Neural networks and optimization

Nicolas Le Roux

Criteo

15/1/16

Goal : classification and regression

- Medical imaging : cancer or not ? Classification
- Autonomous driving : optimal wheel position Regression
- Kinect : where are the limbs ? Regression
- OCR : what are the characters ? Classification

Goal : classification and regression

- Medical imaging : cancer or not ? Classification
- Autonomous driving : optimal wheel position Regression
- Kinect : where are the limbs ? Regression
- OCR : what are the characters ? Classification

It also works for structured outputs

Object recognition

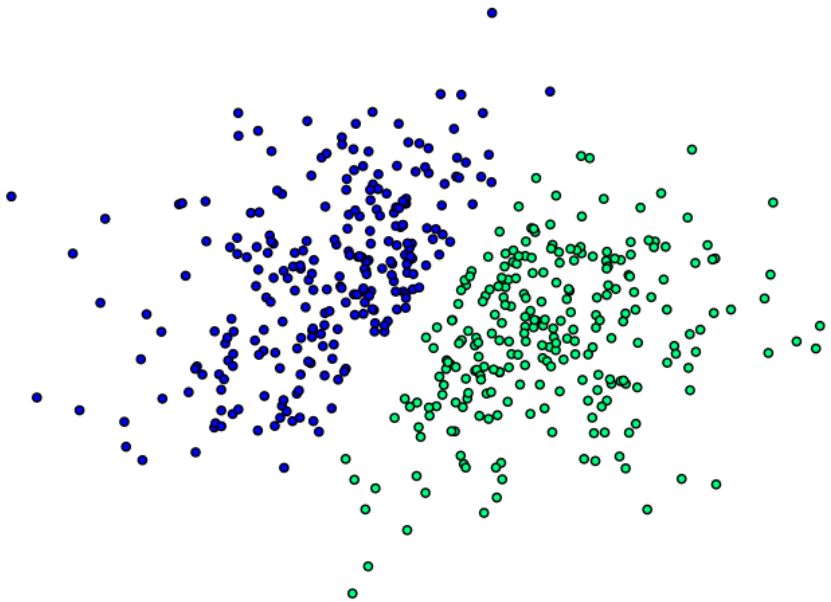
Object recognition

Linear classifier

- Dataset : $\left(X^{(i)}, Y^{(i)}\right)$ pairs, $i = 1, \ldots, N$.
- $X^{(i)} \in \mathbb{R}^n$, $Y^{(i)} \in \{-1, 1\}$.
- Goal : Find w and b such that $\text{sign}(w^\top X^{(i)} + b) = Y^{(i)}$.

Linear classifier

- Dataset : $\left(X^{(i)}, Y^{(i)}\right)$ pairs, $i = 1, \ldots, N$.
- $X^{(i)} \in \mathbb{R}^n$, $Y^{(i)} \in \{-1, 1\}$.
- Goal : Find w and b such that $\text{sign}(w^\top X^{(i)} + b) = Y^{(i)}$.

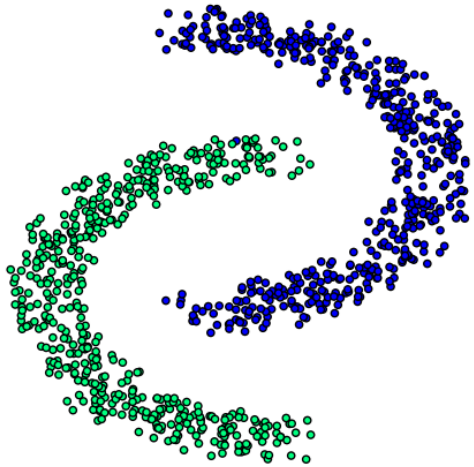


Perceptron algorithm (Rosenblatt, 57)

- $w_0 = 0$, $b_0 = 0$
- $\widehat{Y}^{(i)} = \text{sign}(w^\top X^{(i)} + b)$
- $w_{t+1} \leftarrow w_t + \sum_i \left(Y^{(i)} - \widehat{Y}^{(i)}\right) X^{(i)}$
- $b_{t+1} \leftarrow b_t + \sum_i \left(Y^{(i)} - \widehat{Y}^{(i)}\right)$

Linearly separable demo

Some data are not separable



The Perceptron algorithm is NOT convergent for non linearly separable data.

Non convergence of the perceptron algorithm

Non-linearly separable perceptron

- We need an algorithm which works both on separable and non separable data.

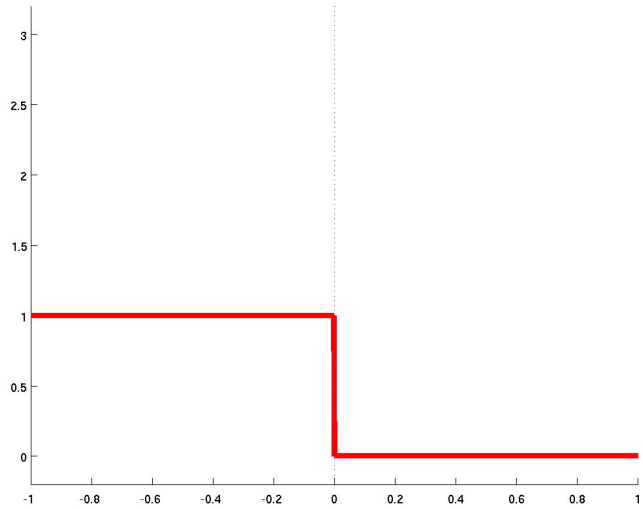
Classification error

A good classifier minimizes

$$\begin{aligned}\ell(w) &= \sum_i \ell_i \\ &= \sum_i \ell\left(\widehat{Y^{(i)}}, Y^{(i)}\right) \\ &= \sum_i 1_{\text{sign}(w^\top X^{(i)}+b)\neq Y^{(i)}}\end{aligned}$$

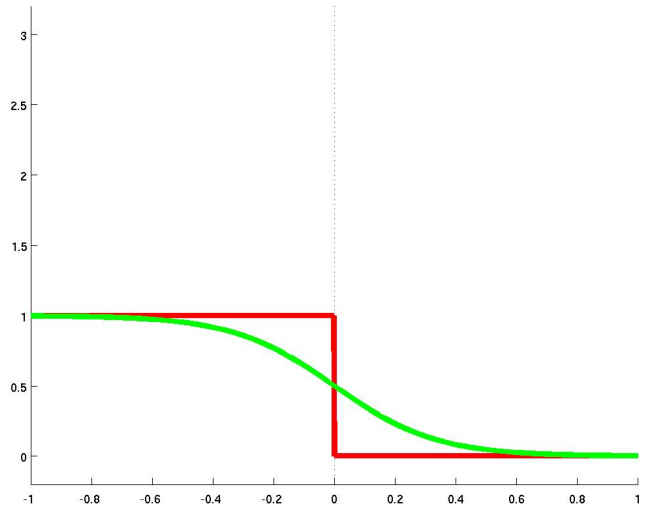
Cost function

- Classification error is not smooth.



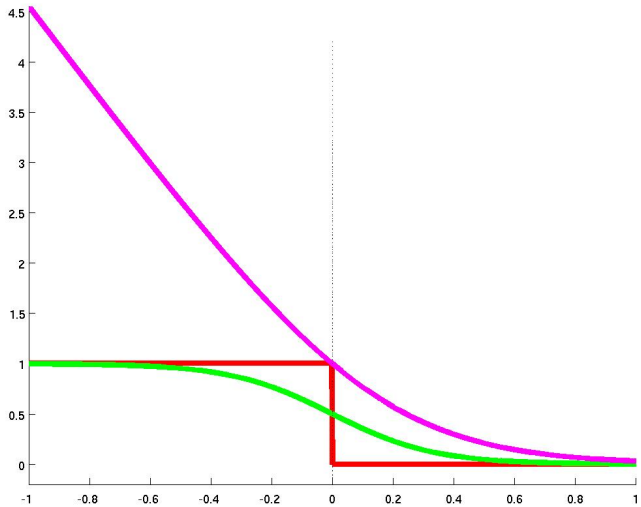
Cost function

- Classification error is not smooth.
- Sigmoid is smooth but not convex.



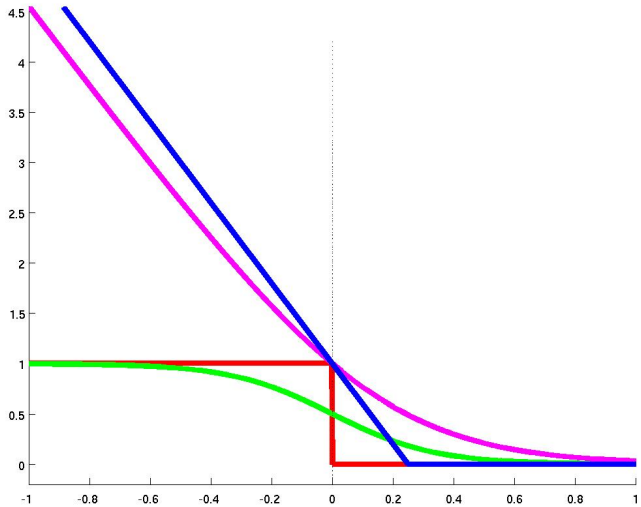
Convex cost functions

- Classification error is not smooth.
- Sigmoid is smooth but not convex.
- Logistic loss is a convex upper bound.



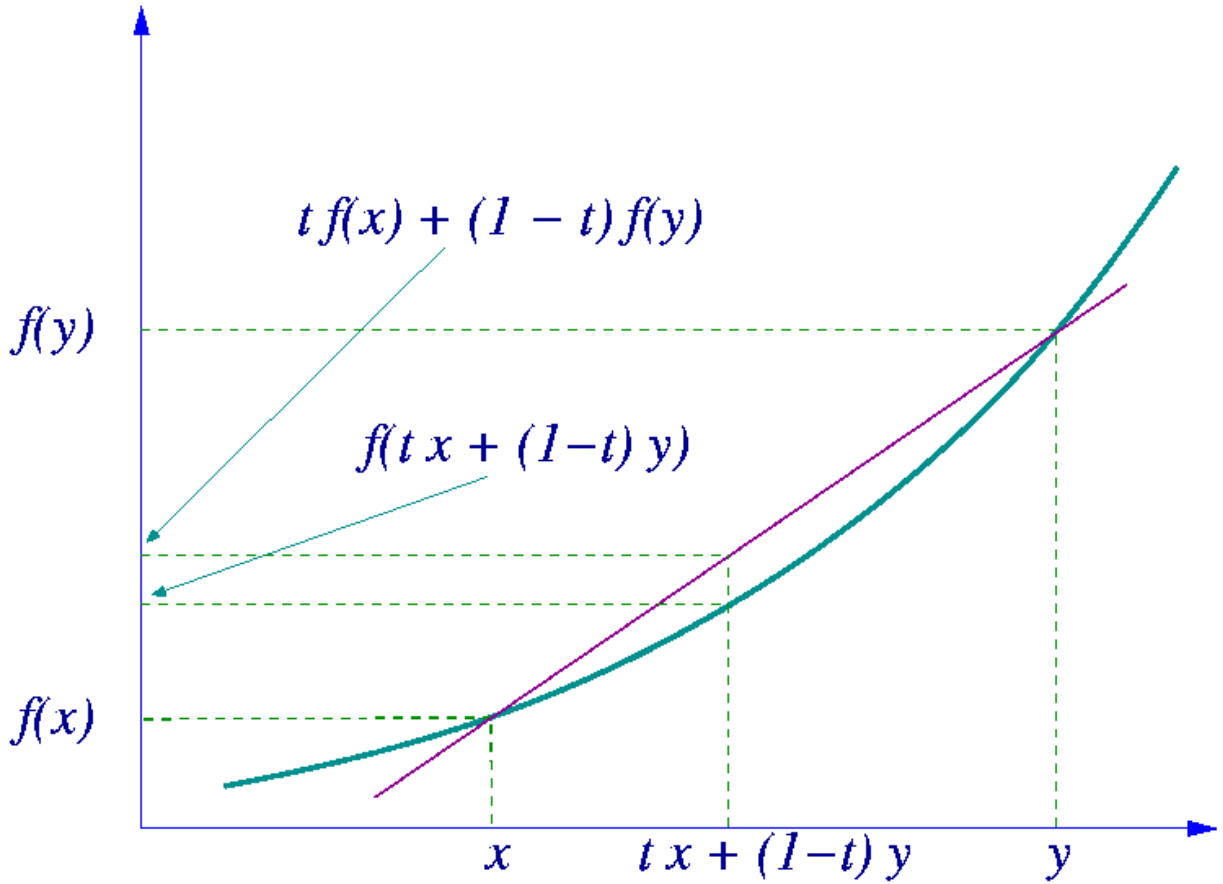
Convex cost functions

- Classification error is not smooth.
- Sigmoid is smooth but not convex.
- Logistic loss is a convex upper bound.
- Hinge loss (SVMs) is very much like logistic.

