Statistical Machine Learning

Christian Walder

Machine Learning Research Group
CSIRO Data61

and

College of Engineering and Computer Science
The Australian National University

Canberra
Semester One, 2020.

(Many figures from C. M. Bishop, "Pattern Recognition and Machine Learning")

Outlines

Part X

Neural Network 3

Number of layers

- expression of a function is compact when it has few computational elements, i.e. few degrees of freedom that need to be tuned by learning
- for a fixed number of training examples, expect that compact representations of the target function would yield better generalization

- Example representations
  - affine operations, sigmoid ⇒ logistic regression has depth 1, fixed number of units (a.k.a. neurons)
  - fixed kernel, affine operations ⇒ kernel machine (e.g. SVM) has two levels, with as many units as data points
  - stacked neural network of multiple “linear transformation followed by a non-linearity” ⇒ deep neural network has arbitrary depth with arbitrary number of units per layer

Easier to represent with more layers

An old result:

- functions that can be compactly represented by a depth k architecture might require an exponential number of computational elements to be represented by a depth $k-1$ architecture
- Example, the d bit parity function

$$\text{parity} : (b_1, \dots, b_d) \in \{0,1\}^d \mapsto \begin{cases} 1 & \text{if } \sum_{i=1}^d b_i \text{ is even} \\ 0 & \text{otherwise} \end{cases}$$

- Theorem: d-bit parity circuits of depth 2 have exponential size

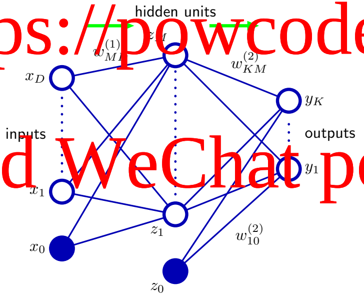
Analogous in modern deep learning:

- “Shallow networks require exponentially more parameters for the same number of modes” — Canadian deep learning mafia.

Recall: Multi-layer Neural Network Architecture

$$y_k(\mathbf{x}, \mathbf{w}) = g\left(\sum_{j=0}^{M} w_{kj}^{(2)}\, h\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i\right)\right)$$

where $\mathbf{w}$ now contains all weight and bias parameters.

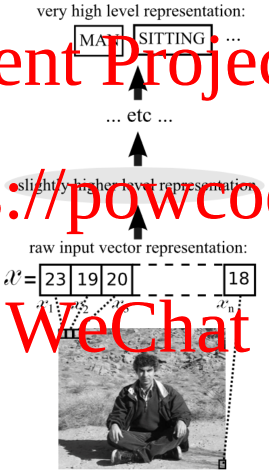


We could add more hidden layers

Empirical observations - pre 2006

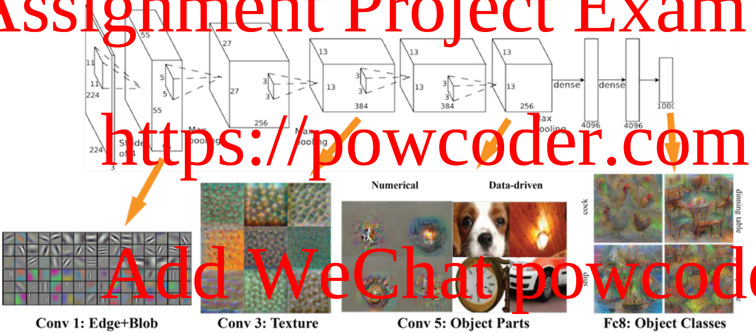
- Deep architectures get stuck in local minima or plateaus
- As architecture gets deeper, more difficult to obtain good generalisation
- Hard to initialise random weights well
- 1 or 2 hidden layers seem to perform better
- 2006: Unsupervised pre-training, find distributed representation

Deep representation - intuition



Bengio, "Learning Deep Architectures for AI", 2009

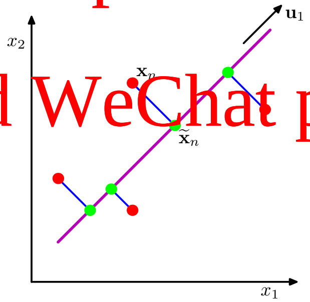
Deep representation - practice



AlexNet / VGG-F network visualized by mNeuron.

