Generalization capability

Pratical question: generalization capability

After training, how does the FNN will perform on new patterns from a test set?

Generalization

- let us measure this with the performance of a model f on a test set T

f= $\Psi(L)$ is produced by a learning algorithm Ψ using a train set L

UNIVERSITÀ DI SIENA 1240

Pratical question: generalization capability

Interesting questions

- what is the predicted performance of f on test set T?
- This question is related to
  - which model should we choose?
  - which learning algorithm should we choose?
  - When learning should be stop?

Answer 1.0

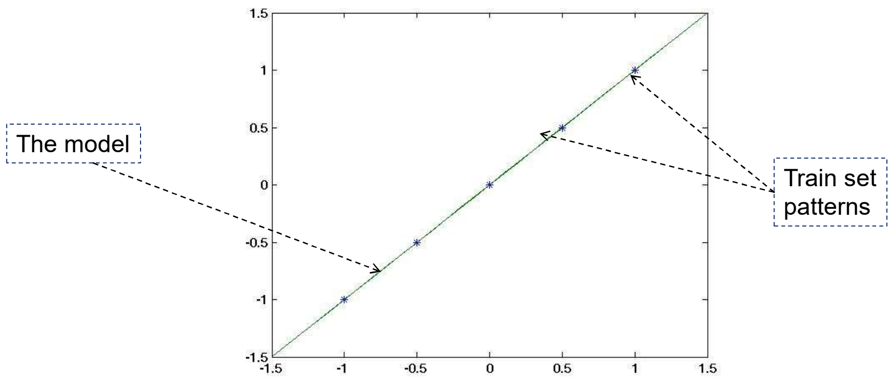
- no answer is possible without assumptions on training algorithm and network architecture

111

UNIVERSITÀ DI SIENA 1240

Measuring generalization capabilty requires assumptions

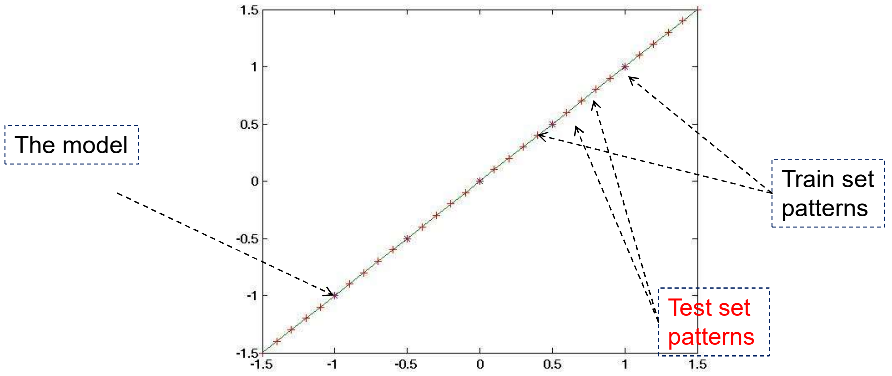
How well will this model generalize on novel patterns?



112

Measuring generalization capabilty requires assumptions

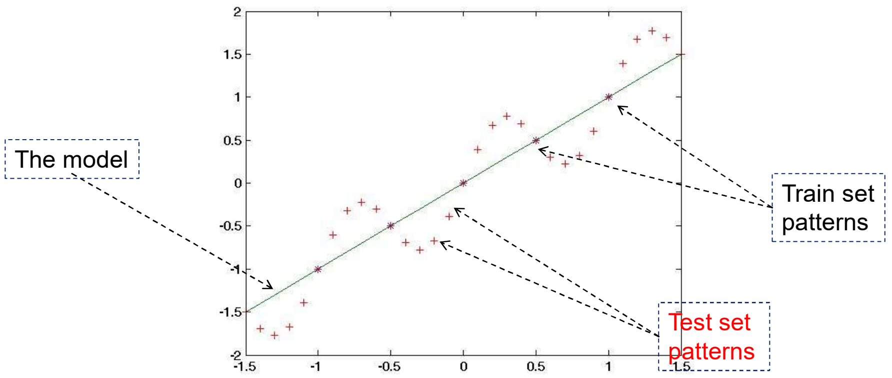
Very well!!!



113

Measuring generalization capabilty requires assumptions

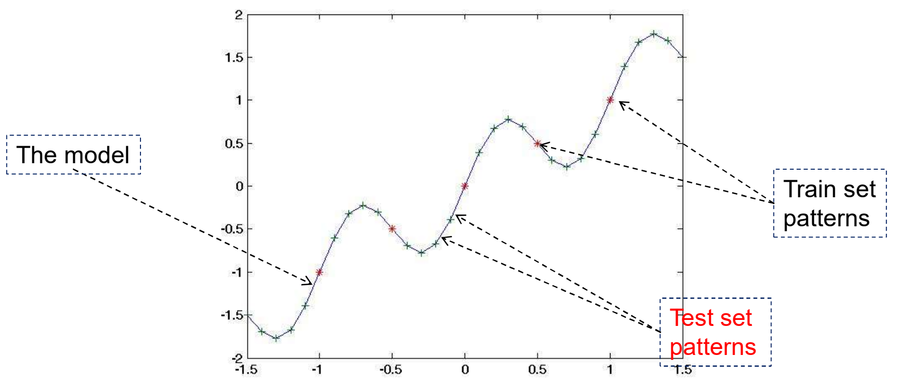
Very bad!!!



