( h \)
input \( x \)
Auto-Encoders (AE) for learning representations

hidden representation \( h = h(x) \)

Encoder: \( h \)

Decoder: \( g \)

input \( x \in \mathbb{R}^d \)

reconstruction \( r = g(h(x)) \)

reconstruction error \( L(x, r) \)

Minimize

\[ J_{AE} = \sum_{x \in D} L(x, g(h(x))) \]
Auto-Encoders (AE) for learning representations

Typical form

hidden representation \( h = h(x) \)

Encoder: \( h \)

Decoder: \( g \)

input \( x \in \mathbb{R}^d \)

reconstruction \( r = g(h(x)) \)

reconstruction error \( L(x, r) \)

Minimize

\[ J_{AE} = \sum_{x \in D} L(x, g(h(x))) \]
Auto-Encoders (AE) for learning representations

Typical form

hidden representation $h = h(x) = s(Wx + b) \in \mathbb{R}^{d_h}$

Encoder: $h$

Decoder: $g$

Input $x \in \mathbb{R}^d$

Reconstruction $r = g(h(x))$

Reconstruction error $L(x, r)$

Minimize

$J_{AE} = \sum_{x \in D} L(x, g(h(x)))$
Auto-Encoders (AE) for learning representations

Typical form

hidden representation \( h = h(x) = s(Wx + b) \) \( \in \mathbb{R}^{d_h} \)

Encoder: \( h \)

input \( x \in \mathbb{R}^d \)

Decoder: \( g \)

reconstruction \( r = g(h(x)) \)

reconstruction error \( L(x, r) \)

Minimize \( J_{AE} = \sum_{x \in D} L(x, g(h(x))) \)
Auto-Encoders (AE) for learning representations

Typical form

hidden representation $h = h(x) = s(Wx + b)$

Encoder: $h$

Decoder: $g$

Input $x \in \mathbb{R}^d$

Reconstruction $r = g(h(x)) = s_d(W'h + b_d)$

Reconstruction error $L(x, r)$

Squared error: $\|x - r\|^2$

or Bernoulli cross-entropy

Minimize

$\mathcal{J}_{AE} = \sum_{x \in D} L(x, g(h(x)))$
Auto-Encoders (AE) for learning representations

Typical form

hidden representation \( h = h(x) = s(Wx + b) \)  
\( s \) is typically sigmoid

Encoder:

\[ h \]

Decoder:

\[ g \]

input \( x \in \mathbb{R}^d \)

reconstruction \( r = g(h(x)) \)

\[ = s_d(W'h + b_d) \]

reconstruction error

squared error: \( \| x - r \|^2 \)
or Bernoulli cross-entropy

Minimize

\[ J_{AE} = \sum_{x \in D} L(x, g(h(x))) \]
conection between

Linear auto-encoders and PCA

d_{h} < d \ (bottleneck, undercomplete representation):

• With linear neurons and squared loss
  ➣ autoencoder learns same subspace as PCA

• Also true with a single sigmoidal hidden layer,
  if using linear output neurons with squared loss
  [Baldu& Hornik 89] and untied weights.

• Won’t learn the exact same basis as PCA,
  but \( W \) will span the same subspace.
Consider an auto-encoder MLP

- with a single hidden layer with sigmoid non-linearity
- and sigmoid output non-linearity.
- Tie encoder and decoder weights: $W' = W^T$.

**Autoencoder:**

\[
\begin{align*}
    h_i &= s(W_i x + b_i) \\
    r_j &= s(W_j^T h + b_{dj})
\end{align*}
\]

**RBM:**

\[
\begin{align*}
    P(h_i=1 \mid v) &= s(W_i v + c_i) \\
    P(v_j=1 \mid h) &= s(W_j^T h + b_j)
\end{align*}
\]

**Differences:**

- deterministic mapping
  - $h$ is a function of $x$.
- stochastic mapping
  - $h$ is a random variable
Greedy Layer-Wise Pre-training with RBMs

Stacking Restricted Boltzmann Machines (RBM) ➞ Deep Belief Network (DBN) [Hinton et al. 2006]
Greedy Layer-Wise Pre-training with Auto-Encoders

Stacking basic Auto-Encoders [Bengio et al. 2007]
Supervised fine-tuning

- Initial deep mapping was learnt in an **unsupervised** way.

- → **initialization** for a **supervised** task.

- **Output layer** gets added.