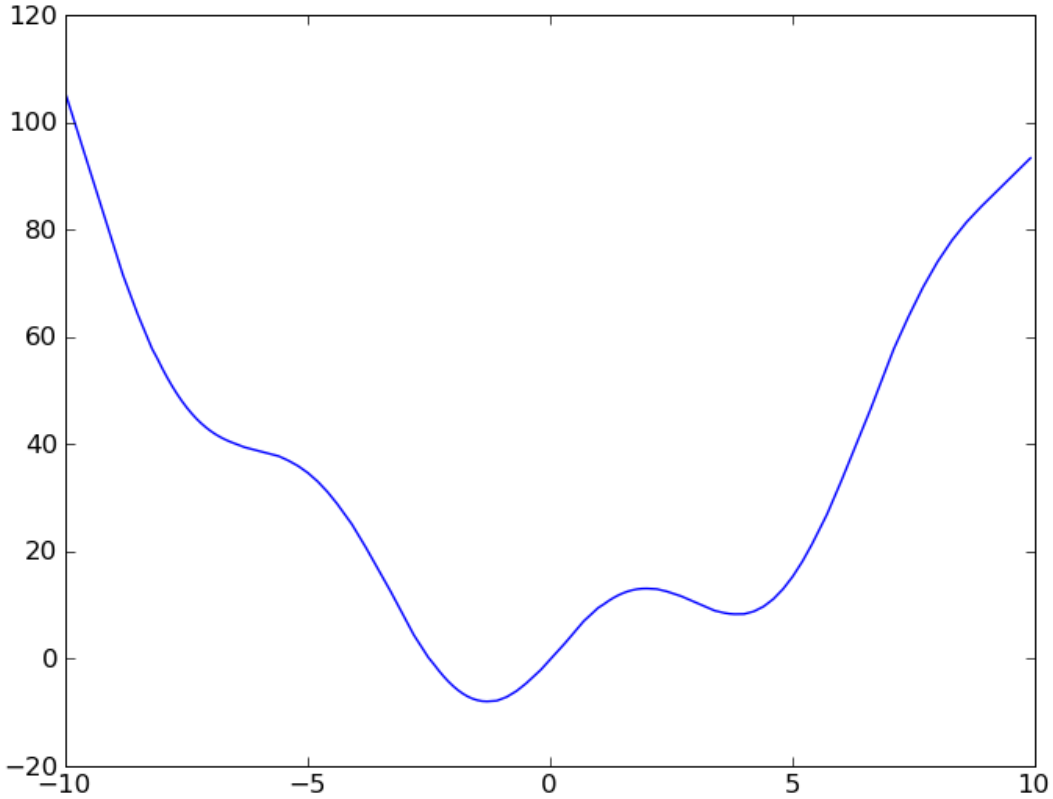


Interlude

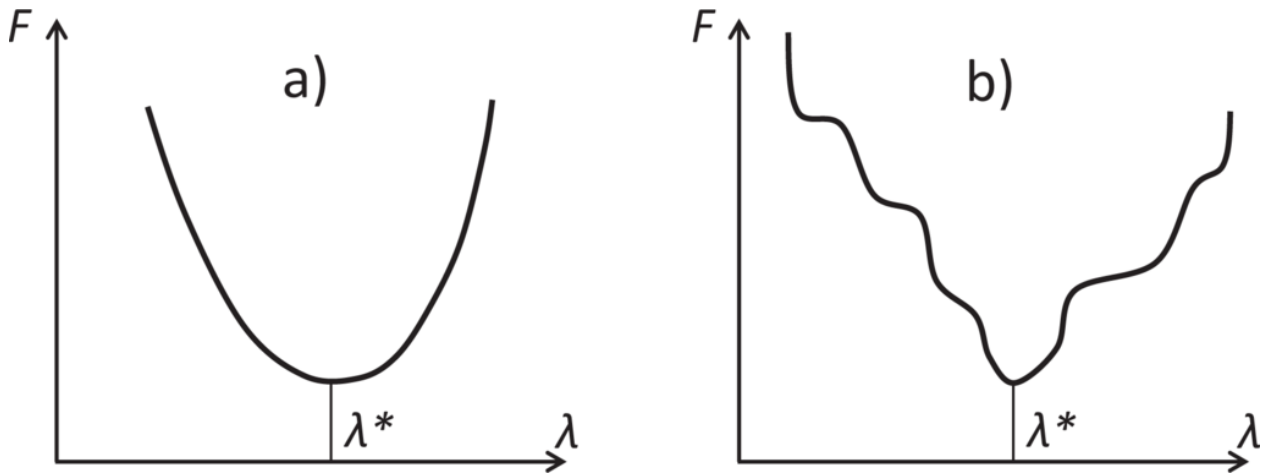
Convex function



Non-convex function



Not-convex-but-still-OK function



End of interlude

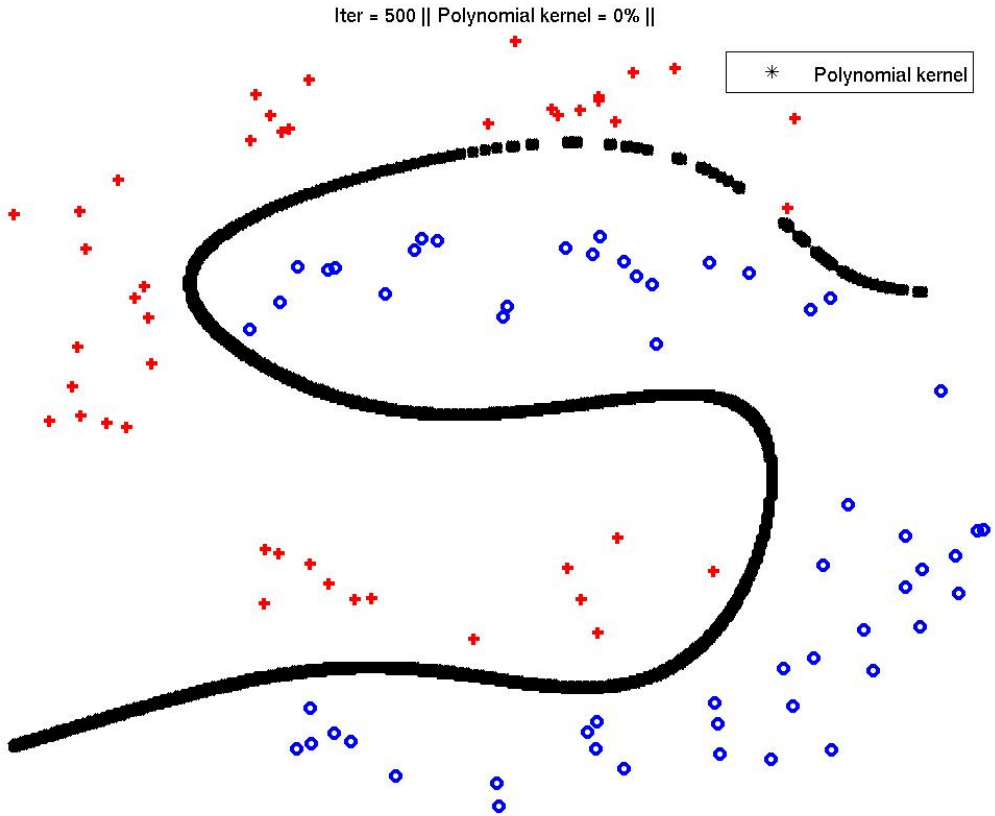
Solving separable AND non-separable problems

- *Non-linearly separable logistic*
- *Linearly separable logistic*

Non-linear classification

Non-linearly separable polynomial

Non-linear classification



- Features : $X_1, X_2 \rightarrow$ linear classifier
- Features : $X_1, X_2, X_1X_2, X_1^2, \ldots \rightarrow$ non-linear classifier

Choosing the features

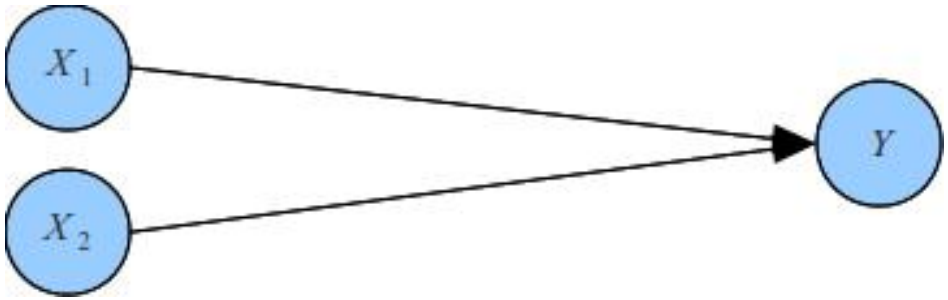
- To make it work, I created lots of extra features :
- $(X_1^i X_2^j)$ for $i, j \in [0, 4]$ (p = 18)

Choosing the features

- To make it work, I created lots of extra features :
- $(X_1^i X_2^j)$ for $i, j \in [0, 4]$ (p = 18)
- Would it work with fewer features ?
- Test with $(X_1^i X_2^j)$ for $i, j \in [1, 3]$

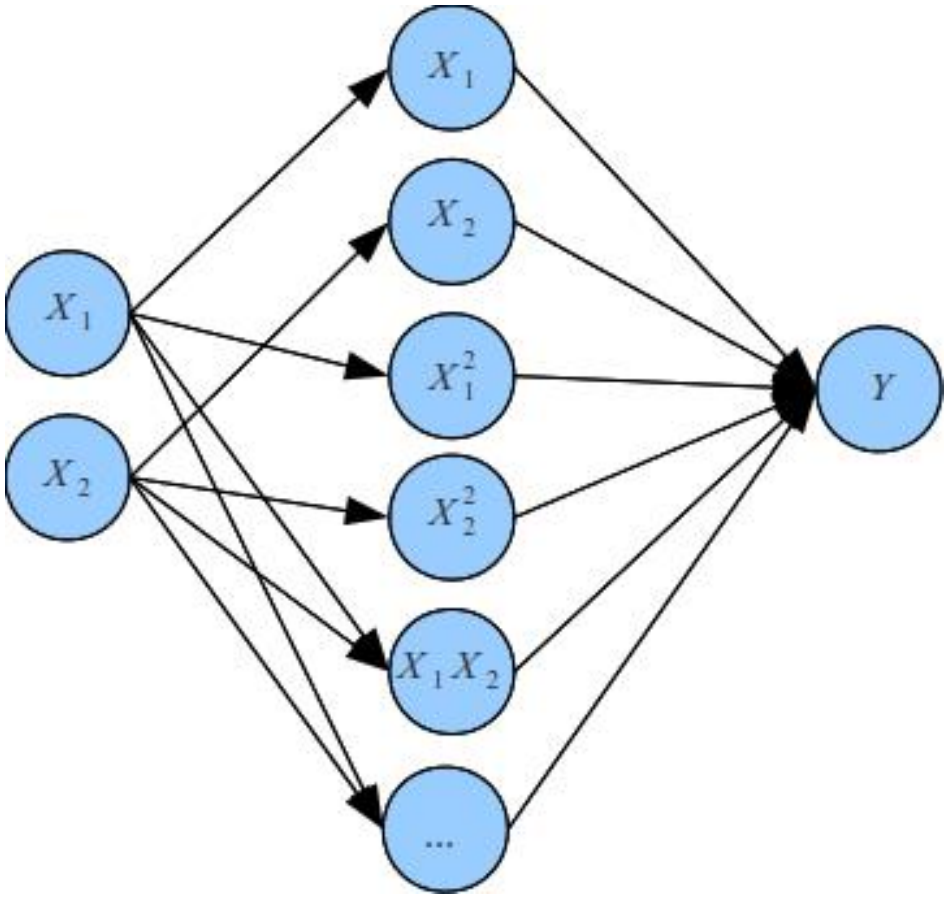
Non-linearly separable polynomial degree 2

A graphical view of the classifiers



$$f(X) = w_1 X_1 + w_2 X_2 + b$$

A graphical view of the classifiers



$$f(X) = w_1X_1 + w_2X_2 + w_3X_1^2 + w_4X_2^2 + w_5X_1X_2 + \dots$$

Non-linear features

- A linear classifier on a non-linear transformation is non-linear.
- A non-linear classifier relies on non-linear features.
- Which ones do we choose ?

Non-linear features

- A linear classifier on a non-linear transformation is non-linear.
- A non-linear classifier relies on non-linear features.
- Which ones do we choose ?
- Example : $H_j = X_1^{p_j} X_2^{q_j}$

Non-linear features

- A linear classifier on a non-linear transformation is non-linear.
- A non-linear classifier relies on non-linear features.
- Which ones do we choose ?
- Example : $H_j = X_1^{p_j} X_2^{q_j}$
- SVM : $H_j = K(X, X^{(j)})$ with K some kernel function

Non-linear features

- A linear classifier on a non-linear transformation is non-linear.
- A non-linear classifier relies on non-linear features.
- Which ones do we choose ?
- Example : $H_j = X_1^{p_j} X_2^{q_j}$
- SVM : $H_j = K(X, X^{(j)})$ with K some kernel function
- Do they have to be predefined ?

Non-linear features

- A linear classifier on a non-linear transformation is non-linear.
- A non-linear classifier relies on non-linear features.
- Which ones do we choose ?
- Example : $H_j = X_1^{p_j} X_2^{q_j}$
- SVM : $H_j = K(X, X^{(j)})$ with K some kernel function
- Do they have to be predefined ?
- A neural network will learn the H_j's

A 3-step algorithm for a neural network

1. Pick an example x

A 3-step algorithm for a neural network

1. Pick an example x
2. Transform it in $\hat{x} = Vx$ with some matrix V

A 3-step algorithm for a neural network

1. Pick an example x
2. Transform it in $\hat{x} = Vx$ with some matrix V
3. Compute $w^\top \hat{x} + b$

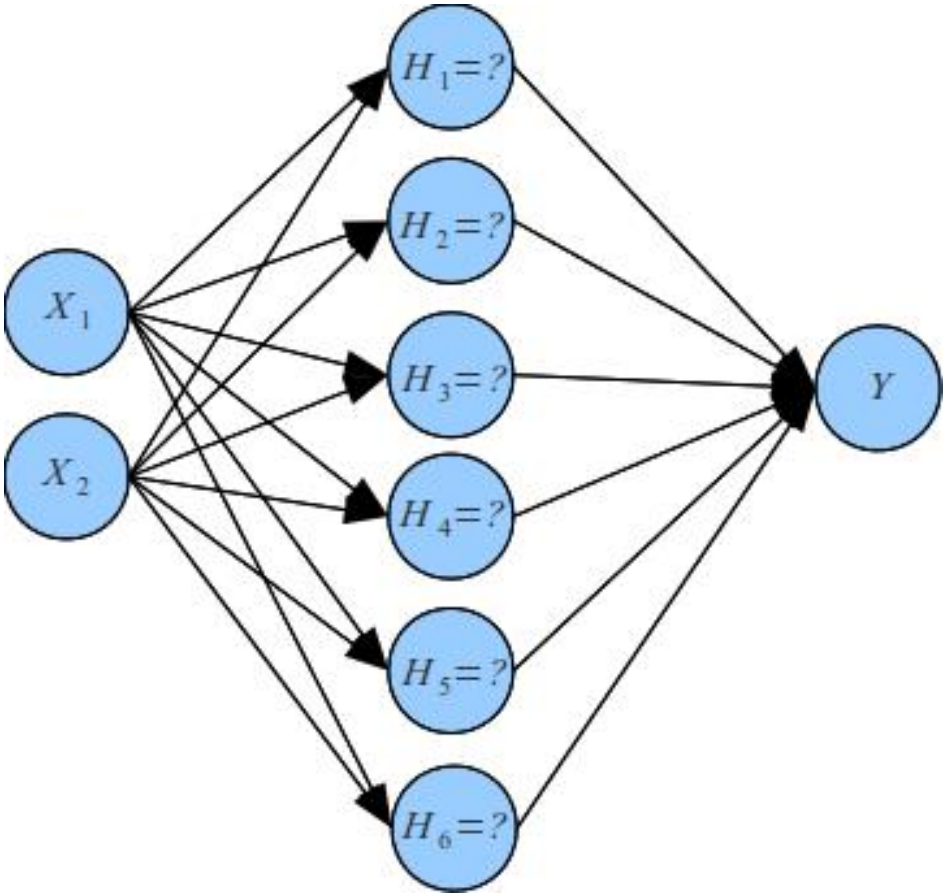
A 3-step algorithm for a neural network

1. Pick an example x
2. Transform it in $\hat{x} = Vx$ with some matrix V
3. Compute $w^\top \hat{x} + b = w^\top Vx + b = \hat{w}x + b$

A 3-step algorithm for a neural network

1. Pick an example x
2. Transform it in $\hat{x} = Vx$ with some matrix V
3. Apply a nonlinear function g to all the elements of $\hat{x}$
4. Compute $w^\top \hat{x} + b = w^\top g(Vx) + b \neq \hat{w}x + b$

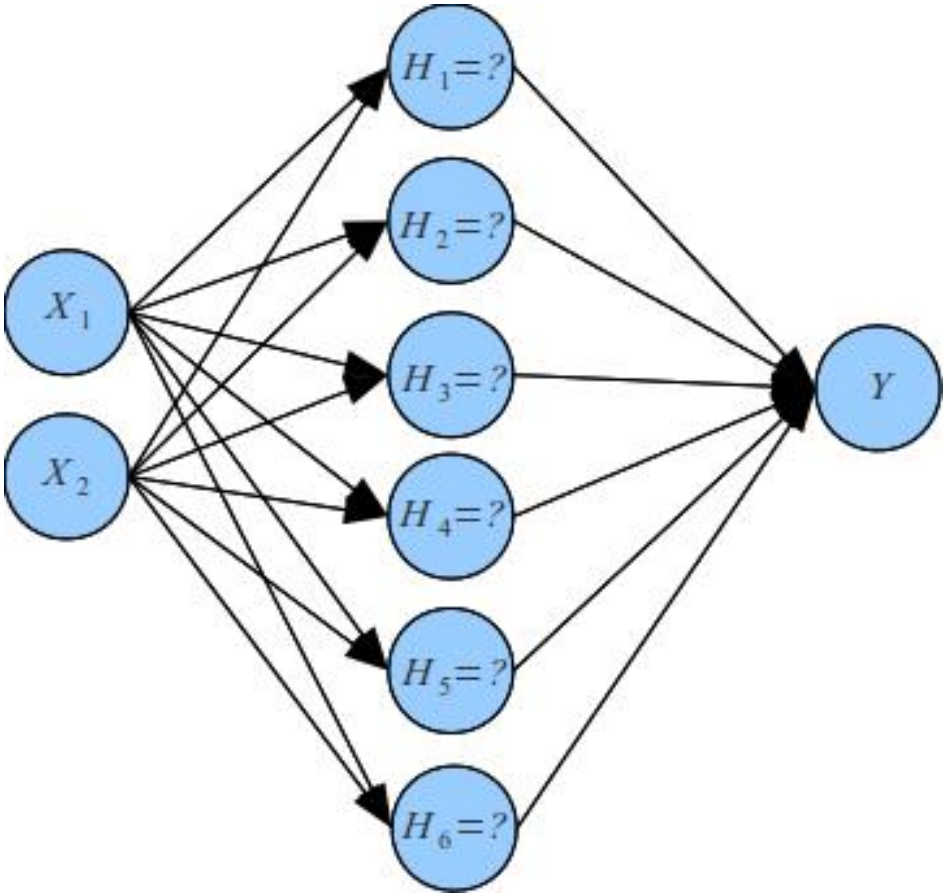
Neural networks



- Usually, we use

$$H_j = g(v_j^\top X)$$

Neural networks



- Usually, we use

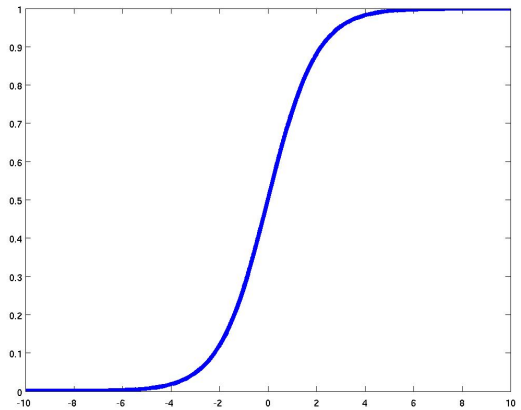
  $H_j = g(v_j^\top X)$

- H_j : Hidden unit
- v_j : Input weight
- g : Transfer function

Transfer function

$$f(X) = \sum_j w_j H_j(X) + b = \sum_j w_j g(v_j^\top X) + b$$

- g is the *transfer function*.
- g used to be the sigmoid or the tanh.