114

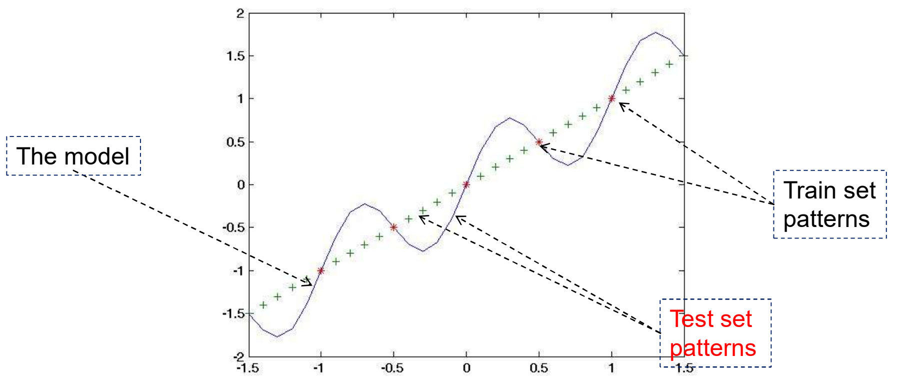
Measuring generalization capabilty requires assumptions

Another model?



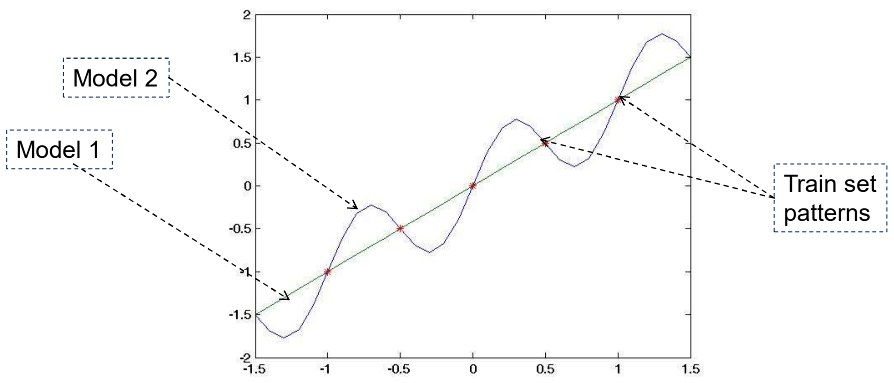
Measuring generalization capabilty requires assumptions

Another model? …. It does not generalize well in this case!



Measuring generalization capabilty requires assumptions

Which model is best to have a good generalization?



117

No free lunch theorem

Intuitive version

- Without constraints on the considered problem, all the learning algorithms may show the same generalization error!!

Formally

- *A training algorithm which produces a model f:X->Y , using a train set*
- X= a fine set of inputs, Y= a finite set of outputs
- *Train set* $L=\{(x_i,t_i) \mid x_i \in X, t_i \in Y\}$
- *Test set* $T=\{x_i \mid x_i \in X\}$
- *The target function t:X->Y*
- *Error on test set* $e_{test} = \sum_{x_i \in T} L(t(x_i), f(x_i))$ *for some error function L*

If the target function t is uniformly sampled, for any learning algorithm *mean(e_{test})* is constant

118

No free lunch theorem

It means that without assumptions

- The learned model which is better on a problem is worse on another:
  - If A1 is better than A2 on certain kind of problems, there must be another kind of problems where A2 is better than A1
  - averaged over all the problems, both algorithms are equally good.
- Even a random learning algorithm performs as well as the other algorithms
- Generalization is not possible without a (often implicit) bias of the algorithm

119

Measuring generalization capabilty requires assumptions

Assumption 1: data distribution

- The data on test set (working environment) are drawn from the same distribution as the data in train set
  - Not obvious in real applications
  - It means that the pair pattern-targets must be drawn from the same distributions

120

UNIVERSITÀ DI SIENA 1240

Measuring generalization capabilty requires assumptions

Occam's razor (law of parsimony):

- the simplest explanation is usually the correct one

Assumption 2:

- the model that produces the best generalization is the simplest (among those that classifies correctly the train set)

Such an assumption is about real life applications!