- Global fine tuning by gradient descent on **supervised criterion**.

\[
\text{supervised cost}
\]

\[
f_\theta \rightarrow f_\theta^{(2)} \rightarrow f_\theta^{(3)} \rightarrow f_\theta^{\text{sup}}
\]

\[
x \rightarrow \text{Target}
\]
Supervised Fine-Tuning is Important

- Greedy layer-wise unsupervised pre-training phase with RBMs or auto-encoders on MNIST
- Supervised phase with or without unsupervised updates, with or without fine-tuning of hidden layers

Classification performance on benchmarks:
- Pre-training basic auto-encoder stack better than no pre-training
- Basic auto-encoder stack almost matched RBM stack...
Basic auto-encoders not as good feature learners as RBMs...

What’s the problem?

- Traditional autoencoders were for **dimensionality reduction** ($d_h < d_x$)

- Deep learning success seems to depend on ability to learn **overcomplete representations** ($d_h > d_x$)

- Overcomplete basic autoencoder yields trivial useless solutions: identity mapping!

- Need for alternative **regularization/constraining**
Denoising auto-encoders: motivation
(Vincent, Larochelle, Bengio, Manzagol, ICML 2008)

* Simple idea «destroying information» of randomly selected input features; train to restore it.
  => 0-masking noise (now called «dropout» noise)

* Denoising corrupted input is a vastly more challenging task than mere reconstruction.

* Even in widely over-complete case... it must learn intelligent encoding / decoding.

* Will encourage representation that is robust to small perturbations of the input.
Denoising auto-encoder (DAE)

Encoder:

\[ h = h(x) \]

Decoder:

\[ g(h(x)) \]

Features:

(hidden representation)

Input \( x \) corrupted by noise \( q(\tilde{x}|x) \) leads to \( \tilde{x} \) which is encoded to \( h \). The reconstructed output \( r = g(h(x)) \) is compared to the ground truth \( x \) using the reconstruction error \( L(x, r) \).
Denoising auto-encoder (DAE)

Encoder: \( h = h(x) \)

Decoder: \( r = g(h(x)) \)

Input: \( x \)

Corrupted input: \( \tilde{x} \)

Noise: \( q(\tilde{x}|x) \)

Features: (hidden representation)

Reconstruction error: \( L(x, r) \)

Reconstruction: \( r = g(h(x)) \)

Minimize:

\[
J_{\text{DAE}}(\theta) = \sum_{x \in D} \mathbb{E}_{q(\tilde{x}|x)} [L(x, g(h(\tilde{x})))]
\]
Denoising auto-encoder (DAE)

features: \( h = h(x) \)
(hidden representation)

Encoder: \( h \)

Decoder: \( g \)

Corrupted input \( \tilde{x} \)

Noise \( q(\tilde{x}|x) \)

Input \( x \)

Reconstruction error \( L(x, r) \)

Reconstruction \( r = g(h(x)) \)

Minimize:
\[
J_{\text{DAE}}(\theta) = \sum_{x \in D} \mathbb{E}_{q(\tilde{x}|x)} [L(x, g(h(\tilde{x})))]
\]

- learns robust & useful features
- easier to train than RBM features
- yield similar or better classification performance (as deep net pre-training)
Denoising auto-encoder (DAE)

- Autoencoder training minimizes:
  \[ J_{AE}(\theta) = \sum_{x \in D} L(x, g(h(\tilde{x}))) \]

- Denoising autoencoder training minimizes
  \[ J_{DAE}(\theta) = \sum_{x \in D} \mathbb{E}_{q(\tilde{x} | x)} [L(x, g(h(\tilde{x})))] \]

Cannot compute expectation exactly

\(\Rightarrow\) use stochastic gradient descent,

**sampling corrupted inputs** \(\tilde{x}|x\)
Denoising auto-encoder (DAE)

- Autoencoder training minimizes:

\[ J_{AE}(\theta) = \sum_{x \in D} L(x, g(h(\tilde{x}))) \]

- Denoising autoencoder training minimizes

\[ J_{DAE}(\theta) = \sum_{x \in D} \mathbb{E}_{q(\tilde{x}|x)} [L(x, g(h(\tilde{x})))] \]

Cannot compute expectation exactly

⇒ use stochastic gradient descent,

**sampling corrupted inputs** \( \tilde{x}|x \)

**Possible corruptions** \( q \):
- zeroing pixels at random (now called «dropout» noise)
- additive Gaussian noise
- salt-and-pepper noise
- ...

mercredi 5 août 2015
Denoising auto-encoder (DAE)