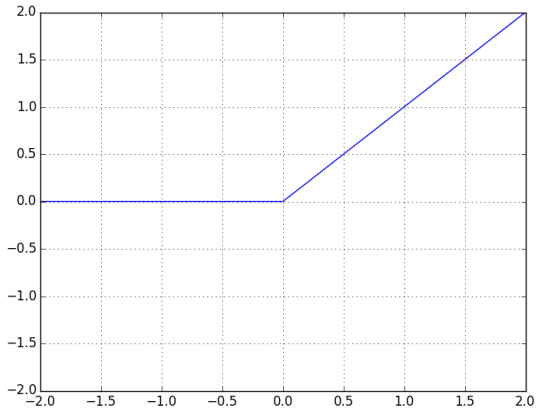


- If g is the sigmoid, each hidden unit is a *soft classifier*.

Transfer function

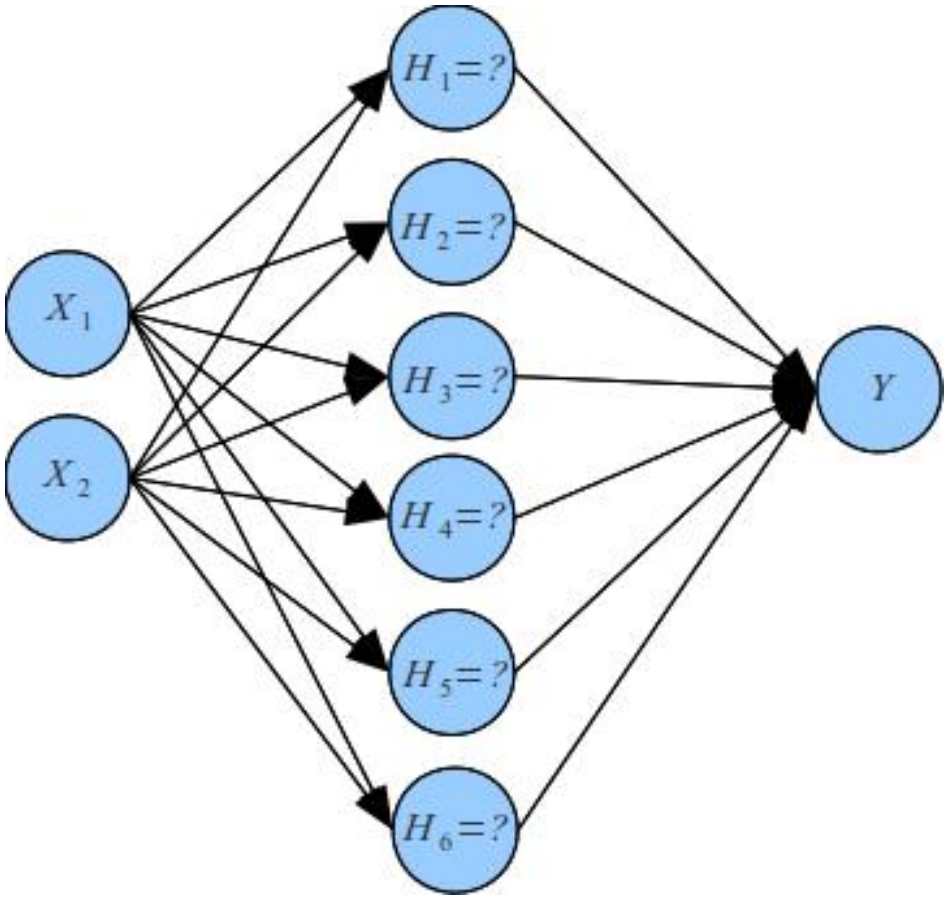
$$f(X) = \sum_j w_j H_j(X) + b = \sum_j w_j g(v_j^\top X) + b$$

- g is the *transfer function*.
- g is now often the *positive part*.



- This is called a rectified linear unit.

Neural networks



$$f(X) = \sum_j w_j H_j(X) + b = \sum_j w_j g(v_j^\top X) + b$$

Example on the non-separable problem

Non-linearly separable neural net 3

Training a neural network

- Dataset : $\left(X^{(i)}, Y^{(i)}\right)$ pairs, $i = 1, \ldots, N$.
- Goal : Find w and b such that

$$\text{sign}\left(w^\top X^{(i)} + b\right) = Y^{(i)}$$

Training a neural network

- Dataset : $\left(X^{(i)}, Y^{(i)}\right)$ pairs, $i = 1, \ldots, N$.
- Goal : Find w and b to minimize

$$\sum_i \log\left(1 + \exp\left(-Y^{(i)}\left(w^\top X^{(i)} + b\right)\right)\right)$$

Training a neural network

- Dataset : $\left(X^{(i)}, Y^{(i)}\right)$ pairs, $i = 1, \ldots, N$.
- Goal : Find $v_1, \ldots, v_k$, w and b to minimize

$$\sum_i \log\left(1 + \exp\left(-Y^{(i)}\left[\sum_j w_j g\left(v_j^\top X^{(i)}\right)\right]\right)\right)$$

Cost function

s = cost function (logistic loss, hinge loss, ...)

$$\begin{aligned}\ell(v, w, b, X^{(i)}, Y^{(i)}) &= s\left(\widehat{Y}^{(i)}, Y^{(i)}\right) \\ &= s\left(\sum_j w_j H_j(X^{(i)}), Y^{(i)}\right) \\ &= s\left(\sum_j w_j g\left(v_j^\top X^{(i)}\right), Y^{(i)}\right)\end{aligned}$$

Cost function

s = cost function (logistic loss, hinge loss, ...)

$$\begin{aligned}\ell(v, w, b, X^{(i)}, Y^{(i)}) &= s\left(\widehat{Y}^{(i)}, Y^{(i)}\right) \\ &= s\left(\sum_j w_j H_j(X^{(i)}), Y^{(i)}\right) \\ &= s\left(\sum_j w_j g\left(v_j^\top X^{(i)}\right), Y^{(i)}\right)\end{aligned}$$

We can minimize it using gradient descent.

Finding a good transformation of x

- $H_j(x) = g(v_j^\top x)$
- Is it a good H_j ? What can we say about it ?

Finding a good transformation of x

- $H_j(x) = g(v_j^\top x)$
- Is it a good H_j ? What can we say about it ?
- It is theoretically enough (universal approximation)

Finding a good transformation of x

- $H_j(x) = g(v_j^\top x)$
- Is it a good H_j ? What can we say about it ?
- It is theoretically enough (universal approximation)
- That does not mean you should use it

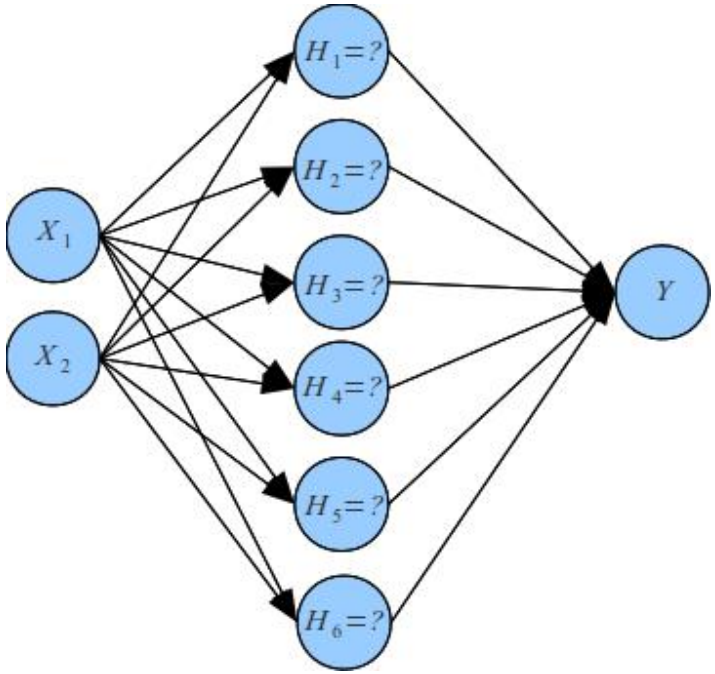
Going deeper

- $H_j(x) = g(v_j^\top x)$
- $\hat{x}_j = g(v_j^\top x)$

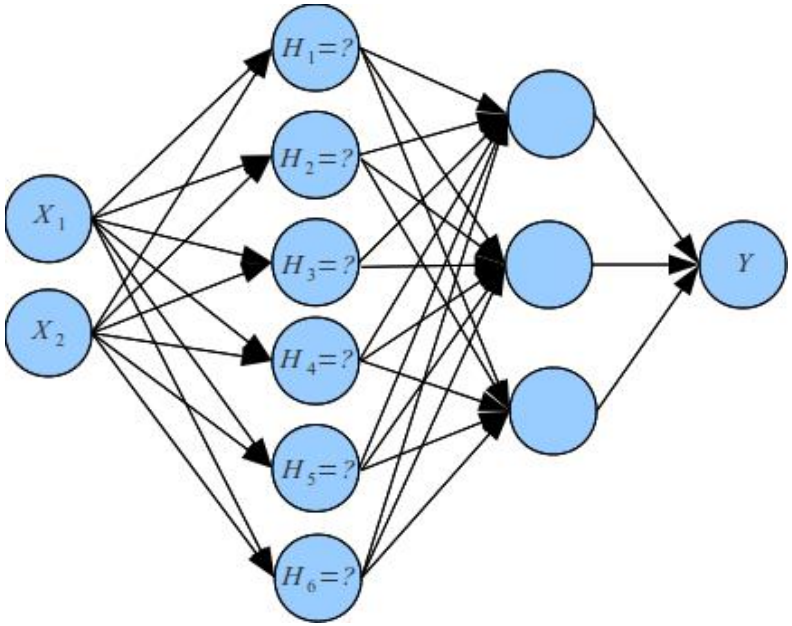
Going deeper

- $H_j(x) = g(v_j^\top x)$
- $\hat{x}_j = g(v_j^\top x)$
- We can transform $\hat{x}$ as well.

Going from 2 to 3 layers

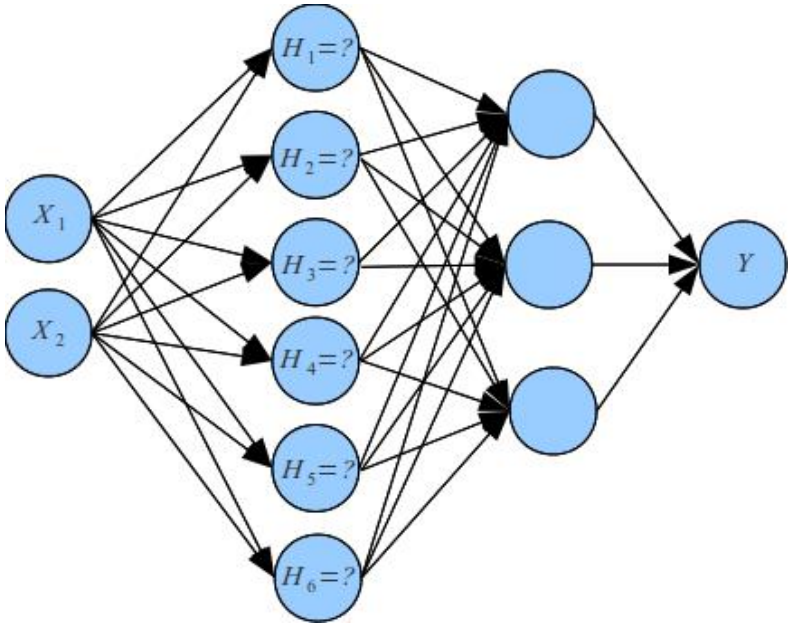


Going from 2 to 3 layers



- We can have as many layers as we like

Going from 2 to 3 layers

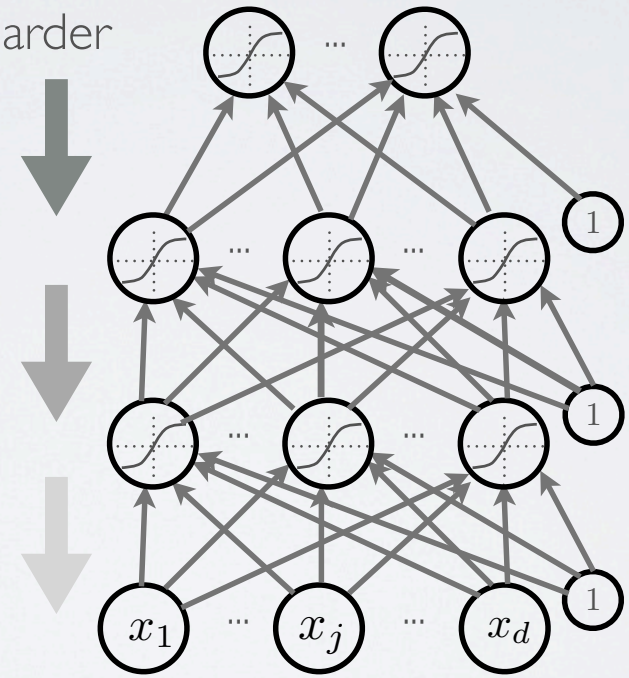


- We can have as many layers as we like
- But it makes the optimization problem harder

DEEP LEARNING

Topics: why training is hard

- First hypothesis: optimization is harder (underfitting)
  - vanishing gradient problem
  - saturated units block gradient propagation
- This is a well known problem in recurrent neural networks

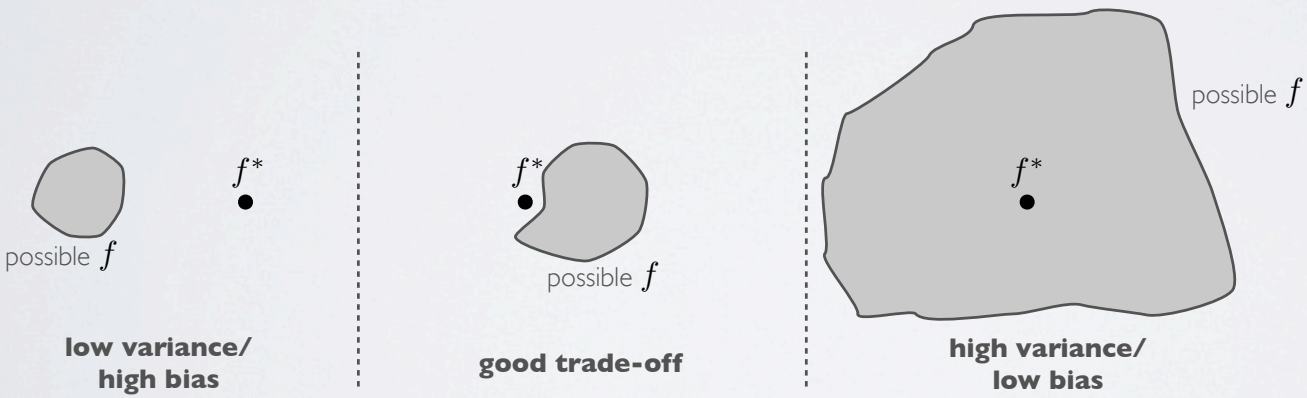


Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

DEEP LEARNING

Topics: why training is hard

- Second hypothesis: overfitting
  - we are exploring a space of complex functions
  - deep nets usually have lots of parameters
- Might be in a high variance / low bias situation



Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

DEEP LEARNING

Topics: why training is hard

- Depending on the problem, one or the other situation will tend to dominate

- If first hypothesis (underfitting): use better optimization
  - this is an active area of research

- If second hypothesis (overfitting): use better regularization
  - unsupervised learning
  - stochastic «dropout» training

Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

Regularizing deep networks

- Deep networks have many parameters
- How do we avoid overfitting ?