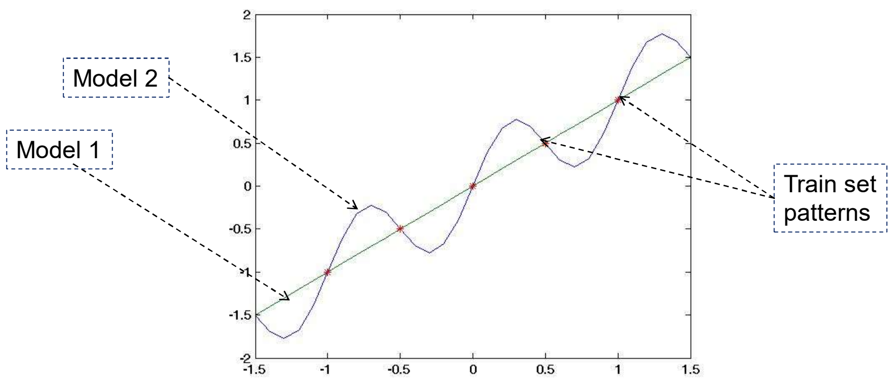
Next step … how do we measure simplicity/complexity?

121

UNIVERSITÀ DI SIENA 1240

Measuring simplicity

Better model 1, which is the simplest



122

UNIVERSITÀ DI SIENA 1240

Measuring simplicity

Intuitive ideas: the complexity of a parametric model depends on

- the number of roots the model can have
- the number of maxima/minima the model can have
- the number of patterns the model can interpolate
- the number of ways a set of patterns can be classified
- …

123

UNIVERSITÀ DI SIENA 1240

Measuring simplicity: Vapnik-Chervonenkis dimension (VCD)

Intuitive definition of “shatter”

- A classifier f_w is said to shatter a set of patterns $x_1,..xk,$ if by changing the parameters we can classify the patterns in any possible way

Formal definition of “shatter”

- for any possible assigmament $t_1,..t_k$, $t_i \in \{0,1\}$, there is a set of parameters w such that

$$f_w(x_i) > 0 \text{ if } t_i=1$$
$$f_w(x_i) < 0 \text{ if } t_i=0$$

124

Examples of shattering

Example 1

- A linear function $f_w(x)= w_1x+w_0$ can shatter the set {1,2}... (and any set of 2 reals)

Example 2

- A polynomial $f_w(x)= w_3x^3+w_2x^2+w_1x+w_0$ can shatter the set {0,1,2,3}... (and any set of 4 reals)

Example 3

- The function $f_w(x)= tanh(w_1x+w_0)$ can shatter the set {1,2}... (and any set of 2 reals)

Example 4

- The function $f_w(x)= sign(sin(w_1x+w_0))$ can shatter any set in *R*!!!

Vapnick-Chervonenkis dimension (VCD)

Intuitive definition

- The VCD dimension of classifier f_w is the dimension of the largest pattern set on which the model can produce any combination of outputs

Formal definition

$$VCD(f_w) = \max_X(|X|), \qquad \text{X is a set shattered by } {}_{\text{fw}}$$

Intuitive ideas about VCD

VCD provides lower bounds on

- the maximum number of roots
  (models with a single input)
- the maximum number of minima/maxima
  (models with a single input)
- the maximum number of regions partitioned by the model
  (models with many inputs)

127

Examples of VCD

Examples

- linear functions $f_w(x)= w_1x+w_0$
  VCD(f_w)=2
- Polynomials of order k, $f_w(x)= w_kx^k+\ldots+w_1x+w_0$
  VCD(f_w)=k+1
- Neural networks with k neurons, single hidden layer, ReLU activation function
  VCD(f_w)=k+1

128

UNIVERSITÀ DI SIENA 1240

VCD of neural networks

Neural networks with *p* parameters, any number of layers, bounds for the order of growth of VCD(f_w)

- FNN with Relu and L layers
  Upper bound: O(pL log p),
  Lower bound o(pL log p/L)
- FNNs with piecewise polynomial activation function
  Upper bound: O(pL^2+pLlog p),
  Lower bound o(pL log p/L)
- FNNs with, tanh, logsig, atan activation function
  Upper bound: O(p^4) … this may be overestimated
  Lower bound o(p^2)

129

UNIVERSITÀ DI SIENA 1240

VCD … the obvious general rule

- For a general neural network model, the smaller number of neurons/parameters/feature/levels, the smaller VCD
- But different architectures cannot be directly compared using the number of neurons/parameters/feature/levels.... since the architecture affect the VCD

130

VCD and test error

Vapnik proved that

- the problem is to learn a binary classifier f_w
- V is VC dimension of f_w
- Mean train error $e_{train} = \frac{1}{N}\sum_{i=1}^{N}|t_i - fw(xi)|$
- Mean test error $e_{test} = \frac{1}{N}\sum_{i=1}^{N2}|\overline{t_i} - fw(\overline{x_i})|$
- $0<\varepsilon<1$
- Train and test patterns (and targets) are drawn by the same distribution

Then

$$P\left(e_{test} \leq e_{train} + \sqrt{\frac{V(\log(\frac{2N}{V}) + 1) - \log(\frac{\varepsilon}{4})}{N}}\right) \geq 1 - \varepsilon$$

