Medical Image Analysis, 2017
Machine learning 3

KALLE ÅSTRÖM
Contents

• Previous lecture:
  – Visualization, dimensionality reduction
    » The many applications of SVD
    » Multi-dimensional scaling
    » ISOMAP – non-linear dimensionality reduction
• Previous course?:
  – Deep Learning for classification (review)
• This lecture
• Deep Learning for
  – Autoencoder: dimensionality reduction, noise reduction, image inpainting and visualization
  – Segmentation: U-net, Inception, etc.
Applications of Singular Value Decomposition (SVD)

• The singular value decomposition
• $M = U S V^T$
• Essentially a unique factorization
• $U$ and $V$ rotation matrices (unitary matrices)
• $S$ diagonal with decreasing non-negative diagonal elements called singular values
Applications of Singular Value Decomposition (SVD)

• Find vector $x$ that ’solves’ $Ax = 0$
• If $A$ has fewer rows than columns
  – Underdetermined, many solutions
• If $A$ is overdetermined find $x$ that gives smallest $|Ax|$
  – (But smaller $x$ gives better results -> constrain $x$)
• Minimize $|Ax|$ under the constraint that $|x|=1$
• Is solved by setting $x$ to last column of $V$,
  – Where $A = U S V^T$
Applications of Singular Value Decomposition (SVD)

- Given data matrix $A$,
  - find rank $k$ matrix $A_k$ that is closest to $A$
- Solution
  - Make singular value decomposition
  - $A = U S V^T$
  - Let $U_k$ be first $k$ columns of $U$
  - Let $V_k$ be first $k$ columns of $V$
  - Let $S_k$ be upper-left $k \times k$ submatrix of $S$
  - $A_k = U_k S_k V_k^T$
  - The matrix $A_k$ is the solution to the minimization problem
Applications of Singular Value Decomposition (SVD)

• Find optimal basis (and coordinates) for a set of images
• “Optimal (linear) dimensionality reduction
• Solution
  – Put images as columns in matrix A, remove mean first
  – Make singular value decomposition
  – $A = U S V^T$
  – Possibly make best low rank approximation
    » $A_k = U_k S_k V_k^T$
  – $U_k$ are basis for optimal subspace of dimension $k$
  – $X_k = S_k V_k^T$ are optimal coordinate
Applications of Singular Value Decomposition (SVD)

• Given symmetric matrix $A$,
  – Factorize $A = BB^T$

• Solution
  – Make singular value decomposition
  – $A = U S V^T$
  – Take the square root of $S = D^2$
  – Set $B = U D$
  – $A = BB^T$
Multi-Dimensional Scaling

- Assume that 'distances' 'similarities' are measured between each pair (i,j) of feature vectors $(x_1, x_2, \ldots, x_n)$

- Can you reconstruct the feature vectors

  $$(x_1, x_2, \ldots, x_n)$$

- from the interpoint distances

  $$d_{ij} = |x_i - x_j|$$
Multi-Dimensional Scaling - The trick

- Choose coordinate system so that $x_1 = 0$
- Square the distances $t_{ij} = d_{ij}^2 = (x_i - x_j)^T(x_i - x_j)$
  $$t_{ij} = d_{ij}^2 = x_i^T x_i + x_j^T x_j - 2x_i^T x_j$$
  $$t_{1i} = x_i^T x_i \quad t_{1j} = x_j^T x_j$$
- Form a new matrix
  $$s_{ij} = -(t_{ij} - t_{1i} - t_{1j})/2 = -x_i^T x_j$$
  $$s = \begin{pmatrix} x_1^T \\ x_2^T \\ x_3^T \\ \vdots \\ x_n^T \end{pmatrix} \begin{pmatrix} x_2 & x_3 & \ldots & x_n \end{pmatrix}$$
Multi-Dimensional Scaling

\[ S = \begin{pmatrix} x_2^T \\ x_3^T \\ \vdots \\ x_n^T \end{pmatrix} \begin{pmatrix} x_2 & x_3 & \ldots & x_n \end{pmatrix} = X^T X \]

- Use singular value decomposition to calculate \( X \) from \( S \) (see previous slide)
MDS example
Non-linear dimensionality reduction