Recall: PCA

- Idea: Linearly project the data points onto a lower dimensional subspace such that
  - the variance of the projected data is maximised, or
  - the distortion error from the projection is minimised.
- Both formulation lead to the same result.
- Need to find the lower dimensional subspace, called the principal subspace.



Multiple PCA layers? - linear transforms

- Principle Component Analysis is a linear transformation (because it is a projection)
- The composite of two linear transformations is linear
- Linear transformations $M : \mathbb{R}^m \to \mathbb{R}^n$ are matrices
- Let S and T be matrices of appropriate dimension such that ST is defined

$$ST(X + X') = ST(X) + ST(X')$$

- Similarly for multiplication with a scalar

$\Rightarrow$ multiple PCA layers pointless

Multiple PCA layers? - projection

Ong & Walder & Webers
Data61 | CSIRO
The Australian National University

- Let $X^T X = U \Lambda U^T$ be the eigenvalue decomposition of the covariance matrix (what is assumed about the mean?).
- Define U_k to be the matrix of the first k columns of U, for the k largest eigenvalues. Define Λ_k similarly
- Consider the features formed by projecting onto the principal components

$$Z = XU_k$$

- We perform PCA a second time, $Z^T Z = V \Lambda_Z V^T$.
- By the definition of Z and $X^T X$, and the orthogonality of U

$$Z^T Z = (XU_k)^T (XU_k) = U_k^T X^T X U_k = U_k^T U \Lambda U^T U_k = \Lambda_k$$

- Hence $\Lambda_Z = \Lambda_k$ and V is the identity, therefore the second PCA has no effect

$\Rightarrow$ again, multiple PCA layers pointless

Autoencoder

- An autoencoder is trained to encode the input x into some representation $c(x)$ so that the input can be reconstructed from that representation
- the target output of the autoencoder is the autoencoder input itself
- With one linear hidden layer and the mean squared error criterion, the k hidden units learn to project the input in the span of the first k principal components of the data
- If the hidden layer is nonlinear, the autoencoder behaves differently from PCA, with the ability to capture multimodal aspects of the input distribution
- Let f be the decoder. We want to minimise the reconstruction error

$$\sum_{n=1}^{N} \ell\left(x_n, f(c(x_n))\right)$$

Cost function

- Recall: $f(c(x))$ is the reconstruction produced by the network
- Minimisation of the negative log likelihood of the reconstruction, given the encoding $c(x)$

$$\text{RE} = -\log P(x|c(x))$$

- If $x|c(x)$ is Gaussian, we recover the familiar squared error
- If the inputs x_i are either binary or considered to be binomial probabilities, then the loss function would be the cross entropy

$$-\log P(x|c(x)) = -x_i \log f_i(c(x)) + (1 - x_i)\log(1 - f_i(c(x)))$$

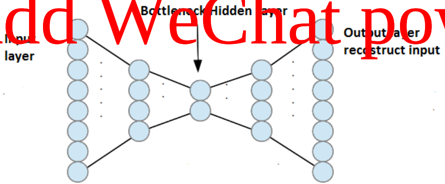
where $f_i(\cdot)$ is the i^{th} component of the decoder

Undercomplete Autoencoder

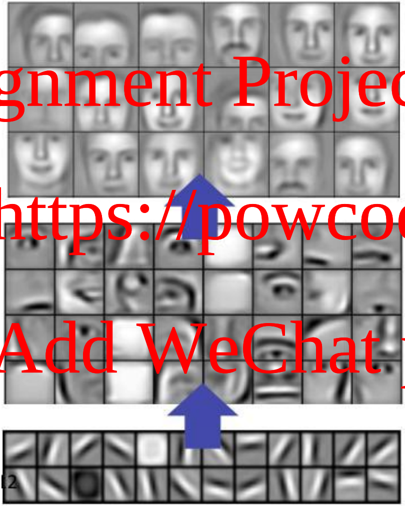
- Consider a small number of hidden units.
- $c(x)$ is viewed as a lossy compression of x
- Cannot have small loss for all x, so focus on training examples
- Hope code $c(x)$ is a distributed representation that captures the main factors of variation in the data

Stacking autoencoders

- Let c_j and f_j be the encoder and corresponding decoder of the j^{th} layer
- Let z_j be the representation after the encoder c_j
- We can define multiple layers of autoencoders recursively.
- For example, let $z_1 = c_1(x)$, and $z_2 = c_2(z_1)$, the corresponding decoding is given by $f_1(f_2(z_2))$
- Because of non-linear activation functions, the latent feature z_2 can capture more complex patterns than z_1.