Regularizing deep networks

- Deep networks have many parameters
- How do we avoid overfitting ?
  - Regularizing the parameters

Regularizing deep networks

- Deep networks have many parameters
- How do we avoid overfitting ?
  - Regularizing the parameters
  - Limiting the number of units

Regularizing deep networks

- Deep networks have many parameters
- How do we avoid overfitting ?
  - Regularizing the parameters
  - Limiting the number of units
  - Dropout

Dropout

- Overfitting happens when each unit gets too specialized

Dropout

- Overfitting happens when each unit gets too specialized
- The core idea is to prevent each unit from relying too much on the others

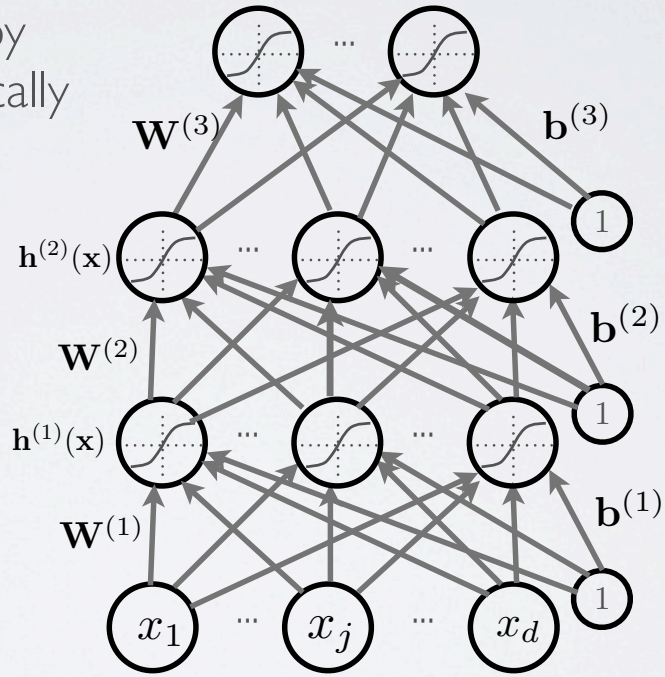
Dropout

- Overfitting happens when each unit gets too specialized
- The core idea is to prevent each unit from relying too much on the others
- How do we do this ?

DROPOUT

Topics: dropout

- Idea: «cripple» neural network by removing hidden units stochastically
  - each hidden unit is set to 0 with probability 0.5
  - hidden units cannot co-adapt to other units
  - hidden units must be more generally useful
- Could use a different dropout probability, but 0.5 usually works well

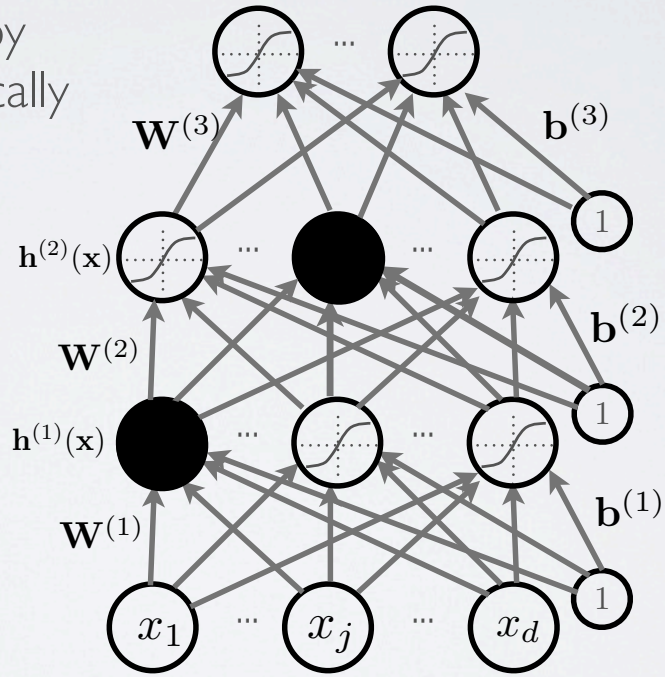


Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

DROPOUT

Topics: dropout

- Idea: «cripple» neural network by removing hidden units stochastically
  - each hidden unit is set to 0 with probability 0.5
  - hidden units cannot co-adapt to other units
  - hidden units must be more generally useful
- Could use a different dropout probability, but 0.5 usually works well

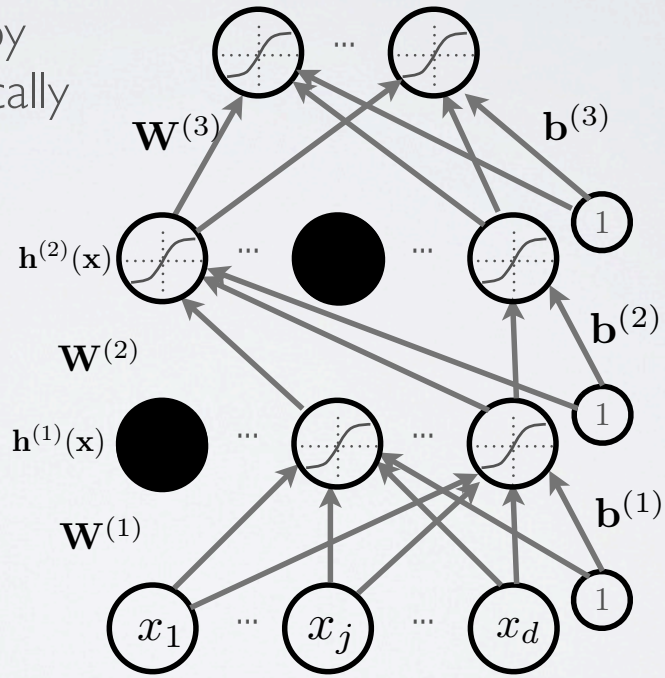


Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

DROPOUT

Topics: dropout

- Idea: «cripple» neural network by removing hidden units stochastically
  - each hidden unit is set to 0 with probability 0.5
  - hidden units cannot co-adapt to other units
  - hidden units must be more generally useful
- Could use a different dropout probability, but 0.5 usually works well



Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

DROPOUT

Topics: dropout

- Use random binary masks $\mathbf{m}^{(k)}$
  - layer pre-activation for $k>0$ $(\mathbf{h}^{(0)}(\mathbf{x}) = \mathbf{x})$

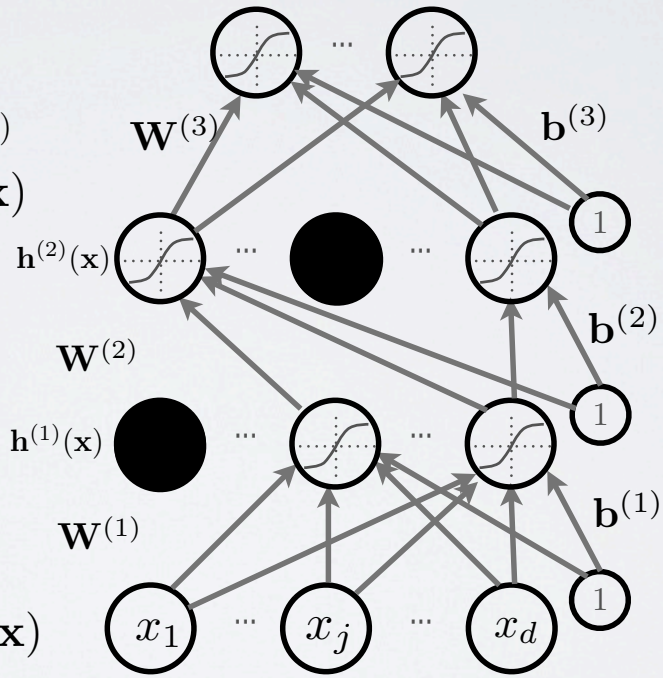
    $$\mathbf{a}^{(k)}(\mathbf{x}) = \mathbf{b}^{(k)} + \mathbf{W}^{(k)}\mathbf{h}^{(k-1)}(\mathbf{x})$$

  - hidden layer activation (k from 1 to L):

    $$\mathbf{h}^{(k)}(\mathbf{x}) = \mathbf{g}(\mathbf{a}^{(k)}(\mathbf{x}))$$

  - output layer activation ($k=L+1$):

    $$\mathbf{h}^{(L+1)}(\mathbf{x}) = \mathbf{o}(\mathbf{a}^{(L+1)}(\mathbf{x})) = \mathbf{f}(\mathbf{x})$$



Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

DROPOUT

Topics: dropout

- Use random binary masks $\mathbf{m}^{(k)}$
  - layer pre-activation for $k>0$ $(\mathbf{h}^{(0)}(\mathbf{x}) = \mathbf{x})$

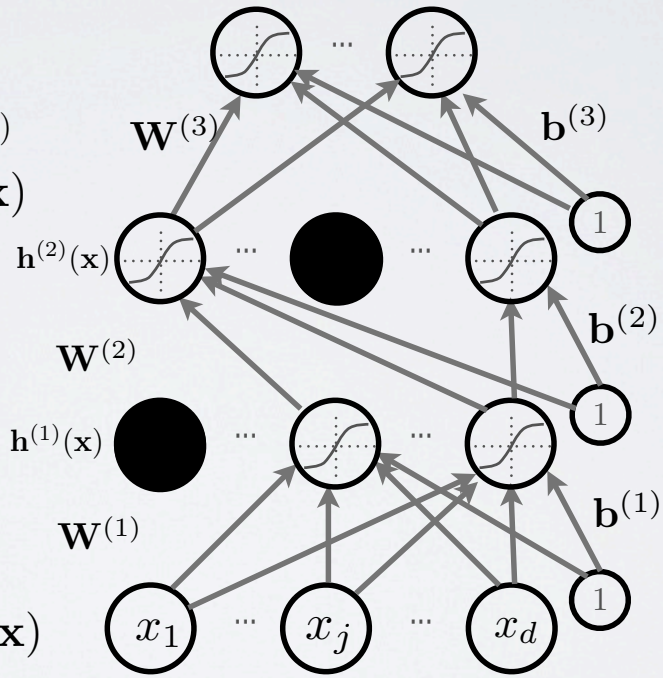
    $$\mathbf{a}^{(k)}(\mathbf{x}) = \mathbf{b}^{(k)} + \mathbf{W}^{(k)}\mathbf{h}^{(k-1)}(\mathbf{x})$$

  - hidden layer activation (k from 1 to L):

    $$\mathbf{h}^{(k)}(\mathbf{x}) = \mathbf{g}(\mathbf{a}^{(k)}(\mathbf{x})) \odot \mathbf{m}^{(k)}$$

  - output layer activation ($k=L+1$):

    $$\mathbf{h}^{(L+1)}(\mathbf{x}) = \mathbf{o}(\mathbf{a}^{(L+1)}(\mathbf{x})) = \mathbf{f}(\mathbf{x})$$



Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

Dropout - a conclusion

- By removing units at random, we force the others to be less specialized
- At test time, we use all the units

Dropout - a conclusion

- By removing units at random, we force the others to be less specialized
- At test time, we use all the units
- This is sometimes presented as an ensemble method.

Neural networks - Summary

- A linear classifier in a feature space can model non-linear boundaries.
- Finding a good feature space is essential.
- One can design the feature map by hand.
- One can learn the feature map, using fewer features than if it done by hand.
- Learning the feature map is potentially HARD (non-convexity).

Recipe for training a neural network

- Use rectified linear units
- Use dropout
- Use stochastic gradient descent
- Use 2000 units on each layer
- Try different number of layers

The need for fast learning

- Neural networks may need many examples (several millions or more).
- We need to be able to use them quickly.

Batch methods

$$L(\theta) = \frac{1}{N} \sum_i \ell(\theta, X^{(i)}, Y^{(i)})$$

$$\theta_{t+1} \rightarrow \theta_t - \frac{\alpha_t}{N} \sum_i \frac{\partial \ell(\theta, X^{(i)}, Y^{(i)})}{\partial \theta}$$

- To compute one update of the parameters, we need to go through all the data.
- This can be very expensive.
- What if we have an infinite amount of data ?

Potential solutions

1. Discard data.
   - Seems stupid
   - Yet many people do it

Potential solutions

1. Discard data.
   - Seems stupid
   - Yet many people do it
2. Use approximate methods.
   - Update = average of the updates for all datapoints.
   - Are these update really different ?
   - If not, how can we learn faster ?