• Many different methods, see

• Examples
  – Kernel PCA
  – Local linear embedding
  – ISOMAP

• Many similarities.
ISOMAP

• Idea (illustrate on blackboard)
  – For each point choose the k nearest neighbours
  – Form weighted graph using distance to the k nearest neighbours
  – Calculate distance matrix D containing all distances $d_{ij}$ of pairs of feature vectors using shortest distance in graph.
  – Use Multi-dimensional scaling to embed points in e.g. $\mathbb{R}^2$
Nonlinear dimensionality reduction

Nonlinear dimensionality reduction

Source: http://isomap.stanford.edu/datasets.html
Nonlinear dimensionality reduction

Source: http://isomap.stanford.edu/datasets.html
Contents

• Visualization, dimensionality reduction
  – The many applications of SVD
  – Multi-dimensional scaling
  – ISOMAP – non-linear dimensionality reduction
• Deep Learning for classification (review)
• Deep Learning for dimensionality reduction and visualization
• U-net
Deep learning
Convolutional Neural Networks

• Slides and material from
  http://www.cs.nyu.edu/~yann/talks/lecun-ranzato-
icml2013.pdf

• MatConvNet
  http://www.robots.ox.ac.uk/~vgg/practicals/cnn/

• Gabrielle Flood’s master’s thesis
• Anna Gummeson’s master’s thesis
• Konrad Gjertsson’s master’s thesis
Components for deep learning

- One neuron
  - Example: Logistic regression
  - Classification model (x feature vector, (w,b) parameters, s smooth thresholding
    \[ x \in \mathbb{R}^d, w \in \mathbb{R}^d, b \in \mathbb{R}, f(x) = s(w^T x + b) \]
  - Logistic regression
    \[ s(z) = \frac{1}{1 + e^{-x}} \]
  - ML estimate of parameters (w,b) is a convex optimization problem
    \[ \min_w \frac{1}{2} w^T w + C \sum_{i=1}^{l} \log(1 + e^{-y_i w^T x_i}). \]
Single Layer Neural Networks

One Neuron

• One neuron

\[ x \in \mathbb{R}^d, w \in \mathbb{R}^d, b \in \mathbb{R}, f(x) = s(w^T x + b) \]
Single Layer Neural Networks
Several Neurons

• Several parallel neurons
  \( x \in \mathbb{R}^d, \ y \in \mathbb{R}^k, \ B \in \mathbb{R}^d, \ W - k \times d \text{ matrix} \)
  \[ y = s(Wx + B) \]

• Elementwise smooth thresholding – \( s \)
Artificial Neural Networks

One hidden layer

- Multi-class classification
- One hidden layer
- Trained by back-propagation
- Popular since the 1990ies
Deep Neural Networks
Many layers

- However
- Naively implemented would give to many parameters
- Example
  - 1M pixel image
  - 1M hidden layers
  - $10^{12}$ parameters between each pairs of layers
Convolutional neural network, CNN
CNN-Blocks - Convolutional layer

Convolution of an image as a filter-operation.

Convolution of an image represented as a sparsely connected ANN.
CNN-Blocks - Convolutional layer

- Input: Data block $x$ of size $m \times n \times k_1$
- Output: Data block $y$ of size $m \times n \times k_2$
- Filter: Filter kernel block $w$ of size $m_w \times n_w \times k_1 \times k_2$
- Offsets: Vector $w_o$ of length $k_2$

$$y(i, j, k) = w_o(k) + \sum_u \sum_v \sum_l x(i - u, j - v, l)w(u, v, l, k)$$
CNN-Blocks - Convolutional layer

\[ y(i, j) = \sum_u \sum_v x(i - u, j - v) w(u, v) \]

\[ y(i, j) = w_0 + \sum_l \left( \sum_u \sum_v x(i - u, j - v, l) w(u, v, l) \right) \]

\[ y(i, j, k) = w_o(k) + \sum_u \sum_v \sum_l x(i - u, j - v, l) w(u, v, l, k) \]
CNN-Blocks - Max-pooling
CNN-Blocks - RELU

\[ f(x) = \max(0, x) \]

\[ y(i, j, k) = \max(x(i, j, k), 0) \]
CNN-Blocks - Softmax

\[ p_j = \frac{e^{d_j}}{\sum_{k=1}^{m} e^{d_k}} \]
Result, Network design
**CNN-Blocks - Softmax**

\[ y = f(x, w) \]