Autoencoder training minimizes:
\[ J_{AE}(\theta) = \sum_{x \in D} L(x, g(h(\tilde{x}))) \]

Denoising autoencoder training minimizes
\[ J_{DAE}(\theta) = \sum_{x \in D} \mathbb{E}_{q(\tilde{x}|x)} [L(x, g(h(\tilde{x})))] \]

Cannot compute expectation exactly
\[ \Rightarrow \text{use stochastic gradient descent, sampling corrupted inputs } \tilde{x}|x \]

Possible corruptions q:
- zeroing pixels at random (now called «dropout» noise)
- additive Gaussian noise
- salt-and-pepper noise
- ...
Learned filters

<table>
<thead>
<tr>
<th>AE</th>
<th>AE with weight decay</th>
<th>DAE</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="a" alt="Images" /> Natural image patches e.g.: <img src="b" alt="Images" /></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

From this experiment, we see clearly that some local blob detectors are recovered compared to using no weight decay. We tried a wide range of weight-decay values and learning rates. For example, with the noise level of 0.05, we did not get any interesting structure. However, with increased noise levels, we observed a much larger proportion of edge detectors and grating filters. Clearly, different co-occurrences of these feature detectors are learnt. The filters shown were obtained using 100 hidden units, but similar filters were also found with 50 or 200 hidden units.

We then trained 200 hidden units over-complete noiseless autoencoders, all starting from the same initialization. Salt-and-pepper noise yielded Gabor-like edge detectors. We experimented with 3 corruption types and levels. Expected that denoising a regular autoencoder with additive Gaussian noise (but no weight decay) resembles simple cell receptive fields from the primary visual cortex.

Cross-entropy reconstruction error, and zero-masking noise. The goal was to better understand the autoencoder learns Gabor-like local oriented edge detectors.
Learned filters

b) MNIST digits

e.g.: 4 3 8 7

AE

DAE

Increasing noise

(d) Neuron A (0%, 10%, 20%, 50% corruption)

(e) Neuron B (0%, 10%, 20%, 50% corruption)
Denoising auto-encoders: manifold interpretation

- DAE learns to «project back» corrupted input onto manifold.
- Representation $h \approx$ location on the manifold

prior: examples concentrate near a lower dimensional “manifold”
Stacked Denoising Auto-Encoders (SDAE)

- No partition function, can measure training criterion
- Very flexible: encoder & decoder can use any parametrization (more layers...)
- Performs as well or better than stacking RBMs for unsupervised pre-training
Encouraging representation to be insensitive to corruption

- DAE encourages **reconstruction** to be insensitive to input corruption
- Alternative: encourage **representation** to be **insensitive**

\[
J_{SCAE}(\theta) = \sum_{x \in D} L(x, g(h(x))) + \lambda \mathbb{E}_{q(\tilde{x}|x)} \left[ \|h(x) - h(\tilde{x})\|^2 \right]
\]

- **Reconstruction error**
- **Stochastic regularization term**

- Tied weights i.e. \( W' = W^T \) prevent \( W \) from collapsing \( h \) to 0.
Encouraging representation to be insensitive to corruption

- DAE encourages **reconstruction** to be insensitive to input corruption
- Alternative: encourage **representation** to be insensitive

\[ J_{SCAE}(\theta) = \sum_{x \in D} E(x, g(h(x))) + \lambda \mathbb{E}_{q(\tilde{x}|x)} [\| h(x) - h(\tilde{x}) \|^2] \]

Reconstruction error + stochastic regularization term

- Tied weights i.e. \( W = W^T \) prevent \( W \) from collapsing \( h \) to 0.
From stochastic to analytic penalty

* SCAE stochastic regularization term: \( \mathbb{E}_q(\tilde{x}|x) \left[ \| h(x) - h(\tilde{x}) \|^2 \right] \)

* For small additive noise \( \tilde{x}|x = x + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2 I) \)

* Taylor series expansion yields \( h(x + \epsilon) = h(x) + \frac{\partial h}{\partial x} \epsilon + \ldots \)

* It can be showed that

\[
\mathbb{E}_q(\tilde{x}|x) \left[ \| h(x) - h(\tilde{x}) \|^2 \right] \approx \sigma^2 \left\| \frac{\partial h}{\partial x}(x) \right\|_F^2
\]

stochastic (SCAE)  analytic (CAE)
Contractive Auto-Encoder (CAE)

(Rifai, Vincent, Muller, Glorot, Bengio, ICML 2011)