131

VCD and test error

Vapnik proved that (Vapnik 1989)

$$P\left(e_{test} \leq e_{train} + \sqrt{\frac{V(\log(\frac{2N}{V}) + 1) - \log(\frac{1-\varepsilon}{4})}{N}}\right) \geq \varepsilon$$

- The larger the number of train samples N, the smaller the generalization error
  - **Overfitting behaviour when training with few examples**

132

UNIVERSITÀ DI SIENA 1240

VCD and test error

Vapnik proved that (Vapnik 1989)

$$P\left(e_{test} \leq e_{train} + \sqrt{\frac{V(\log(\frac{2N}{V}) + 1) - \log(\frac{1-\varepsilon}{4})}{N}}\right) \geq \varepsilon$$

- The larger VD dimension V, the larger generalization error
  - **Complex models produce worse generalization**

133

UNIVERSITÀ DI SIENA 1240

VCD and test error

It has been proved also the converse

- When VCD is large then the generalization probability is large!!
- If VCD is infinitive, then it may be impossible to learn a model with bounded generalization error !!!
  - Can you guess why?

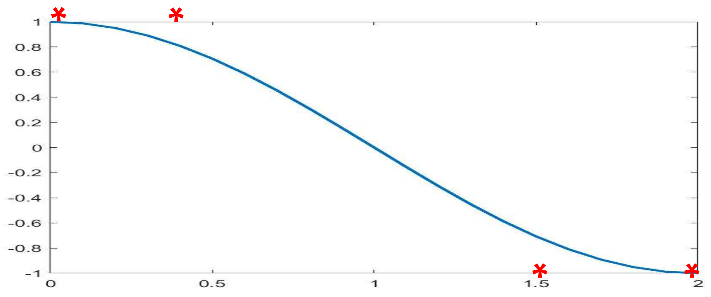
134

UNIVERSITÀ DI SIENA 1240

Infinitive VCD

The function $f_{w,b}(x)$ =*sin(w x+b),*

- *It has infinitive VCD*
- *Suppose, you want to learn a target function t*
  t(x)>0 for x<1 and t(x) for x>0,
  then training with the pattern in the figure you expect to obtain the function f=sin(π x/2+ π /2)



135

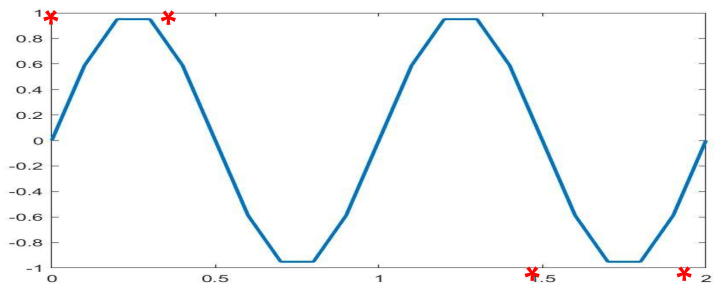
UNIVERSITÀ DI SIENA 1240

Infinitive VCD

But your training algorithm cal also produce

f=sin(4(π x/2+ π /2))

Yes, you can add more training patterns to avoid this, but …



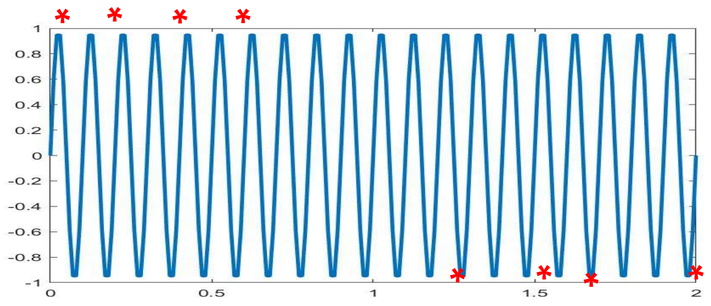
136

UNIVERSITÀ DI SIENA 1240

Infinitive VCD

…. this is a never end story

- In general, using architectures/activation functions with infinitive VCD is not a good idea in machine learning!!
  - or, at leat, we do not know how to manage them!!



137

UNIVERSITÀ DI SIENA 1240

Generalization capablity in practice

In practice

- Suppose that you have a set of architectures that satisfies your purpose from the approximation and learning point of view and you want to chose the one that give you the best generalization, what to do?
- The VCD allows to estimate the error on test, but the VCD cannot be used in practice, since the bounds are too raw

Alternatives for choosing architectures, algorithms, ….