Input: image \( x \) of size \( m \times n \times k \), typically \( k=1 \) (gray-scale) or \( k=3 \) (colour)

Output: vector \( y \) of size \( 1 \times 1 \times N \), which we interpret as \( N \) probabilities \( y_j \)

The probability that the image \( x \) is of class \( j \)
Training data $T = \{(x_1, c_1), \ldots (x_N, c_N)\}$

$y = f(x, w)$

Input: image $x$ of size $m \times n \times k$, typically $k=1$ (gray-scale) or $k=3$ (colour)

Output: vector $y$ of size $1 \times 1 \times N$, which we interpret as $N$ probabilities $y_j$

The probability that the image $x$ is of class $j$

If the class is supposed to be the integer $c_i$ for image $x_i$
then we want $y_{c_i}$

To be large (close to one)

Minimize $\sum_{i=1}^{N} -\log y_{c_i}$
Training data $T = \{(x_1, c_1), \ldots, (x_N, c_N)\}$

- Classification network $y = f(x, w)$

- Evaluate one example $(x_k, c_k)$ (like adding another layer)

\[
\sum_{k=1}^{N} - \log y(x_k, w)_{c_k}
\]

- Evaluation function: $g(T, w) = \sum_{k=1}^{N} - \log y(x_k, w)_{c_k}$

- Solve $\min_w g(T, w)$
Tricks

• **Stochastic Gradient Descent**
  - Computation of \( g(T, w) = \sum_{k=1}^{N} -\log y(x_k, w)_{c_k} \)
  - Requires going through all examples (all N).
  - If N is large and/or if computing \( y(x_k, w) \) is time-consuming, use stochastic gradient descent, i.e. update parameters using subsets of training data.

• **Jittering** - construct a larger training set by perturbing the examples, jittering, translating images, rotating images, warping, mirroring, adding noise, …’

• **Dropout** – in each computation of \( y(x_k, w) \) let a random subset of the neurons die, i.e. set the output to zero.
Thoughts

• Modelling. It takes time to
  – Figure out an appropriate network structure
  – Gather data and ground truth
• The optimization does not always work
  – Parameters explode
  – Nothing happens
• Visualization of the features is important for understanding.
• Feedback in networks
Example: Prostate cancer
Data
Example: Prostate cancer
Data
Result, Cross-validation
Example: Prostate cancer Training
Example: Prostate cancer
Results: Confusion matrix

\[
\begin{bmatrix}
51 & 0 & 0 & 0 & 0 \\
3 & 46 & 3 & 1 \\
0 & 6 & 43 & 0 \\
0 & 3 & 0 & 52
\end{bmatrix}
\]
Visualisation
Visualisation

- Förest kanalen i originalbild
- Resultatet efter det aktuella filter
- Gradient i första kanalen
- Original plus gradienten
Training Deep Learning

• Data
  – Obtain data,
  – cut-outs of right size,
  – jittering,
  – Data expansion (translation, rotation, scaling, mirroring, adding noise, ...)

• Data
  – Obtain ground truth
  – How should the problem be coded
Training Deep Learning

• Hyperparameters
  – How many layers
  – Size of convolution kernels
  – Number of channels
  – Order of layers

• Training parameters
  – Initializing weights
  – Momentum
  – ...