Minimize \( \mathcal{J}_{\text{CAE}} = \sum_{x \in D} L(x, g(h(x))) + \lambda \left\| \frac{\partial h(x)}{\partial x} \right\|^2 \)

- Reconstruction error
- Analytic contractive term

For training examples, encourages both:
- small reconstruction error
- representation insensitive to small variations around example
Contractive Auto-Encoder (CAE)

(Rifai, Vincent, Muller, Glorot, Bengio, ICML 2011)

Minimize \( J_{CAE} = \sum_{x \in D} L(x, g(h(x))) + \lambda \left\| \frac{\partial h(x)}{\partial x} \right\|_2^2 \)

For training examples, encourages both:
- small reconstruction error
- representation insensitive to small variations around example
Computational considerations
CAE for a simple encoder layer

We defined \( h = h(x) = s(Wx + b) \)

Further suppose: \( s \) is an elementwise non-linearity
\( s' \) its first derivative.

Let \( J(x) = \frac{\partial h}{\partial x}(x) \)

\[ J_j = s'(b + x^T W_j) W_j \]

where \( J_j \) and \( W_j \) represent \( j^{th} \) row

CAE penalty is:
\[
\| J \|_F^2 = \sum_{j=1}^{d_h} s'(a_j)^2 \| W_j \|_2^2
\]

Same complexity: \( O(d_h d) \)

Compare to L2 weight decay:
\[
\| W \|_F^2 = \sum_{j=1}^{d_h} \| W_j \|_2^2
\]

Gradient backprop wrt parameters: \( O(d_h d) \)
Higher order Contractive Auto-Encoder (CAE+H)

(Rifai, Mesnil, Vincent, Muller, Bengio, Dauphin, Glorot; ECML 2011)

- CAE penalizes Jacobian norm
- We could also penalize higher order derivatives
- Computationally too expensive: second derivative is a 3-tensor, ...
- **Stochastic approach for efficiency:**

  Encourage Jacobian at $x$ and at $x+\epsilon$ to be the same.

  \[
  J_{\text{CAE+H}} = \sum_{x \in D}^{n} L(x, g(h(x))) + \lambda \left\| \frac{\partial h}{\partial x}(x) \right\|^2 \\
  + \gamma \mathbb{E}_{\epsilon \sim \mathcal{N}(0, \sigma^2)} \left[ \left\| \frac{\partial h}{\partial x}(x) - \frac{\partial h}{\partial x}(x + \epsilon) \right\|^2 \right]
  \]
Higher order Contractive Auto-Encoder (CAE+H)

(Rifai, Mesnil, Vincent, Muller, Bengio, Dauphin, Glorot; ECML 2011)

- CAE penalizes Jacobian norm
- We could also penalize higher order derivatives
- Computationally too expensive, second derivative is a 3-tensor, ...
- **Stochastic approach for efficiency:**
  Encourage Jacobian at $x$ and at $x+\varepsilon$ to be the same.

$$ J_{CAE+H} = \sum_{x \in D} L(x, g(h(x))) + \lambda \left\| \frac{\partial h}{\partial x}(x) \right\|^2 + \gamma \mathcal{E} \sim \mathcal{N}(0, \sigma^2) \left[ \left\| \frac{\partial h}{\partial x}(x) - \frac{\partial h}{\partial x}(x + \varepsilon) \right\|^2 \right] $$

Stochastic & analytic regularization
Learned filters

- AE
- DAE
- CAE
- CAE+H
CAE must capture manifold directions

\[ J_{CAE} = \sum_{x \in D}^{n} L(x, g(h(x))) + \lambda \left\| \frac{\partial h(x)}{\partial x} \right\|^2 \]

Reconstruction \(\iff\) tradeoff \(\Rightarrow\) Contraction

(warning: may require tied weights)

pressure to be insensitive to \textit{all} directions
CAE must capture manifold directions

\[ J_{\text{CAE}} = \sum_{x \in D} L(x, g(h(x))) + \lambda \left\| \frac{\partial h(x)}{\partial x} \right\|^2 \]

Reconstruction \iff tradeoff \iff Contraction

(reconstruction: may require tied weights)

pressure to be insensitive to all directions
CAE must capture manifold directions

\[ J_{\text{CAE}} = \sum_{x \in D}^{n} L(x, g(h(x))) + \lambda \left\| \frac{\partial h(x)}{\partial x} \right\|^2 \]

Reconstruction \iff \text{tradeoff} \iff \text{Contraction}

(Warning: may require tied weights)