Higher level image features - faces



Pre-training supervised neural networks

- Latent features z_j in layer j can capture high level patterns

$$z_j = c_j(c_{j-1}(\cdots c_2(c_1(x))\cdots))$$

- These features may also be useful for supervised learning tasks.
- In contrast to the feed forward network, the features z_j are constructed in an unsupervised fashion.
- Discard the decoding layers, and directly use z_j with a supervised training method, such as logistic regression.
- Various such pre-trained networks are available on-line, e.g VGG-19.

Xavier Initialisation / ReLU

- Layer-wise unsupervised pre-training helps by extracting useful features for subsequent supervised backprop.
- Pre-training also avoids saturation (large magnitude arguments to, say, sigmoidal units).
- Simpler Xavier initialization can also avoid saturation.
- Let the inputs $x_i \sim \mathcal{N}(0,1)$, weights $w_i \sim \mathcal{N}(0,\sigma^2)$ and activation $z = \sum_{i=1}^m x_i w_i$. Then:

$$\text{VAR}[z] = \mathbb{E}[(z - \mathbb{E}[z])^2] = \mathbb{E}[z^2] = \mathbb{E}[(\sum_{i=1}^m x_i w_i)^2]$$

$$= \sum_{i=1}^m \mathbb{E}[(x_i w_i)^2] = \sum_{i=1}^m \mathbb{E}[x_i^2]\mathbb{E}[w_i^2] = m\sigma^2.$$

- So we set $\sigma = 1/\sqrt{m}$ to have “nice” activations.
- Glorot initialization takes care to have nice back-propagated signals — see the auto-encoder lab.
- ReLU activations $h(x) = \max(x, 0)$ also help in practice.

Ong & Walder & Webers
Data61 | CSIRO
The Australian National University

Higher dimensional hidden layer

- if there is no other constraint, then an autoencoder with d-dimensional input and an encoding of dimension at least d could potentially just learn the identity function

- Avoid by:
  - Regularisation
  - Early stopping of stochastic gradient descent
  - Add noise in the encoding
  - Sparsity constraint on code $c(x)$.

Denoising autoencoder

- Add noise to input, keeping perfect example as output
- Autoencoder tries to:
  1. preserve information of input
  2. undo stochastic corruption process

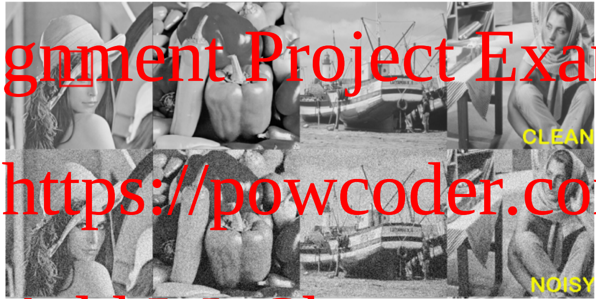
- Reconstruction log likelihood

$$-\log P(x|c(\hat{x}))$$

where x noise free, $\hat{x}$ corrupted

Image denoising

- Images with Gaussian noise added.



- Denoised using Stacked Sparse Denoising Autoencoder

Images from Xie et. al. NIPS 2012

Inpainting - 1

Free a bird

Image from `http://cimg.eu/greycstoration/demonstration.shtml`

Inpainting - 2

Undo text over image

Image from Bach et. al. ICCV tutorial 2009

Recall: Basis functions

- For fixed basis functions $\phi(x)$, we use domain knowledge for encoding features
- Neural networks use data to learn a set of transformations. $\phi_i(x)$ is the i^{th} component of the feature vector, and is learned by the network.
- The transformations $\phi_i(\cdot)$ for a particular dataset may no longer be orthogonal, and furthermore may be minor variations of each other.
- We collect all the transformed features into a matrix Φ.