Non-linear function

- Different choices of non-linear functions.
  \[ f(x) = \max(0, x) \]
  \[ f(x) = \ln(1 + e^x) \]

- Faster learning

- Rectifier …
  \[ f(x) = \begin{cases} 
  x & \text{if } x > 0 \\
  0.01x & \text{otherwise} 
  \end{cases} \]
  \[ f(x) = (1 + e^{-x})^{-1} \]
  \[ f(x) = \tanh(x) \]

- … currently most popular, faster training

- Arguments
Training Deep Learning

• Once all of these are in place, there are several good systems for optimizing the parameters
  – MatConvNet, TensorFlow, Caffe, Torch7, Theano
  – Can train on single CPU
  – Faster if compiled for GPU
  – Even faster on cluster of computers with multiple GPU (e.g. LUNARC, http://www.lunarc.lu.se)

• More links on home page for PhD course
• http://www.control.lth.se/Education/DoctorateProgram/deep-learning-study-circle.html
Contents

• Visualization, dimensionality reduction
  – The many applications of SVD
  – Multi-dimensional scaling
  – ISOMAP – non-linear dimensionality reduction
• Deep Learning for classification (review)
• Autoencoders, Deep Learning for dimensionality reduction and visualization
• Segmentation, U-net, other nets
Autoencoders

- Auto: Greek auto- “self, one’s own”
- Image – x, code - h
- Find useful encoder h = f(x) and decoder x = r(h)
- Construct a deep learning network for f and another for r.
- Instead of having network that produce a classification result, (image -> class), the network xhat = r(f(x), is a network that produce an image (image -> image)
- New block type: Upsampling or inverse convolution
- Loss is some measure L(xhat,x) that compares input x with reconstrction xhat
Autoencoders
Autoencoders
Example of uses of autoencoders

• Dimensionality reduction
• Feature learning
• Unsupervised pre-training
• Manifold learning
Prevent trivial coding $x = f(x)$

Different techniques of preventing the autoencoder from learning the identity function

$x = r(h) = r(f(x)) = I(x)$

- Undercomplete autoencoder
- Denoising autoencoder
- Regularized autoencoder
- Contractive autoencoder

Note that it is easy to obtain training data!
Prevent trivial coding $x = f(x)$
h has smaller dimension than $x$

- Constrain $h$ to have smaller dimension than $x$.
- Hope that $h$ will be a useful representation of $x$.
- Used for, e.g., nonlinear dimensionality reduction.
Autoencoders – non-linear dimensionality reduction

Nonlinear dimensionality reduction

Fredrik Bagge Carlson, Lund University: Autoencoders

Corrupt the input data, to make the encoder noise resilient

Corrupt the data or the activations $h$ with random (possibly structured) noise.

Compliated data manifold

Reconstruction function
Segmentation. Also deep learning net of type image -> image
Segmentation
Segmentation, inception module

Fig. 4.2 The inception module as introduced in [39]. Convolutions with kernels of size 1x1, 3x3 and 5x5 are performed in parallel where the resulting feature maps are concatenated to form the input to the next layer. A max pooling layer is also added to keep the beneficial effects of such layers as observed in previous convolutional architectures. The pooling layer does however have a stride of one which results in feature maps with the same spatial resolution as the input. Also note that additional convolutional layers with 1x1 kernels with a single feature map as output is included in the module. These are present to drastically reduce the number of parameters needed to perform the convolutional operations with larger kernels. Should these be omitted the number of parameters and consequently the training time and computational cost would quickly explode for deeper networks.

4.4 ResNet

The chase for deeper networks did not stop with VGG19 or GoogLeNet however. The authors of [16] experimented with even deeper architectures and further increased the state-of-the-art on the ILSVRC-2015 challenge with an ensemble of convolutional networks consisting of as many as hundreds of layers each. [16] found in their initial experiments that when deeper networks start to converge, a process called degradation occurs, which means that when they start to converge, the accuracy gets saturated and then drastically drops. This phenomenon has been concluded to be unrelated to overfitting and is thus a new obstacle to overcome when one utilizes extremely deep architectures with 30 or more layers. Unconstrained, this means that the deeper networks yields a lower performance compared to their shallower counterparts. To address the degradation problem, [16] introduced the so-called residual building block, see Figure 4.3.
Fig. 4.3 Let $H(x)$ denote the underlying desired mapping given the input variable $x$. Instead of optimizing for $H(x)$ directly, we let the weighted layers fit the another mapping: $F(x) = H(x)$. We then recast the mapping to the residual form of: $F(x) + x$. The hypothesis in [16] was that this is an easier formulation of the optimization problem which will lead to a more stable training of extremely deep convolutional neural networks.