Stochastic gradient descent

$$L(\theta) = \frac{1}{N} \sum_i \ell(\theta, X^{(i)}, Y^{(i)})$$

$$\theta_{t+1} \rightarrow \theta_t - \frac{\alpha_t}{N} \sum_i \frac{\partial \ell(\theta, X^{(i)}, Y^{(i)})}{\partial \theta}$$

Stochastic gradient descent

$$L(\theta) = \frac{1}{N} \sum_i \ell(\theta, X^{(i)}, Y^{(i)})$$

$$\theta_{t+1} \rightarrow \theta_t - \alpha_t \frac{\partial \ell(\theta, X^{(i_t)}, Y^{(i_t)})}{\partial \theta}$$

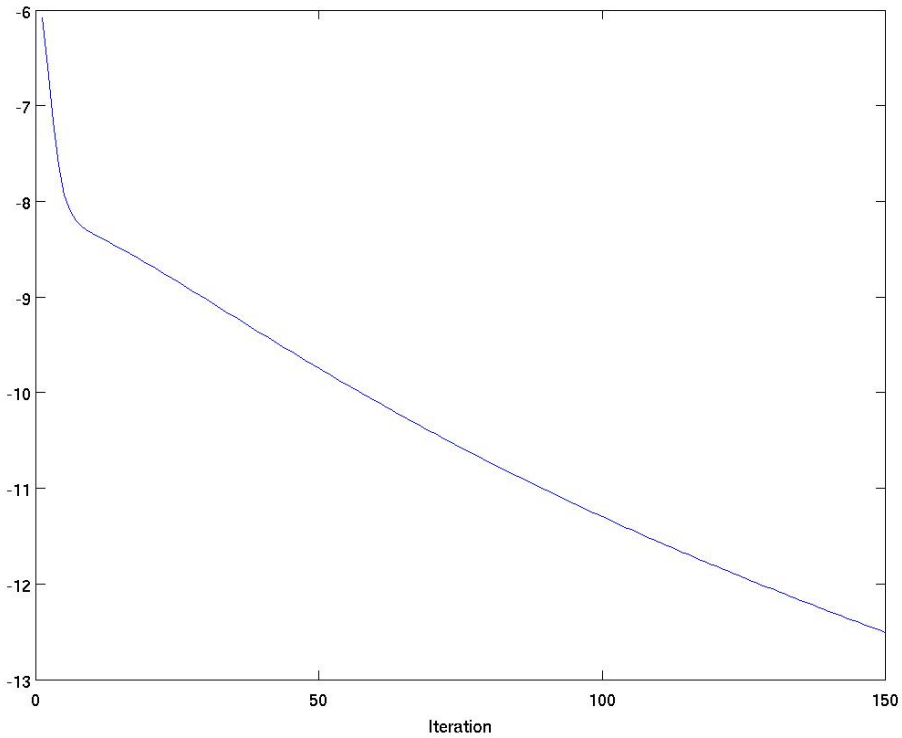
Stochastic gradient descent

$$L(\theta) = \frac{1}{N} \sum_i \ell(\theta, X^{(i)}, Y^{(i)})$$

$$\theta_{t+1} \rightarrow \theta_t - \alpha_t \frac{\partial \ell(\theta, X^{(i_t)}, Y^{(i_t)})}{\partial \theta}$$

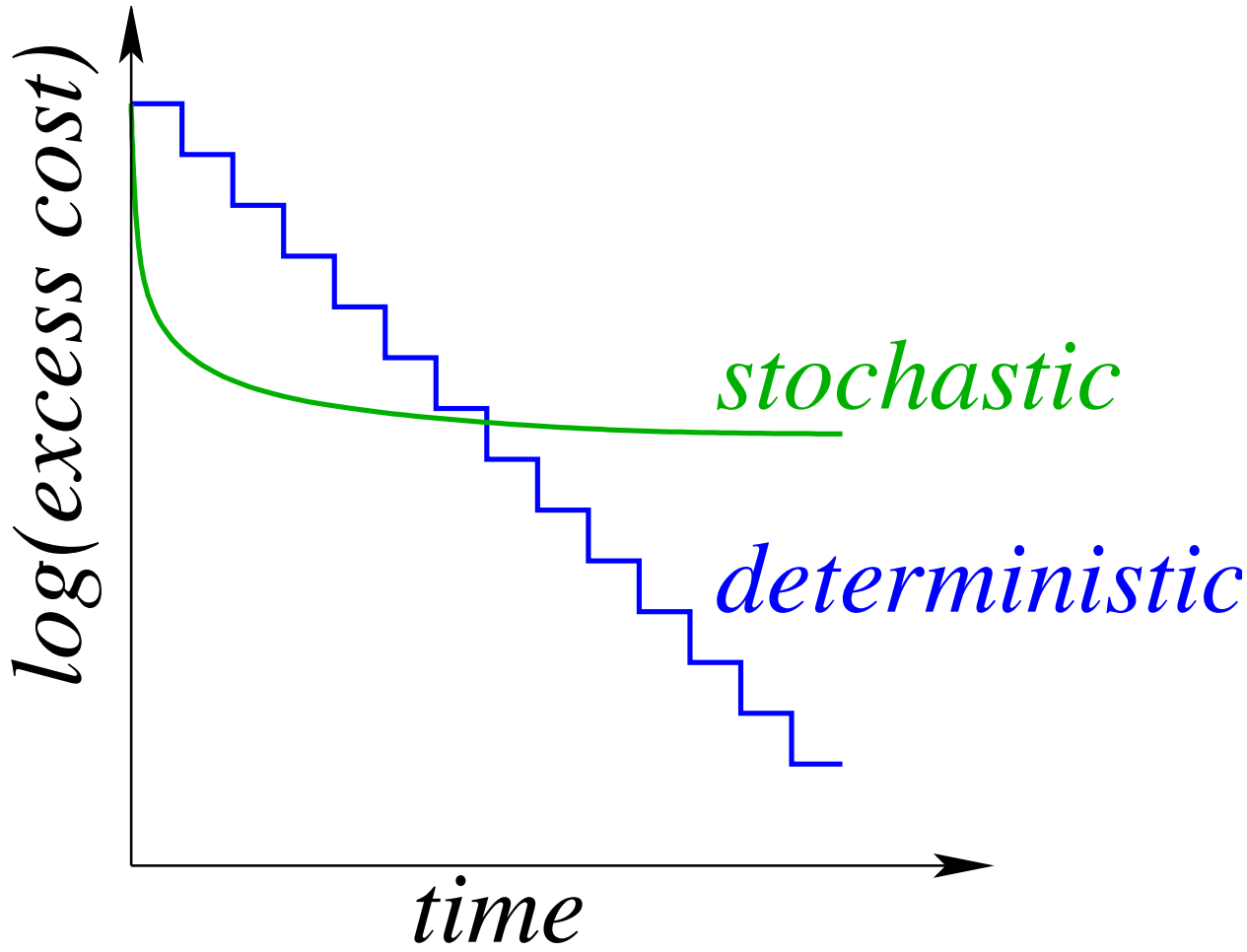
What do we lose when updating the parameters to satisfy just one example ?

Disagreement

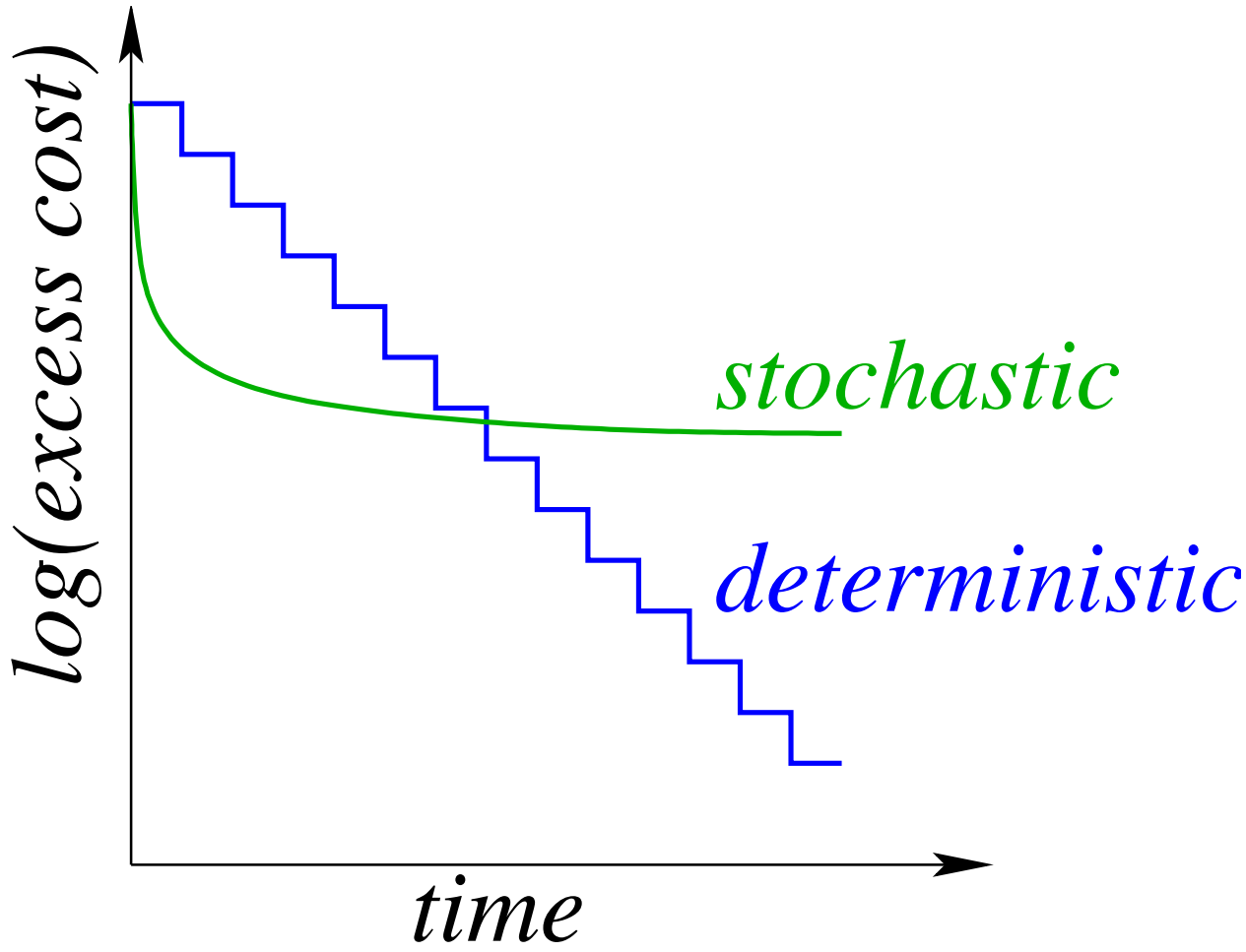


- $\|\mu\|^2/\sigma^2$ during optimization (log scale)
- As optimization progresses, disagreement increases
- Early on, one can pick one example at a time
- What about later ?

Stochastic vs. batch methods

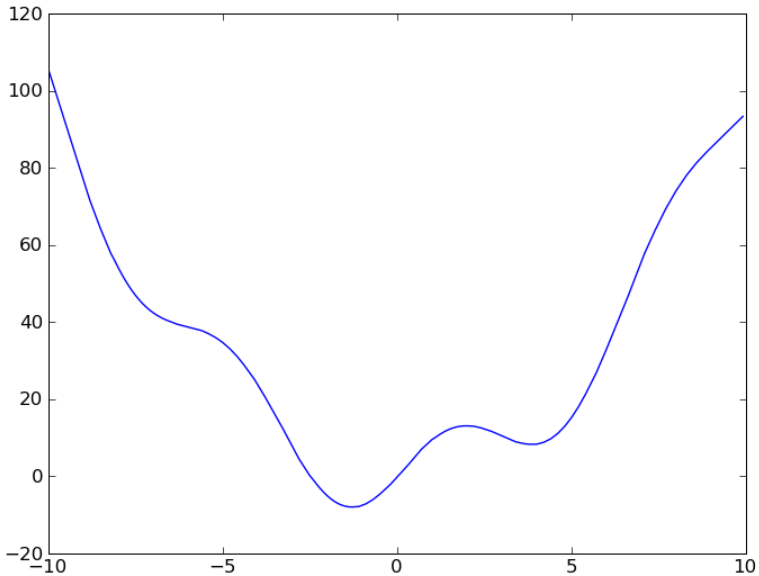


Stochastic vs. batch methods

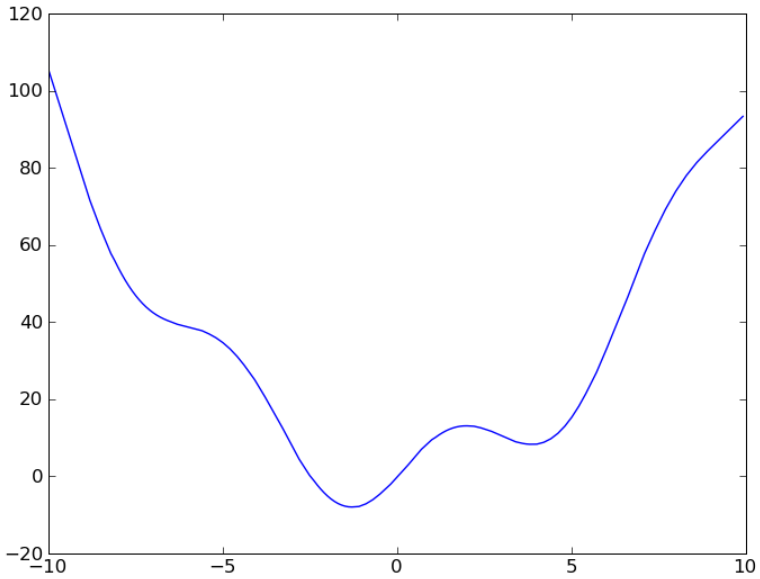


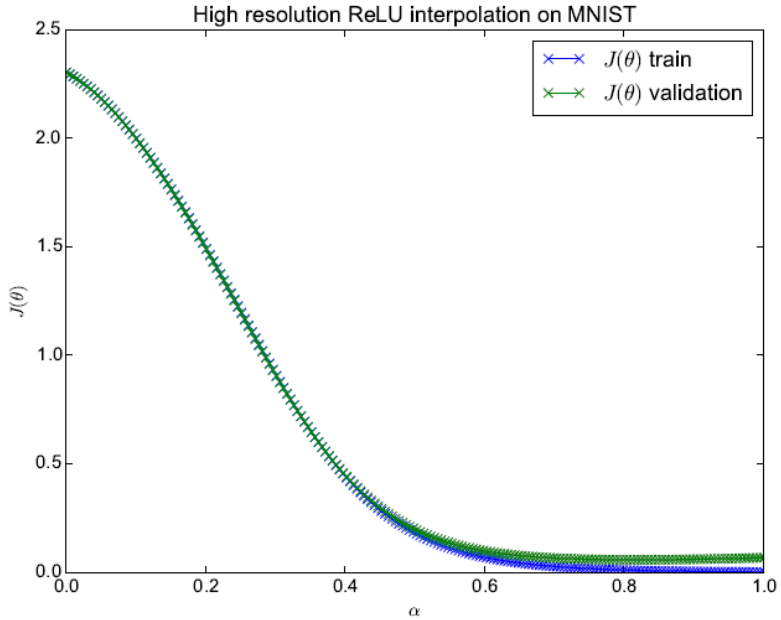
- For non-convex problem, stochastic works best.