Sparse representations

- Idea: Have many hidden nodes, but only a few active for a particular code $c(x)$
- Student t prior on codes
- ℓ_1 penalty on coefficients α
  - Given bases in matrix Φ, look for codes by choosing α such that input signal x is reconstructed with low ℓ_2 reconstruction error, while w is sparse

$$\min_{\alpha} \sum_{n=1}^{N} \frac{1}{2} \|x_n - \Phi \alpha_n\|_2^2 + \lambda \|\alpha\|_1$$

  - Φ is overcomplete, no longer orthogonal
  - Sparse $\Rightarrow$ small number of non-zero α_i.
  - Exact recovery under certain conditions (coherence): $\ell_1 \rightarrow \ell_0$.
  - ℓ_1 regulariser $\sim$ Laplace prior $p(\alpha_i) = \frac{\lambda}{2} \exp(-\lambda|\alpha_i|)$.

The image denoising problem

$$\underbrace{y}_{\text{measurements}} = \underbrace{x_{orig}}_{\text{original image}} + \underbrace{w}_{\text{noise}}$$

©2020
Ong & Walder & Webers
Data61 | CSIRO
The Australian National University

Sparsity assumption

- Only have noisy measurements

$$\underbrace{y}_{\text{measurements}} = \underbrace{x_{orig}}_{\text{original image}} + \underbrace{w}_{\text{noise}}$$

- Given $\Phi \in \mathbb{R}^{m \times p}$, find α such that

$$\|\alpha\|_0 \quad \text{is small} \quad \text{for} \quad x \approx \Phi\alpha$$

  where $\|\cdot\|_0$ is the number of non-zero elements of α.
- Φ is not necessarily features constructed from training data.
- Minimise reconstruction error

$$\min_{\alpha} \sum_{n=1}^{N} \frac{1}{2} \|x_n - \Phi\alpha_n\|_2^2 + \lambda\|\alpha\|_0$$

Convex relaxation

- Want to minimise number of components

$$\min_{\alpha} \sum_{n=1}^{N} \frac{1}{2} \|x_n - \Phi \alpha_n\|_2^2 + \lambda \|\alpha\|_0$$

but $\|\cdot\|_0$ is hard to optimise

- Relax to a convex norm

$$\min_{\alpha} \sum_{n=1}^{N} \frac{1}{2} \|x_n - \Phi \alpha_n\|_2^2 + \lambda \|\alpha\|_1$$

where $\|\alpha\|_1 = \sum_n |\alpha_n|$.

- In some settings does minimisation with ℓ_1 regularisation give the same solution as minimisation with ℓ_0 regularisation (exact recovery)?

Mutual coherence

- Expect to be ok when columns of Φ "not too parallel"
- Assume columns of Φ are normalised to unit norm
- Let $K = \Phi\Phi^T$ be the Gram matrix, then $K(i,j)$ is the value of the inner product between ϕ_i and ϕ_j.
- Define the mutual coherence

$$M = M(\Phi) = \max_{i \neq j} |K(i,j)|$$

- If we have an orthogonal basis, then Φ is an orthogonal matrix, hence $K(i,j) = 0$ when $i \neq j$.
- However, if we have very similar columns, then $M \approx 1$.

Ong & Walder & Webers
Data61 | CSIRO
The Australian National University

Exact recovery conditions

- If a minimiser of the true ℓ_0 problem, α^* satisfies

$$\|\alpha^*\|_0 < \frac{1}{M}$$

then it is the unique sparsest solution.

- If α^* satisfies the stronger condition

$$\|\alpha^*\|_0 < \frac{1}{2}\left(1 + \frac{1}{M}\right)$$

then the minimiser of the ℓ_1 relaxation has the same sparsity pattern as α^*.

References

- Yoshua Bengio, "Learning Deep Architectures for AI", Foundations and Trends in Machine Learning, 2009
- http://deeplearning.net/tutorial/
- http://www.deeplearningbook.org/contents/autoencoders.html
- Fuchs, "On Sparse Representations in Arbitrary Redundant Bases", IEEE Trans. Info. Theory, 2004
- Xavier Glorot and Yoshua Bengio, "Understanding the difficulty of training deep feedforward neural networks", 2010.