The residual building block achieved its purpose in that it mitigates the degradation process and allowed He et al. [16] to construct extremely deep networks that outperformed the shallower networks of the past. Their submission to the ILSVRC-2015 consisted of network with an increase in depth by a factor 8 as compared to the VGG networks. While the depth of the networks employed in the research connected to this thesis is not of the same scale as in [16], the residual building block as been incorporated to increase stability and performance. Should the networks benefit from the addition of the residual building block it will serve to validate their use in shallower versions of deep neural networks.

4.5 Atrous Convolutions

As has been previously mentioned, the main hurdle to overcome when one adapts classification network architectures to semantic segmentation is to combat the decrease in resolution that comes with the pooling operations. The pooling layers play a crucial role in object detection and registration and is vital for location invariance. In addition to these benefits, they also increase the receptive field of the kernels in the deeper parts of the network which is essential for understanding the input, which makes them indispensable. This is true for classification and segmentation tasks both, even though they result in a drastically reduced spatial resolution. In semantic segmentation one wishes to perform pixel wise predictions,
Segmentation

Fig. 6.1 A random sample from the validation set which shows class zero – the patients left scapula – v.s. the background. The scapula is deemed to be in the middle range when it comes to the complexity of the problem due to its size and shape, as compared to the other body parts. Results gained on this class can not be transferred directly to other classes/body parts but provides a good way to compare the different algorithms and gives a rough indication of the quality of the segmentation of the other classes.

For segmentation tasks it is common to use some form of cross entropy as the loss function when training the neural networks. In the problem described in this thesis however, these functions provided a segmentation which was significantly inferior to other tested functions. The problem contains two classes, where the background is extremely dominant.
Segmentation, U-net

Fig. 6.3 Visualization of the U-net model which has been employed in this thesis. Notice that in addition to the standard U-net structure, batch normalization and dropout layers have been added to correspondingly make the model more robust and decrease overfitting.
Segmentation, URI-net

6.3 Atrous Convolutions

Fig. 6.4 A graphical illustration of the modified U-net structure named URI-net. The main modifications that distinguish the model from the standard U-net architecture is the replacement of the standard convolutional layers with inception modules \[39\] and the introduction of residual blocks \[16\] instead of standard skip connections. Other modifications are the use of ELU \[7\] instead of ReLU, convolution with a stride of two instead of sub-sampling by max-pooling and a general decrease in the number of feature maps.
Fig. 6.5 15 different URI-nets have been merged to create a model which is able to perform a segmentation on whole input images where each individual class is identified, located and given a pixel wise segmentation. The model is made up of 15 different networks which have been trained on one of each of the 15 different classes. Each network contains about 20 million parameters which adds up to a total of circa 300 million parameters for the entire, multi-class URI-net model. The constituent networks perform their segmentations independently and their resulting prediction maps are concatenated in a row ⇤ column ⇤ N volume where N represents the number of classes which is 15 in this case. The index of the network which has given the highest predictive value for a certain pixel is chosen as that pixel's class, i.e. an argmax operation is performed on the output volume of the 15 different networks to perform the final segmentation. The multi class model is denoted as a mcURI-net.
Fig. 7.1 An example segmentation with mcURI-net on a randomly selected image from the validation set. The leftmost image shows the ground truth, i.e. the labels which have been produced by traditional image registration using morphology. The image in the middle is the resulting prediction as produced by the mcURI-net where the mean dice score for the validation set was 0.9517. The rightmost image shows the input image and an overlay of the corresponding predictions as constructed by the mcURI-net. The color of each pixel does of course indicate its class.
Semantic Segmentation
Inverse Semantic Segmentation
From segmentation to raw data

Photographic Image Synthesis with Cascaded Refinement Networks
Qifeng Chen and Vladlen Koltun
International Conference on Computer Vision (ICCV)
It might look goofy ...
Deep learning - summary

• What is deep learning
• Supervised vs unsupervised learning
• Goal function, energy function E
• Choice of non-linearity ReLU
• Optimization Back-propagation, SGD
• Tricks dropout
• Examples from speech and vision
• Software
• References