Pressure to be insensitive to all directions
CAE must capture manifold directions

\[ J_{\text{CAE}} = \sum_{x \in D}^{n} L(x, g(h(x))) + \lambda \left\| \frac{\partial h(x)}{\partial x} \right\|^2 \]

Reconstruction \iff \text{tradeoff} \implies \text{Contraction}

(Warning: may require tied weights)

Pressure to be insensitive to \textit{all} directions
CAE must capture manifold directions

\[ J_{CAE} = \sum_{x \in D} L(x, g(h(x))) + \lambda \left\| \frac{\partial h(x)}{\partial x} \right\|^2 \]

**Reconstruction**  \(\Leftarrow\) tradeoff  \(\Rightarrow\)  **Contraction**

(Warning: may require tied weights)

Pressure to be insensitive to *all* directions
CAE must capture manifold directions

\[ J_{\text{CAE}} = \sum_{x \in D}^{n} L(x, g(h(x))) + \lambda \left\| \frac{\partial h(x)}{\partial x} \right\|^2 \]

Reconstruction ⇐ tradeoff ⇒ Contraction

pressure to be insensitive to \textit{all} directions

(\textit{warning: may require tied weights})

mercredi 5 août 2015
Learned tangent space

- Jacobian \( J_h(x) = \frac{\partial h}{\partial x}(x) \) measures sensitivity of \( h \) locally around \( x \).

SVD:

\[
\frac{\partial h(x)^T}{\partial x} = U S V^T
\]

Top singular vectors are tangent directions to which \( h \) is most sensitive.

\[
T_x = \{ U_k | S_{kk} > \epsilon \}
\]
SVD of \( J_h(x) = \frac{\partial h}{\partial x}(x) \)

CIFAR-10

Jacobi singular values

# singular values

○ AE

◆ CAE

mercredi 5 août 2015
Learned tangents CIFAR-10

- Input Point
- Tangents

Local PCA (as e.g. in Manifold Parzen)

Contractive Auto-Encoder (singular vectors of $J_h(x)$)

Not based on explicit neighbors or pairs of points!
Learned tangents:

Input Point

Local PCA (as e.g. in Manifold Parzen)

Contractive Auto-Encoder (singular vectors of $J_h(x)$)

Not based on explicit neighbors or pairs of points!
How to leverage the learned tangents

- Simard et al, 1993 exploited tangents derived from prior-knowledge of image deformations we can use our learned tangents instead.

- Use them to define **tangent distance** to use in your favorite distance (k-NN) or kernel-based classifier...

- Use them with **tangent propagation** when fine-tuning a deep-net classifier to make class prediction insensitive to tangent directions. *(Manifold Tangent Classifier, Rifai et al. NIPS 2011)* 0.81% on MNIST

- Moving preferably along tangents allows **efficient quality sampling**
How to leverage the learned tangents

- **Simard et al, 1993** exploited tangents derived from prior-knowledge of image deformations, we can use our learned tangents instead.

- Use them to define *tangent distance* to use in your favorite distance (k-NN) or kernel-based classifier...

- Use them with *tangent propagation* when fine-tuning a deep-net classifier to make class prediction insensitive to tangent directions. *(Manifold Tangent Classifier, Rifai et al. NIPS 2011)* 0.81% on MNIST

- Moving preferably along tangents allows *efficient quality sampling*
Analytic v.s. stochastic?

a) Analytic approximation of stochastic perturbation

• - Equiv. to tiny perturbations: does not probe far away

• + Potentially more efficient. Ex:
  CAE’s Jacobian penalty probes sensitivity in all $d$ directions in $O(d_h d)$
  With DAE or SCAE it would require encoding $d$ corrupted inputs: $O(d_h d^2)$

b) Stochastic approximation of analytic criterion

• + can render practical otherwise computationally infeasible criteria
  Ex: CAE+H

• - less precise, more noisy

CAE+H actually leverages both
Score matching
(Hyvärinen 2005)

We want to learn a p.d.f.

\[ p_{\theta}(x) = \frac{1}{Z(\theta)} e^{-E_{\theta}(x)} \]

with intractable partition function \( Z \)