- Predict the perfomance on test set (by validation)
- Keeping weight small
  - This includes Support Vector Machine …. not a matter of this course
- Pooling

138

Validation by random subsampling

Let *D* be the available dataset

1. Divide *D* randomly into a train T and a validation subset V
2. Train the model on T
3. Evaluate the model on the validation set N
4. Repeat *k* times
5. Calculate the average error rate

139

Validation k-fold cross validation

Let *D* be the available dataset

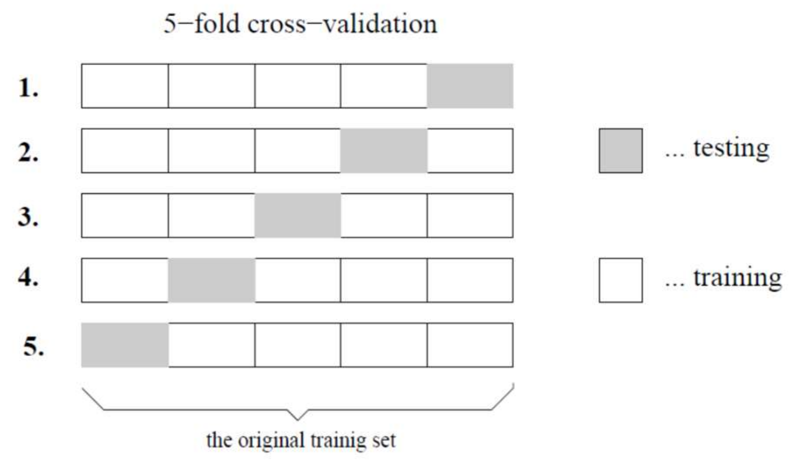
1. Divide *D* randomly into a train T1,..,Tk subsets
2. For i=1 to k

   Train the model on all the set except T_i

   Evaluate the model on T_i
3. Calculate the average error rate

140

Validation k-fold cross validation



141

Validation: why does it work?

Why does it work?

- Validation just allows to experimentally predict the error on test set
- We must assume that validation set is drawn from the same distribution of test and train set

142

Validation: what is it useful for ?

- To compare different models (neural networks, Bayesian models, …)
- To compare different architectures (number of layers, number of neurons, …)
- Decide when to stop learning
- ….

143

The role of weight sizes in neural networks

Does weight size affect generalization?

- network with small weights produce smooth function
- smooth functions look simpler than non-smooth ones

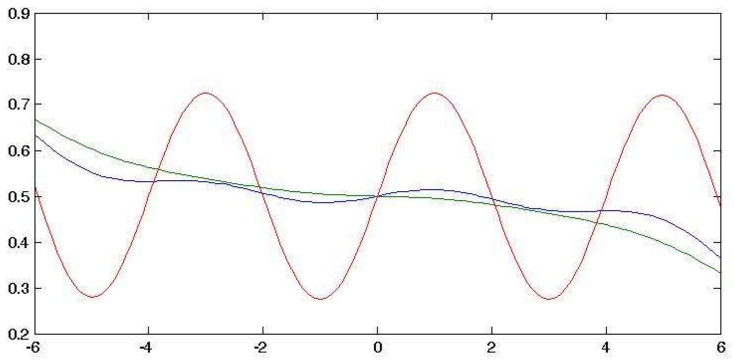
Notice

- Networks with small weights are universal approximators ….
  - Can you prove this?

144

The role of weight sizes in neural networks

A neural network a single layer and different weights sizes (w=0.3, 0.4, 0.5)



145

VCD and weight size

An extended version of VC dimension (Bartlet)

- Usual mean test error $e_{test} = \frac{1}{N}\sum_{i=1}^{N2} |\overline{t}_i - fw(\overline{x}_i)|$
- Error on train set
  (patterns are shattered with a margin γ)
  $e_{train}^{\gamma} = \frac{1}{N}\sum_{i=1}^{N} d_{\gamma}(t_i, f_w(xi))$,
  $d_{\gamma}(ti, f_w(xi))$ =1 if $t_i = 1$ and $f_{w(x_i)} \geq \gamma$, and $d_{\gamma}(ti, f_w(xi))$=0 otherwise
- $d_{\gamma}(ti, f_w(xi))$ =1 if $t_i = 0$ and $f_{w(x_i)} \leq -\gamma$, and $d_{\gamma}(ti, f_w(xi))$=0 otherwise

- $V(\gamma)$ fat-shattering dimension
  - $V(\gamma)$ i is the maximum dimension of a set shattered with error smaller than γ

146

UNIVERSITÀ DI SIENA 1240

VCD and weight size

An approximated version of VC dimension (Bartlet)

$$P\left(e_{test} \leq e_{train}^{\gamma} + \sqrt{2\,\frac{V(\gamma/16)\,\ln\left(\frac{34eN}{V(\gamma/16)}\right)\log(578N) + \ln\left(\frac{4}{1-\varepsilon}\right)}{N}}\right) \leq \varepsilon$$

- The bound has similar properties w.r.t those of VC,
  - The larger $V(\gamma)$ the worse the generalization
- even if
  - e_{train}^{γ} is larger than e_{train}
  - $V(\gamma)$ is larger than V

147

UNIVERSITÀ DI SIENA 1240

VCD and weight size

Bartlet proved that

- Neural network with sigmoidal activation function, having output in [-M/2,M/2] and module of derivative smaller than β
- Module of input smaller than B
- L layers
- **Weights bounded by α**