Understanding the loss function of deep networks

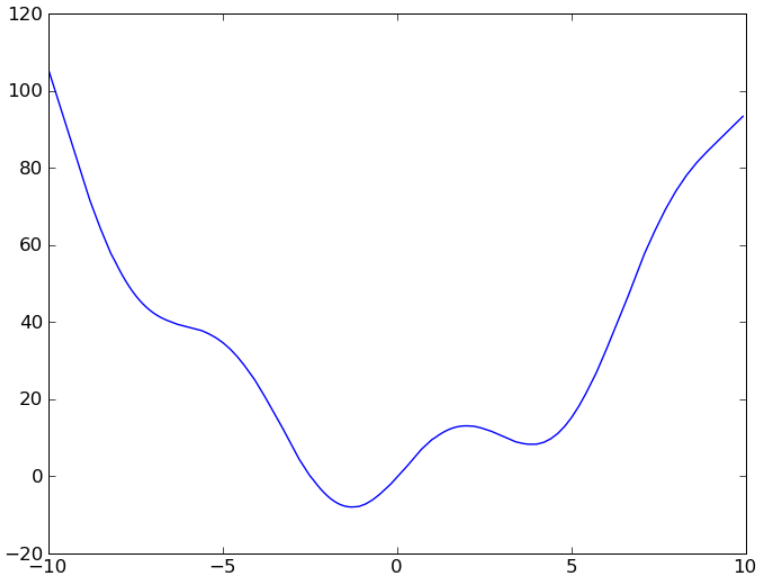


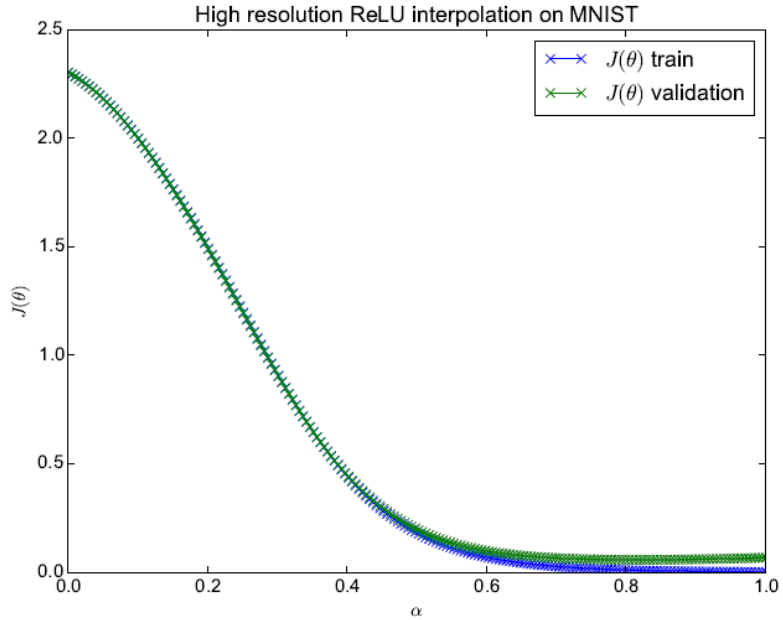
Understanding the loss function of deep networks





Understanding the loss function of deep networks





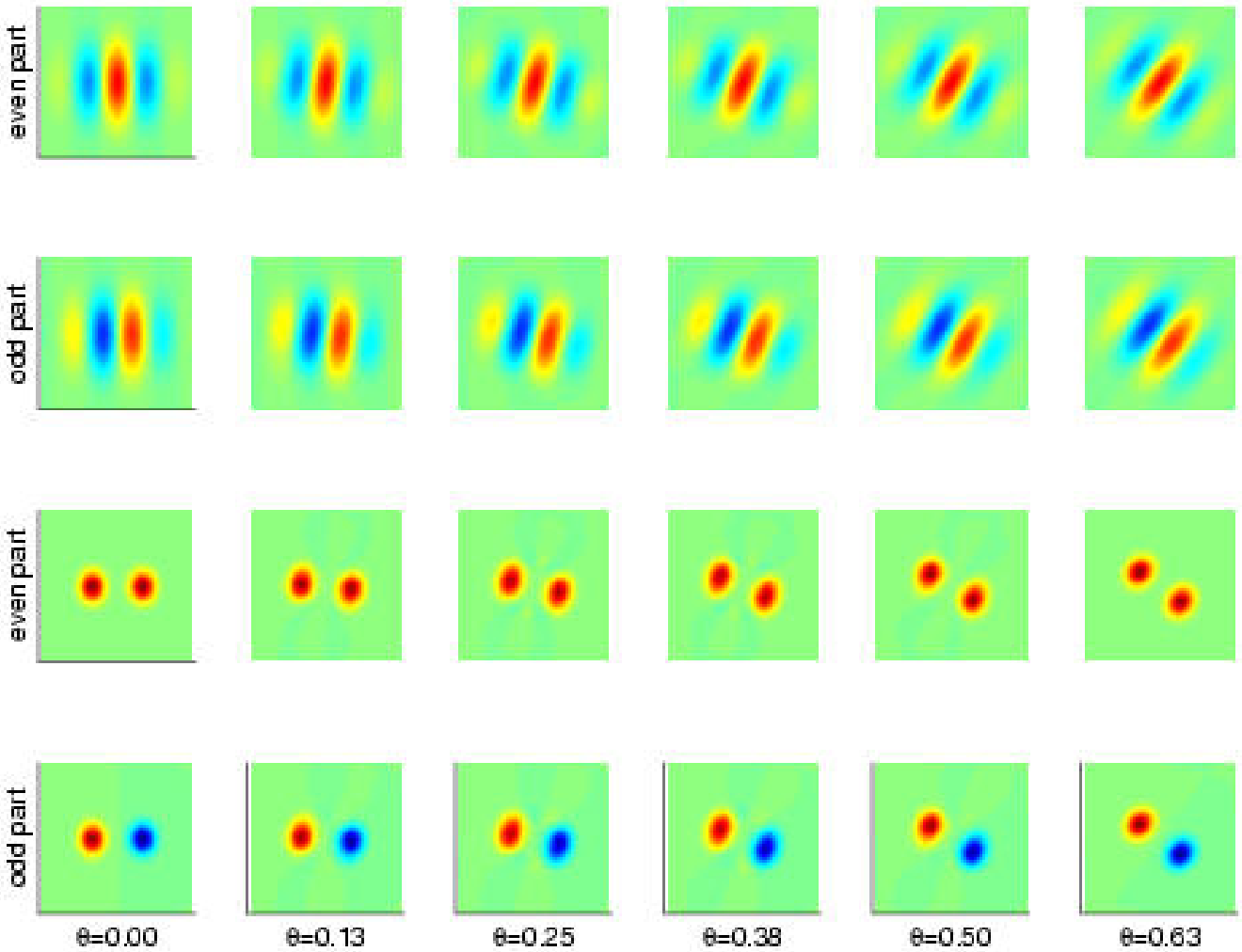
- Recent studies say most local minima are close to the optimum.

v_j's for images

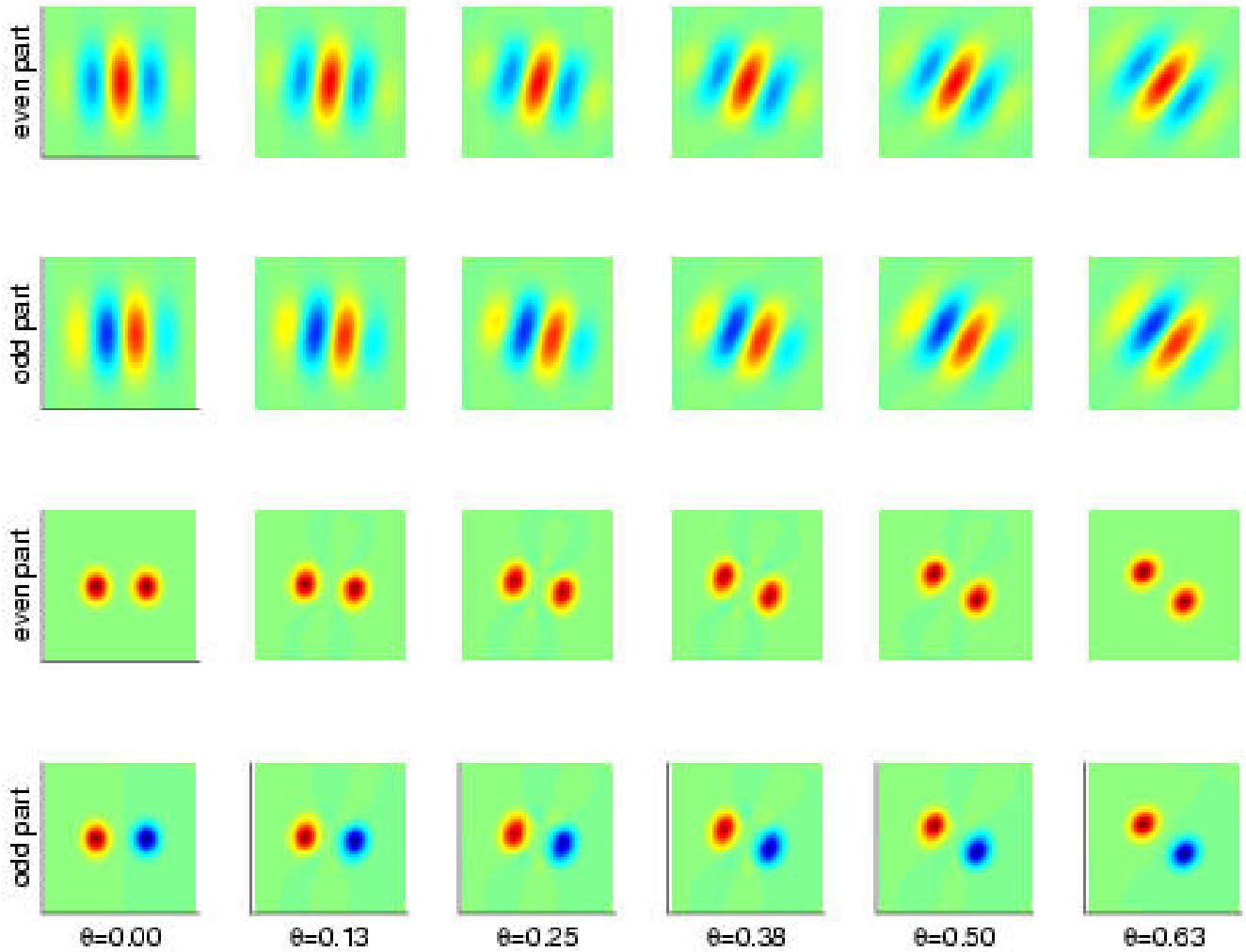
$$f(X) = \sum_j w_j H_j(X) + b = \sum_j w_j g(v_j^\top X) + b$$

- If X is an image, v_j is an image too.
- v_j acts as a filter (presence or absence of a pattern).
- What does v_j look like ?

v_j's for images - Examples



v_j's for images - Examples



- Filters are mostly local

COMPUTER VISION

Topics: computer vision

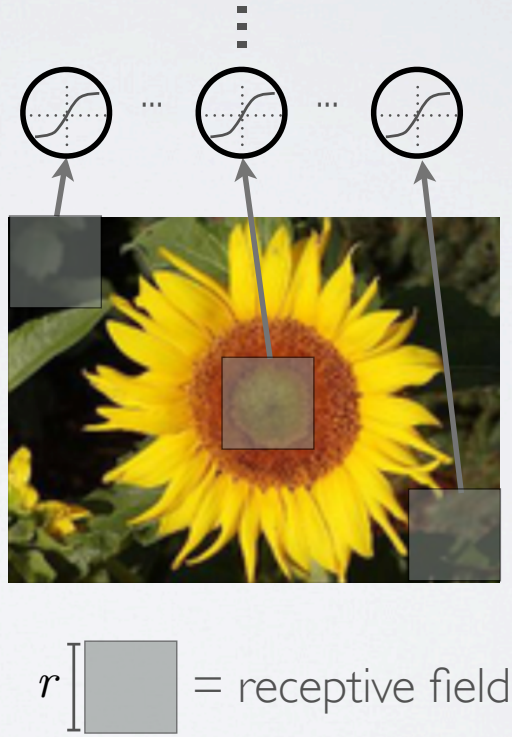
- We can design neural networks that are specifically adapted for such problems
  - must deal with very high-dimensional inputs
    - 150 x 150 pixels = 22500 inputs, or 3 x 22500 if RGB pixels
  - can exploit the 2D topology of pixels (or 3D for video data)
  - can build in invariance to certain variations we can expect
    - translations, illumination, etc.
- Convolutional networks leverage these ideas
  - **local connectivity**
  - parameter sharing
  - pooling / subsampling hidden units

Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

COMPUTER VISION

Topics: local connectivity

- First idea: use a local connectivity of hidden units
  - each hidden unit is connected only to a subregion (patch) of the input image
  - it is connected to all channels
    - 1 if greyscale image
    - 3 (R, G, B) for color image
- Solves the following problems:
  - fully connected hidden layer would have an unmanageable number of parameters
  - computing the linear activations of the hidden units would be very expensive

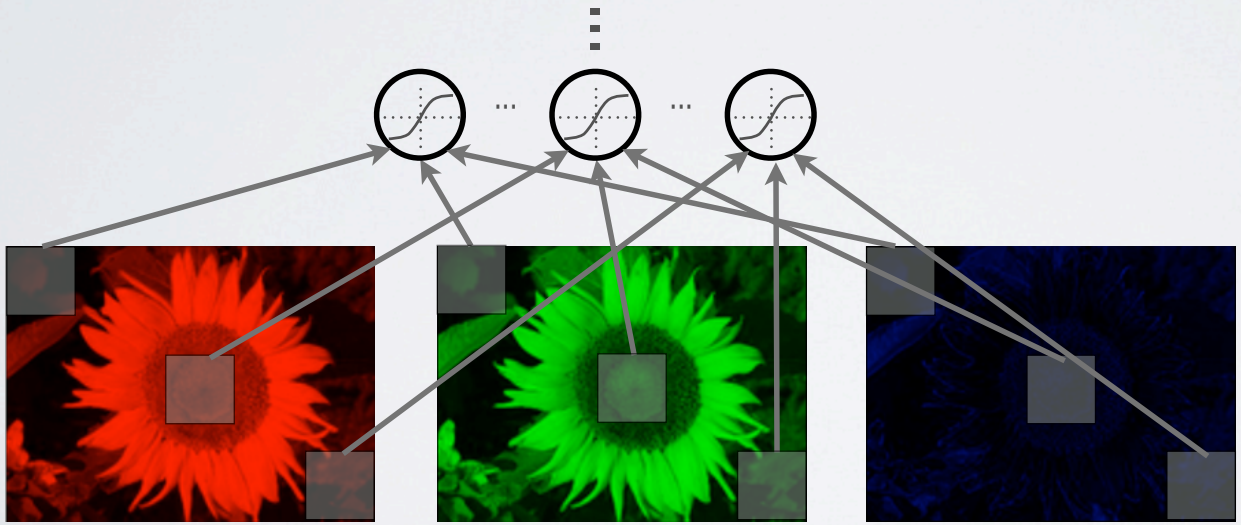


Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

COMPUTER VISION

Topics: local connectivity

- Units are connected to all channels:
  - 1 channel if grayscale image, 3 channels (R, G, B) if color image



Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

COMPUTER VISION

Topics: computer vision

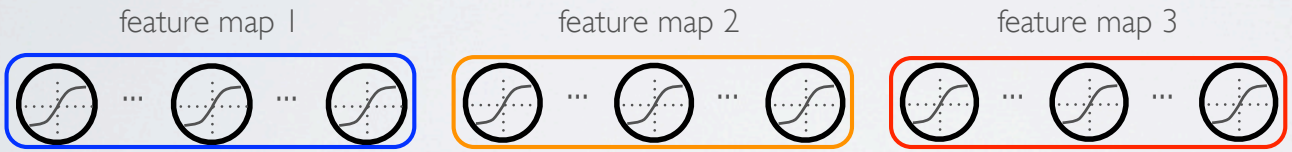
- We can design neural networks that are specifically adapted for such problems
  - must deal with very high-dimensional inputs
    - 150 x 150 pixels = 22500 inputs, or 3 x 22500 if RGB pixels
  - can exploit the 2D topology of pixels (or 3D for video data)
  - can build in invariance to certain variations we can expect
    - translations, illumination, etc.
- Convolutional networks leverage these ideas
  - local connectivity
  - **parameter sharing**
  - pooling / subsampling hidden units

Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

COMPUTER VISION

Topics: parameter sharing

- Second idea: share matrix of parameters across certain units
  - units organized into the same "feature map" share parameters
  - hidden units within a feature map cover different positions in the image



same color
=
same matrix
of connections

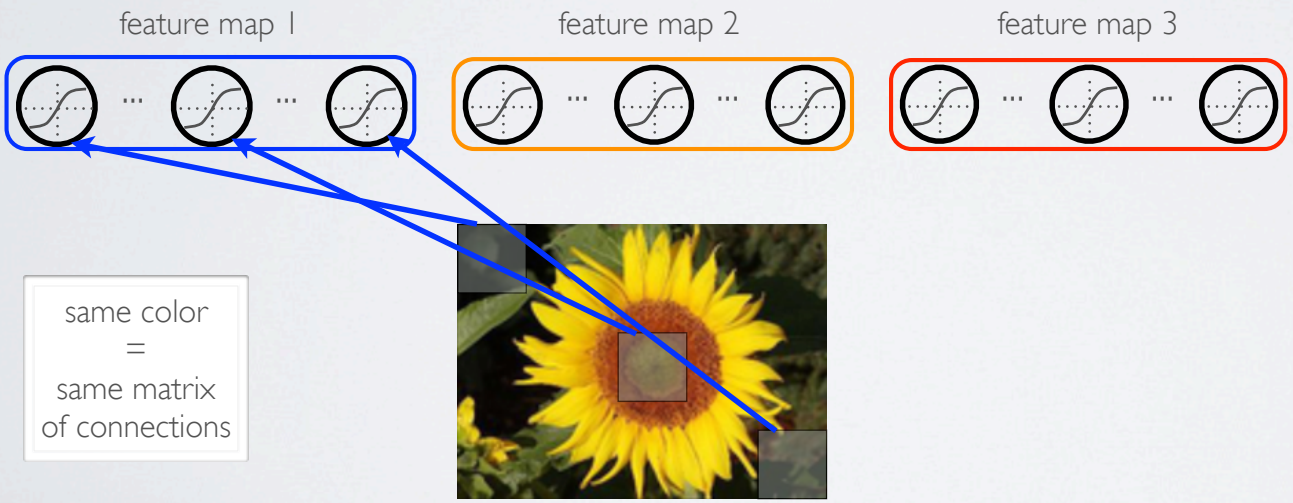
6

Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

COMPUTER VISION

Topics: parameter sharing

- Second idea: share matrix of parameters across certain units
  - units organized into the same "feature map" share parameters
  - hidden units within a feature map cover different positions in the image