Score matching: alternative inductive principle to max. likelihood

Find parameters that minimize objective:

\[
J_{SM}(\theta) = \sum_{x \in D} \left( \left\| \frac{\partial E}{\partial x}(x) \right\|^2 - \sum_{i=1}^{d} \frac{\partial^2 E}{\partial x_i^2}(x) \right)
\]
Score matching
my geometric interpretation

\[ J_{SM}(\theta) = \sum_{x \in D} \left( \left\| \frac{\partial E}{\partial x}(x) \right\|^2 - \sum_{i=1}^{d} \frac{\partial^2 E}{\partial x_i^2}(x) \right) \]
Score matching
my geometric interpretation

\[ J_{SM}(\theta) = \sum_{x \in D} \left( \left\| \frac{\partial E}{\partial x}(x) \right\|^2 - \sum_{i=1}^{d} \frac{\partial^2 E}{\partial x_i^2}(x) \right) \]

\[ \| J_E(x) \|^2 \]
Score matching
my geometric interpretation

\[ J_{SM}(\theta) = \sum_{x \in D} \left( \left\| \frac{\partial E}{\partial x}(x) \right\|^2 - \sum_{i=1}^{d} \frac{\partial^2 E}{\partial x_i^2}(x) \right) \]

First derivative encouraged to be small: ensures training points stay close to local minima of E
Score matching
my geometric interpretation

\[ J_{SM}(\theta) = \sum_{x \in D} \left( \left\| \frac{\partial E}{\partial x}(x) \right\|^2 - \sum_{i=1}^{d} \frac{\partial^2 E}{\partial x_i^2}(x) \right) \]

First derivative encouraged to be small: ensures training points stay close to local minima of E
Score matching  
my geometric interpretation

\[ J_{SM}(\theta) = \sum_{x \in D} \left( \left\| \frac{\partial E}{\partial x}(x) \right\|^2 - \sum_{i=1}^{d} \frac{\partial^2 E}{\partial x_i^2}(x) \right) \]

First derivative encouraged to be small: ensures training points stay close to local minima of \( E \)

Encourage large positive curvature in all directions

\[ \text{Tr}(H_E(x)) \]
Score matching
my geometric interpretation

\[ J_{SM}(\theta) = \sum_{x \in D} \left( \| \frac{\partial E}{\partial x}(x) \|^2 - \sum_{i=1}^{d} \frac{\partial^2 E}{\partial x^2_i}(x) \right) \]

First derivative encouraged to be small: ensures training points stay close to local minima of \( E \)

Encourage large positive curvature in all directions
Score matching
my geometric interpretation

\[ J_{SM}(\theta) = \sum_{x \in D} \left( \left\| \frac{\partial E}{\partial x}(x) \right\|^2 - \sum_{i=1}^{d} \frac{\partial^2 E}{\partial x_i^2}(x) \right) \]

First derivative encouraged to be small: ensures training points stay close to local minima of \( E \)

Encourage large positive curvature in all directions

\[ \| J_E(x) \|^2 \]

\[ \text{Tr}(H_E(x)) \]

\( e^{-E(x)} \)

sharply peaked density

mercredi 5 août 2015
Score matching variants

Original score matching (Hyvärinen 2005):

\[ J_{SM}(\theta) = \sum_{x \in D} \left( \frac{1}{2} \left\| \frac{\partial E}{\partial x}(x) \right\|^2 - \sum_{i=1}^{d} \frac{\partial^2 E}{\partial x_i^2}(x) \right) \]

Regularized score matching (Kingma & LeCun 2010):

\[ J_{SM_{reg}}(\lambda)(\theta) = J_{SM} + \sum_{x \in D} \lambda \sum_{i=1}^{d} \frac{\partial^2 E}{\partial x_i^2}(x) \]

Denoising score matching (Vincent 2011)

\[ J_{DSM}(\sigma) = \sum_{x \in D} \left( \mathbb{E}_{\epsilon \sim \mathcal{N}(0, \sigma^2 I)} \left[ \frac{1}{2} \left\| \frac{\partial E}{\partial x}(x + \epsilon) - \frac{1}{\sigma^2} \epsilon \right\|^2 \right] \right) \]
DAE training has a deeper relationship to RBMs

- Same functional form as RBM:
  - $h(x)$ is expected hidden given visible
  - $g(h)$ is expected visible given hidden

- With linear reconstruction and squared error, DAE amounts to learning the following energy

$$E(x; W, b, c) = -\frac{1}{2} \|x\|^2 + \sum_{j=1}^{d_h} \text{softplus} \left( \langle W_j, x \rangle + b_j \right)$$

using the denoising score matching inductive principle.

- Above energy closely related to free energy of Gaussian-binary RBM (identical for $\sigma=1$)
Questions ?