Weight dimension

$$V(\ \ ) \leq \frac{4B^2}{\gamma^2}\left(\frac{M}{\gamma}\right)^{2(L-1)} (2\alpha\beta)^{L(L+1)} log\left(3^{L-1}(L-1)!\,(2n-1)\right)r^2$$

- **The smaller the weights, the smaller the fat-shattering dimension V(γ) !!**

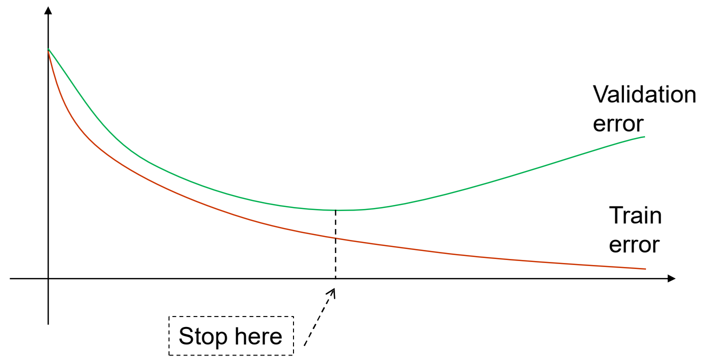
148

UNIVERSITÀ DI SIENA 1240

Keeping weight small

Early stopping

- Initialize the weights to small values
- Train the network
  - stop when error on validation increases



149

UNIVERSITÀ DI SIENA 1240

Keeping weight small

Penalty (weight decay)

- Add penalty on a weight to error

$$e_{train} = \frac{1}{N}\sum_{i=1}^{N}(t_i - fw(xi)) + \lambda p(w)$$

where

$$p(w) = \sum_i w_i{}^2$$

Notice that

$$\frac{\partial p(w)}{\partial w_i} = -2wi$$

Weight decay

150

Keeping weight small

Constraint on neuron weights

- For each neuron k, activate a constraint when the input weight is larger than a given maximum

$$p_k(w) = \begin{cases} \sum_i {w_{ik}}^2 - M & if \sum_i {w_{ik}}^2 > M \\ 0 & otherwise \end{cases}$$

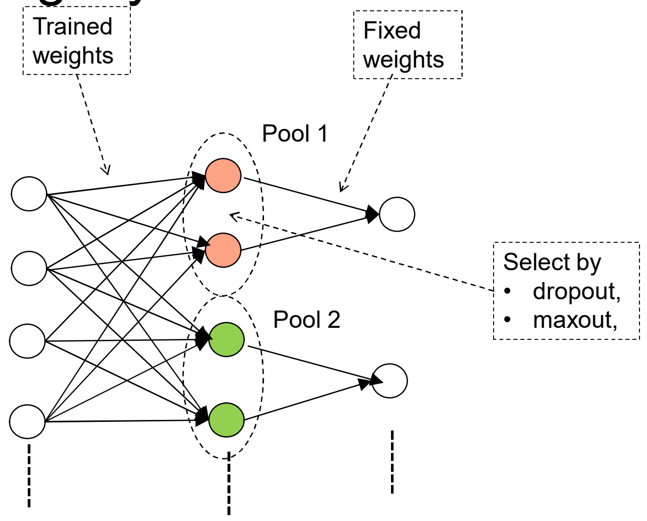
151

Pooling layers

- Neurons of a layer are grouped in subset
- For each pattern, only a fraction of the neurons in a group are activated
  - The output of the other neurons is not considered
- The active neurons an a group are selected by
  - Taking the max (maxout)
  - Taking a random set (dropout)

152

Pooling layers



153

Pooling layers

- Pooling layers reduce the number of parameters and neurons
- Pooling layers reduce VC dimension

154

UNIVERSITÀ DI SIENA 1240

Other explanations

- Early stopping helps because it predicts the test set performance
- Weight decay disactivate some of the weights
- Pooling removes similar features
- …

155

UNIVERSITÀ DI SIENA 1240

Putting everything together: approximation, learning, and generalization, the global picture

You have a set of architectures, you want to chose the bestone from all point of views: approximation, learning and generalization

Antagonist goals

- Larger models improve approximation and learning, but they decrease generalization
- Larger weights improve approximation, but they decrease generalization
- Larger trainset improve generalization, but it makes learning more difficult

156

Approximation, learning, generalization: the global picture

	approximation	learning	generalization
Large train set		worse	better
Large weights	better	?	worse
Large model	better	better	worse

157

Other constraints to considered to select the architecture

Constraints from the considered problem

- There is a minimum dimension for the model
  - In a practical application, the approximation (error) cannot be smaller than a minimum
  - Too small models cannot solve the problem as desired
- There is a maximum amount of data available for training and validation
  - Collecting/labelling data is expensive, …
- The computation resources are bounded
  - Dimension of train set and model dimension affect the required computational resources

158

Why generalization may be different from what expected

It is assumed that the patterns of train, validation, and test sets are drawn from the same distribution