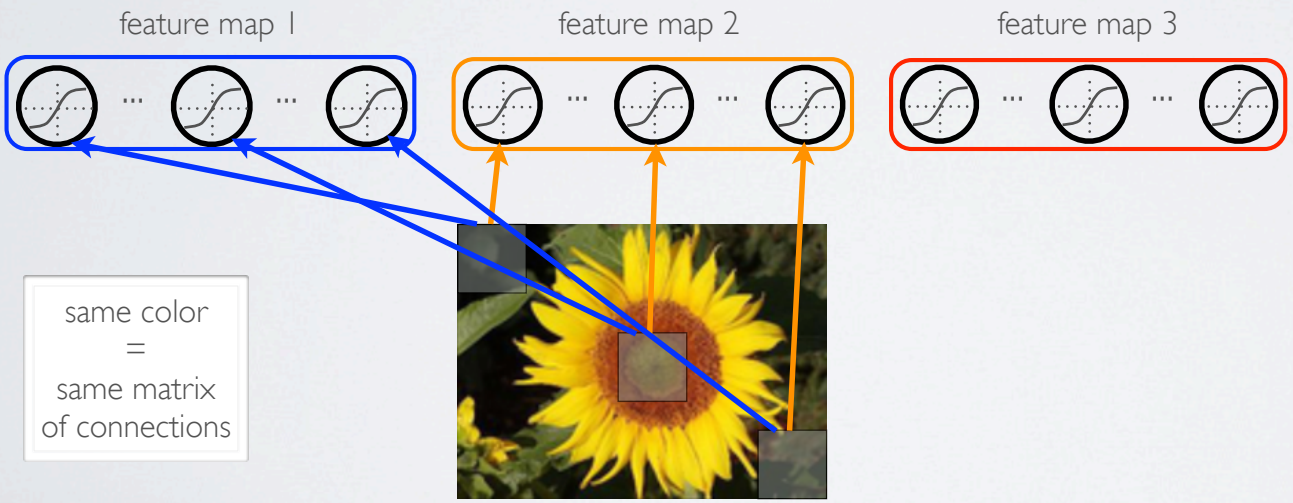
6

Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

COMPUTER VISION

Topics: parameter sharing

- Second idea: share matrix of parameters across certain units
  - units organized into the same "feature map" share parameters
  - hidden units within a feature map cover different positions in the image



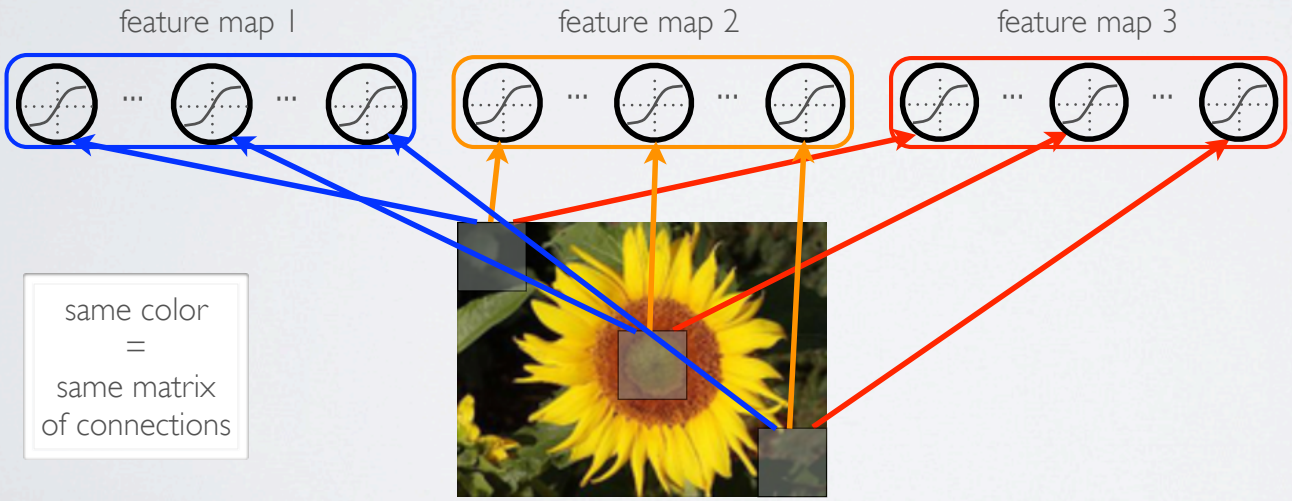
6

Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

COMPUTER VISION

Topics: parameter sharing

- Second idea: share matrix of parameters across certain units
  - units organized into the same "feature map" share parameters
  - hidden units within a feature map cover different positions in the image



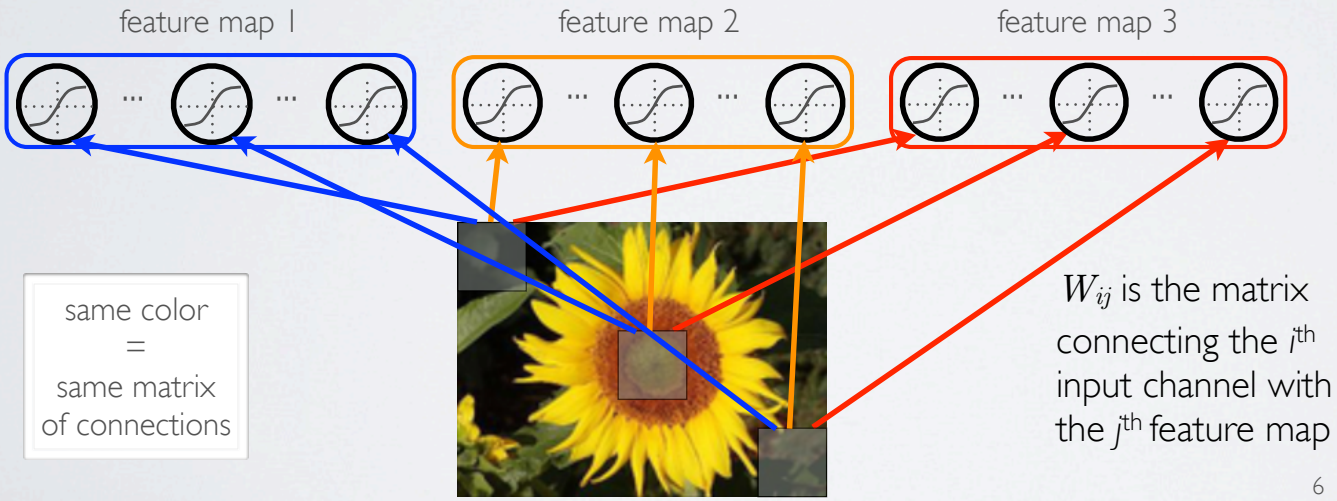
6

Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

COMPUTER VISION

Topics: parameter sharing

- Second idea: share matrix of parameters across certain units
  - units organized into the same "feature map" share parameters
  - hidden units within a feature map cover different positions in the image

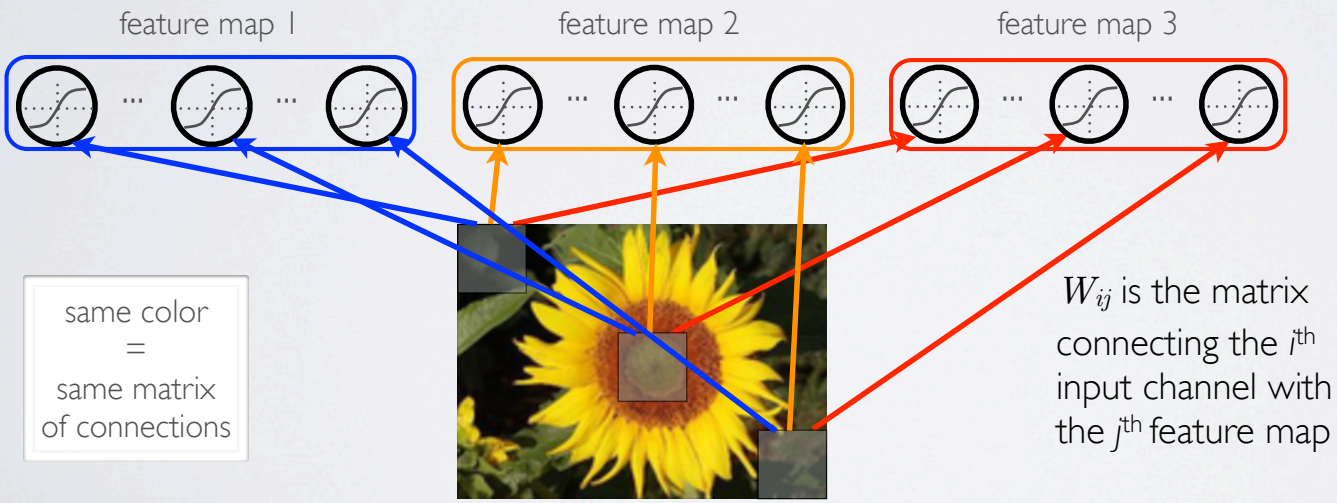


Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

COMPUTER VISION

Topics: parameter sharing

- Solves the following problems:
  - reduces even more the number of parameters
  - will extract the same features at every position (features are "equivariant")



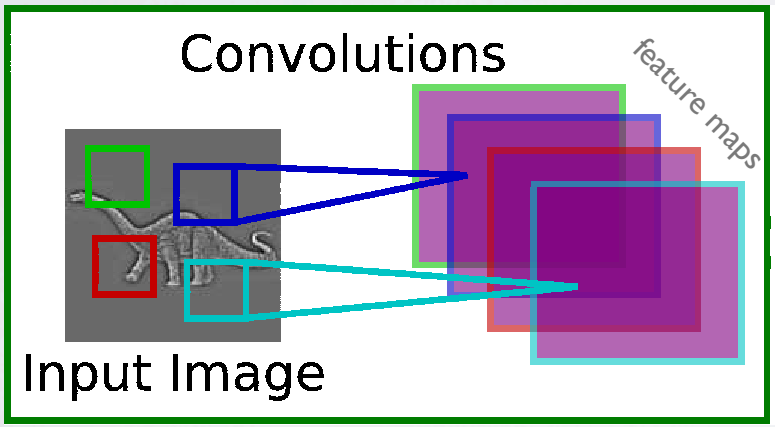
Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

COMPUTER VISION

Topics: parameter sharing

Jarret et al. 2009

- Each feature map forms a 2D grid of features
  - can be computed with a discrete convolution ($*$) of a kernel matrix k_{ij} which is the hidden weights matrix W_{ij} with its rows and columns flipped



- x_i is the i[th] channel of input
- k_{ij} is the convolution kernel
- g_j is a learned scaling factor
- y_j is the hidden layer

(could have added a bias)

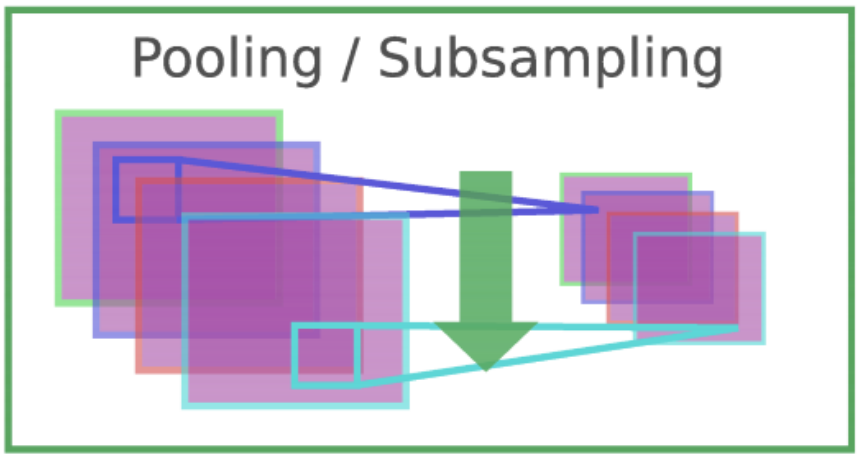
$$y_j = g_j \tanh(\sum_i k_{ij} * x_i)$$

Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

COMPUTER VISION

Topics: pooling and subsampling

- Third idea: pool hidden units in same neighborhood
  - pooling is performed in non-overlapping neighborhoods (subsampling)



Jarret et al. 2009

- $x_{i,j,k}$ is value of the ith feature map at position j,k
- p is vertical index in local neighborhood
- q is horizontal index in local neighborhood
- y_{ijk} is pooled and subsampled layer

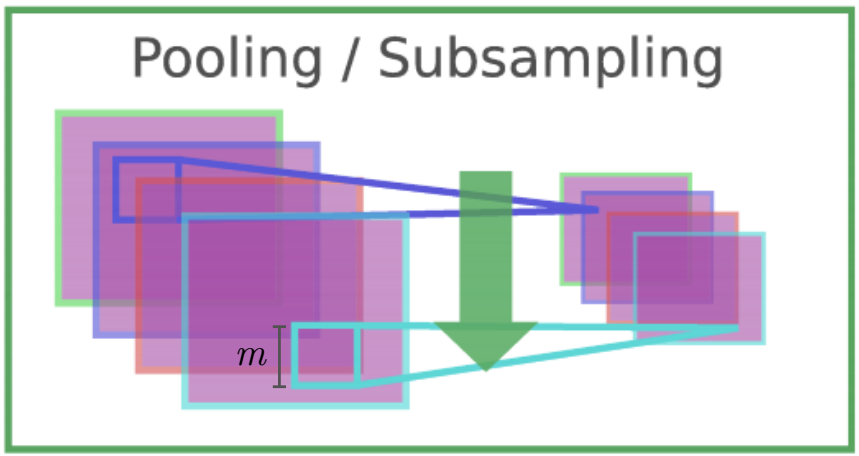
$$y_{ijk} = \max_{p,q} x_{i,j+p,k+q}$$

Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

COMPUTER VISION

Topics: pooling and subsampling

- Third idea: pool hidden units in same neighborhood
  - an alternative to "max" pooling is "average" pooling



Jarret et al. 2009

$$y_{ijk} = \frac{1}{m^2} \sum_{p,q} x_{i,j+p,k+q}$$

- $x_{i,j,k}$ is value of the i[th] feature map at position j,k
- p is vertical index in local neighborhood
- q is horizontal index in local neighborhood
- y_{ijk} is pooled and subsampled layer
- m is the neighborhood height/width

Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

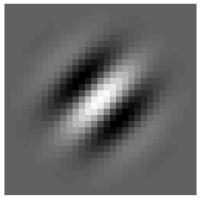
Basic idea of convolutional neural networks

- Filters are mostly local.
- Instead of using image-wide filters, use small ones over patches.
- Repeat for every patch to get a response image.
- Subsample the response image to get local invariance.