- The distribution is different in most of real life applications (performance on test much worse than on validation/train)
- Patterns in test may appear also in training (performance on test better than expected, when a lookup table is used)
- Often there are relationships between patterns
  - Patterns are not independent
  - Using relationships improve performance on test

159

Each problem/architecture is a singleton

- But remember that the generalization/approximation/learning capability depend in complex way on
  - the model architecture
    (the type, the number of weights/neurons, the size of the weights), and also on
  - the problem
    (the type, the number of examples in train set)
- Thus expect that the rules in the previous slides work for networks with the same architecture and a different number of neurons/weights, but they may fail for different types of architectures
- In the following part of the course, we are going to see how the network architecture may play an important role in its properties

160

UNIVERSITÀ DI SIENA 1240

Our analysis of generalization does not include reliability

Reliability

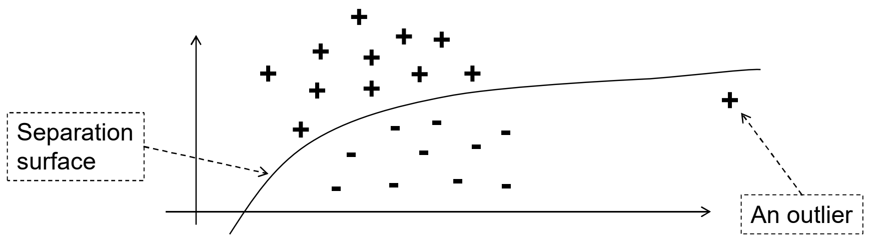
- A measure of how much you can trust the prediction
- Very important in verification problems
- Generalization theory is not of help
  - It tells you how many errors your model will do on the whole test set
  - It does tell you anything about the reliability of the prediction on a single pattern

161

UNIVERSITÀ DI SIENA 1240

Reliability

- A measure of how much you can trust the prediction
- Very important in verification problems
- Common neural networks do not provide a reliability measure
- The prediction may be very unreliable for outliers!
  - Separation surface is unreliable where there are no train patterns



162

Reliability in machine learning

Predictive models (bayesian models, autoencoder)

- Model trainset distribution
- Predict the probability that a pattern is generated from the same distribution as that of training set
- Good to recognize outliers
- Good for verification problems

Discriminative models (common neural networks, SVMs,..)

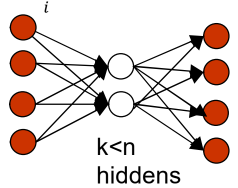
- Do not model trainset distribution
- Predict the most probable class (or targets)
- Good for classification problems

163

Autoencoders

- Input and output layer have the same number of neurons *n*
- One (or many) hidden layer with k<n neurons
- The network is trained to copy input x to output $f_w(x_i)$

$$e_{train} = \sum_i (x_i - fw(xi))^2$$



164

Autoencoders: how they are used

For verificaton

- During trainining, the autoencoder is trained to copy input to output
- During test, we use $(x-f_w(x))^2$ as a measure of the probability that x belong to the train set distribution

For classificaton into k classes

- During trainining, the k-th autoencoder is trained to copy input to output using only positive patterns of each class
  - Eventually error is changed to accommodate negative examples
- During test, it is returned the class of the autoencoder that obtains the smallest errror $(x-f_w(x))^2$

165

Autoencoders: how they are used

For data generation

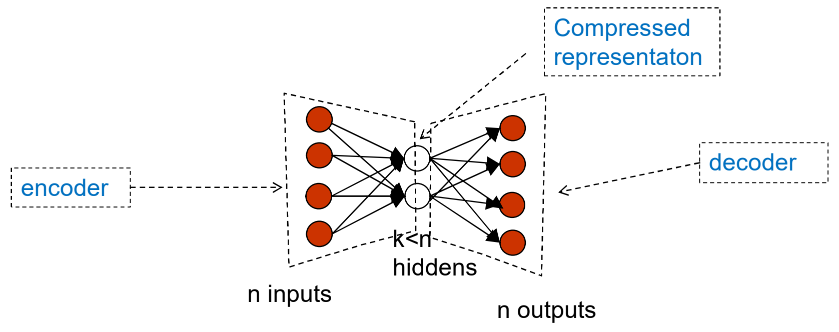
- After training, the decoder is feeded with some data. We expect that it generate a value that resemble those in the train set

166

Autoencoders: how they are used

For feature dimension reduction

- During trainining, the autoencoder is trained to copy input to ourput
- During test, the hidden node output is used as compressed representation of the features

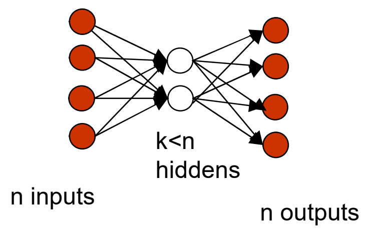


167

Autoencoders: why they work

Why autoencoders work

- The autoencoder cannot approximate the identity function
- The autoencoder are forced to compress the input information into a smaller space
  - The smaller the hidden layers, the larger the compression



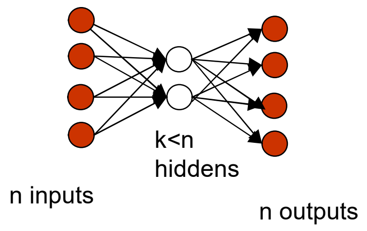
168

Autoencoders: why they work

- The function implemented by an autoencoder is

$$f(x) = W_2\sigma(W_1 x + b_1) + b_2$$

- An autoencoder implements a sort of non linear version of PCA (Principal component analysis)



169

Autoencoders vs common networks

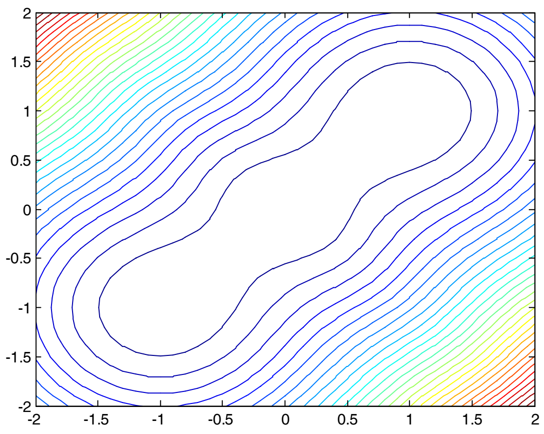
It has been proved that (me … and Gori)

- The separation surfaces for autoencoders are always closed
(provided that the number of hidden neurons is smaller than the number of inputs)
- The separation surface of a FNN can be both open and closed

170

Autoencoders vs common networks

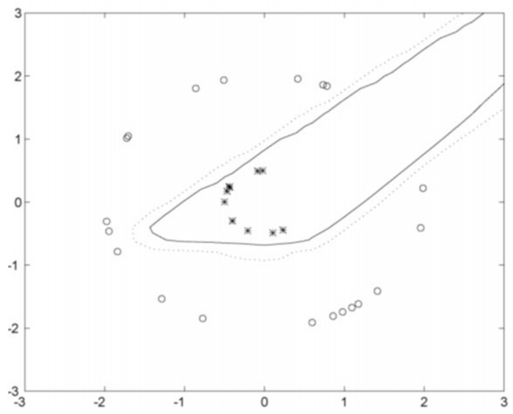
- A contour plot of an autoencoder



171

Autoencoders vs common networks

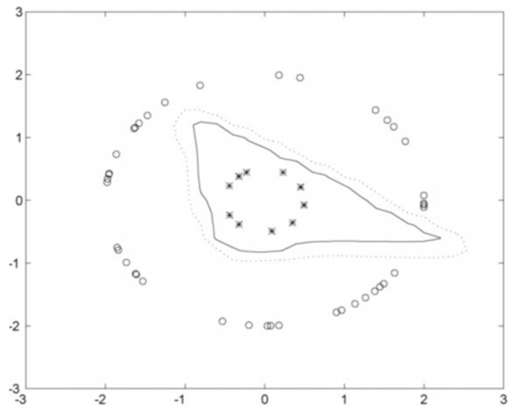
The separation surface of a feedforward network with 5 hiddens



172

Autoencoders vs common networks

The separation surface of a feedforward network with 5 hiddens



173

Autoencoders vs common networks

Intuitively

- Autoencoders are advantageous for verification
  - due to closed surface fact
  - due to the fact that it is a predictve model

174

UNIVERSITÀ DI SIENA 1240

A modern architecture: variational autoencoder

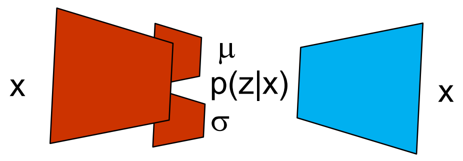
- Old autoencoders work only with one layer encoders and one layer decoders
  - In this way, the encoders and the decoders are not universal approximators
- With more layers,
  - surfaces may be open
  - the identity function from input to output can be implemented even if the hidden layer is smaller then the input
    - All the data is auto associated

175

UNIVERSITÀ DI SIENA 1240

A modern architecture: variational autoencoder

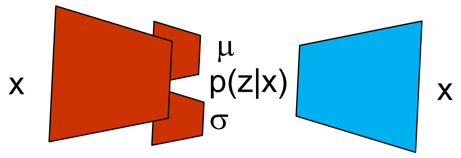
- Variational autoencoders use encoders and decoders with several layers,
- The enconder takes in input x and
  - generates the parameters (mean μ, and variance σ) of the distribution of latent hidden variable $p(z|x)$
  - The decoder takes the variable z and generate x



176

A modern architecture: variational autoencoder

- By assuming that hidden variables *p*(*z*|*x*) have a normal distribution, we limit the freedom of the encoding and the decoding networks.
  - We can use several layers
- Variational autoencoders are used to generate images, etc



177

Recurrent neural networks

178