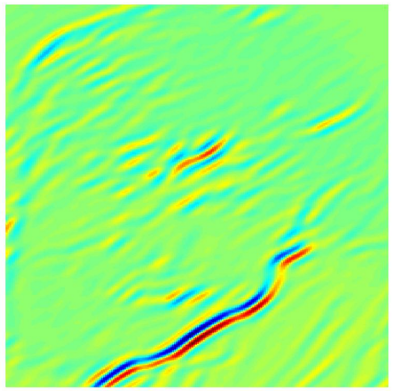
Filtering - Filter 1

Original image

Filter



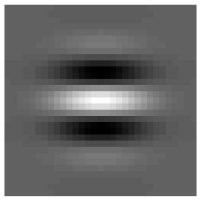
Output image



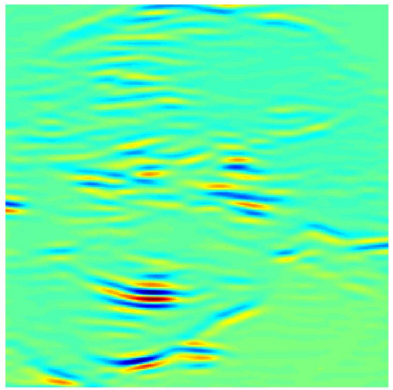
Filtering - Filter 2

Original image

Filter



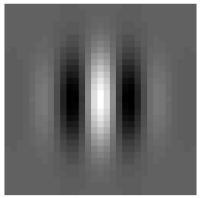
Output image



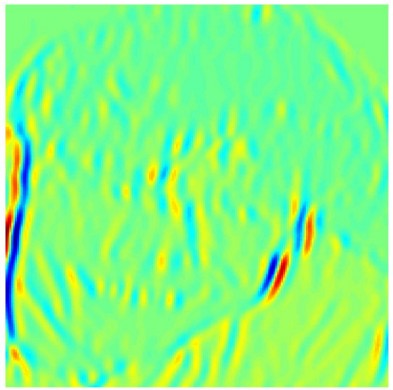
Filtering - Filter 3

Original image

Filter



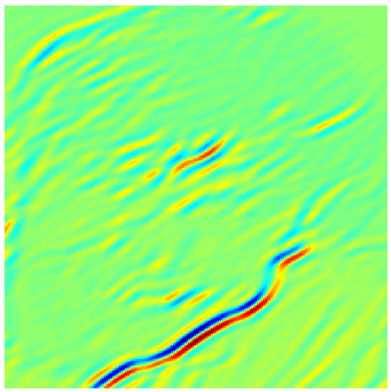
Output image



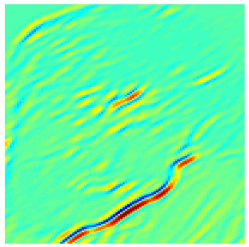
Pooling - Filter 1

Original image

Output image



Subsampled image



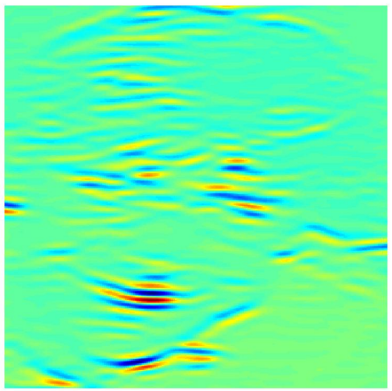
How to do 2x subsampling-pooling :

- Output image = O, subsampled image = S.
- $S_{ij} = \max_{k \text{ over window around } (2i,2j)} O_k$.

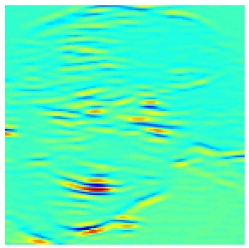
Pooling - Filter 2

Original image

Output image



Subsampled image



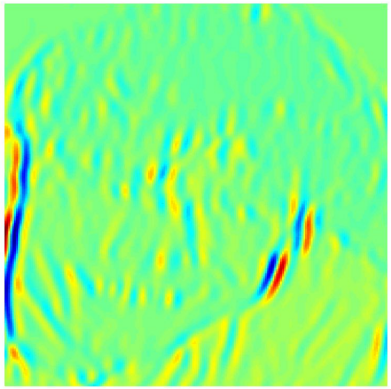
How to do 2x subsampling-pooling :

- Output image = O, subsampled image = S.
- $S_{ij} = \max_{k \text{ over window around } (2i,2j)} O_k$.

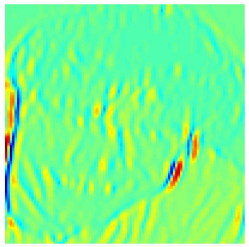
Pooling - Filter 3

Original image

Output image



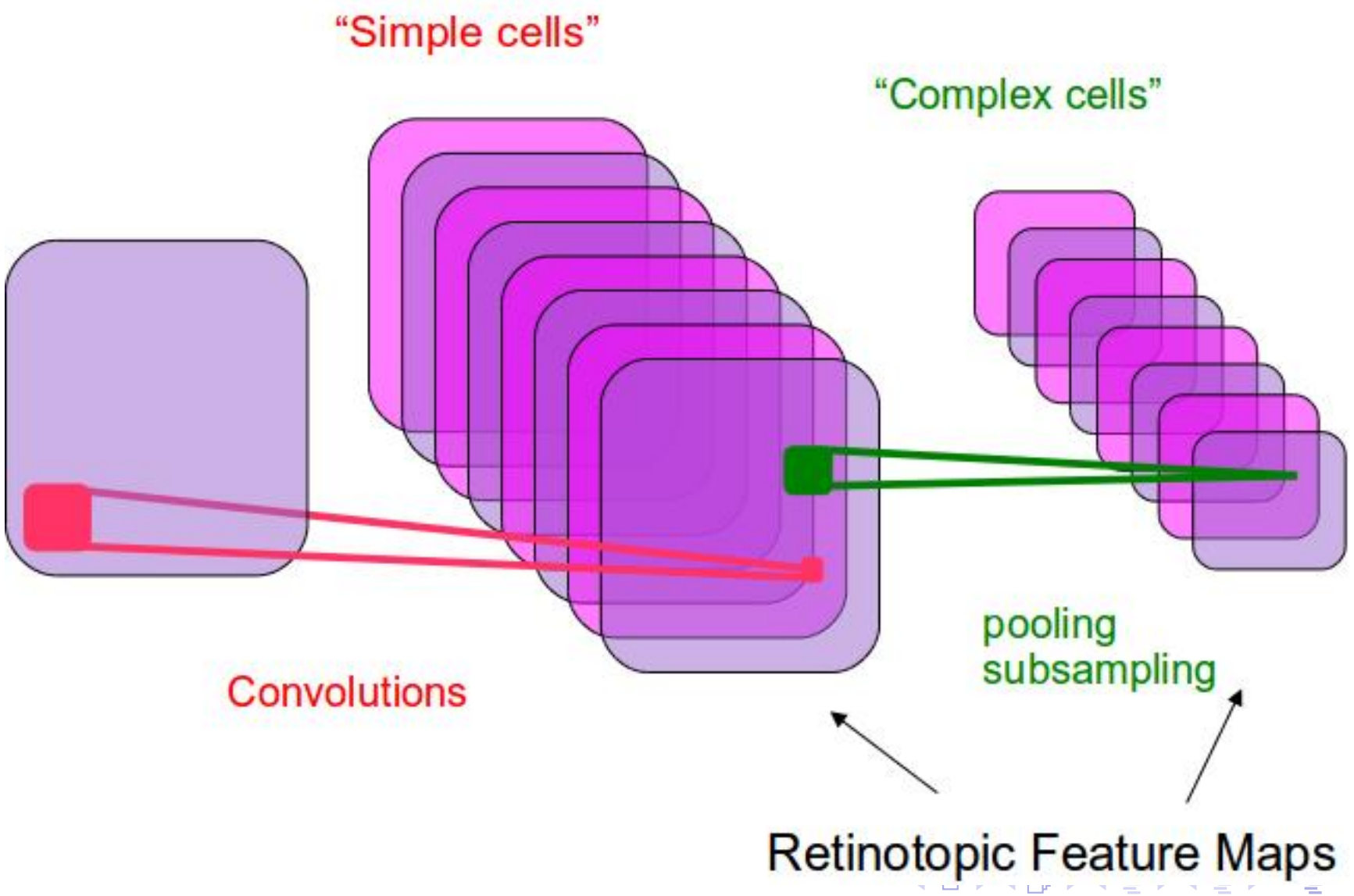
Subsampled image



How to do 2x subsampling-pooling :

- Output image = O, subsampled image = S.
- $S_{ij} = \max_{k \text{ over window around } (2i,2j)} O_k$.

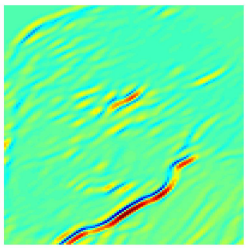
A convolutional layer

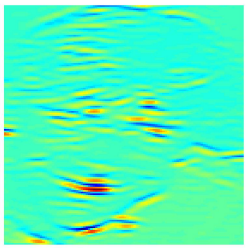


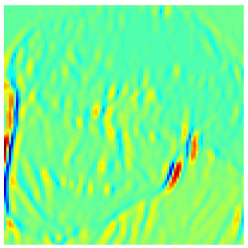
Transforming the data with a layer

Original datapoint

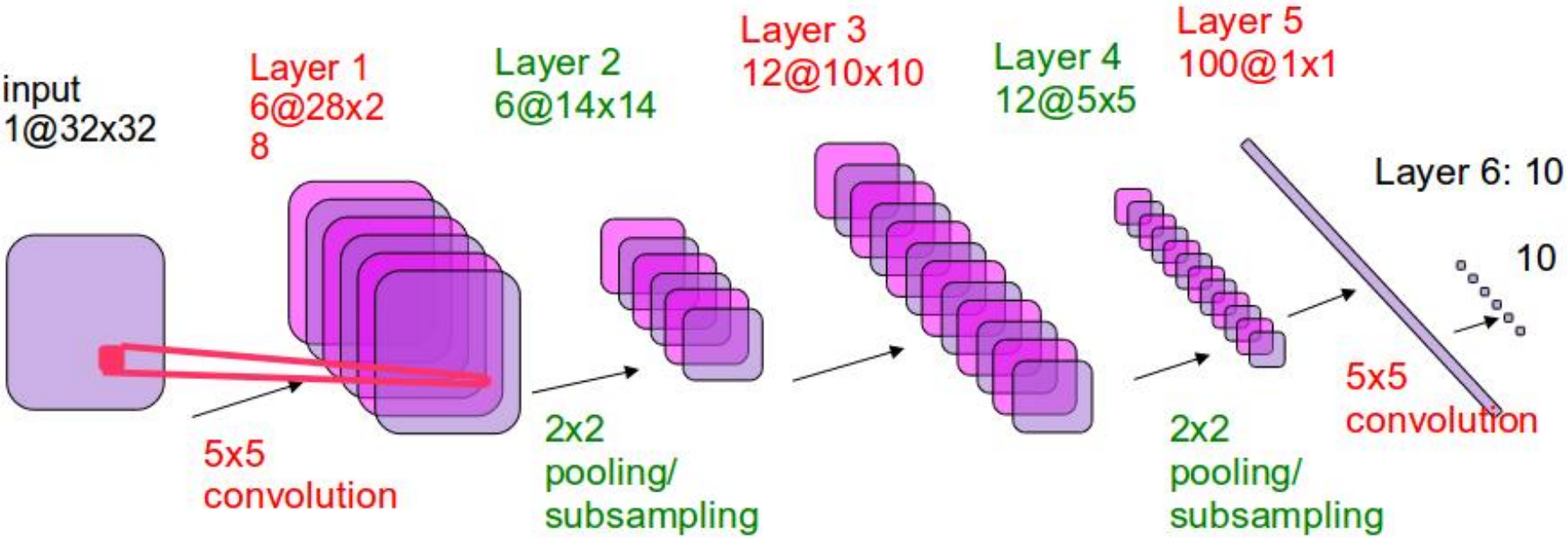
New datapoint

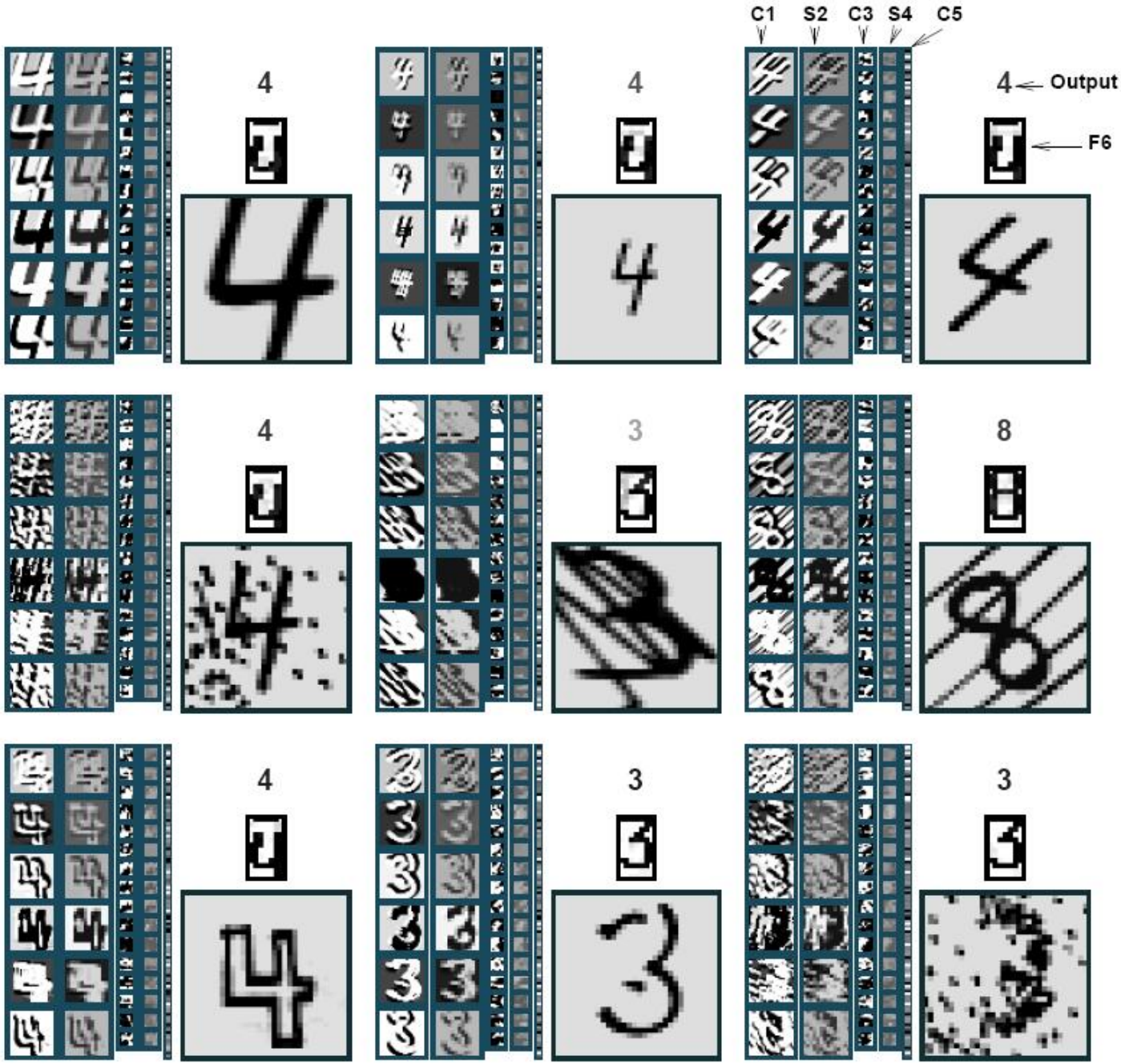






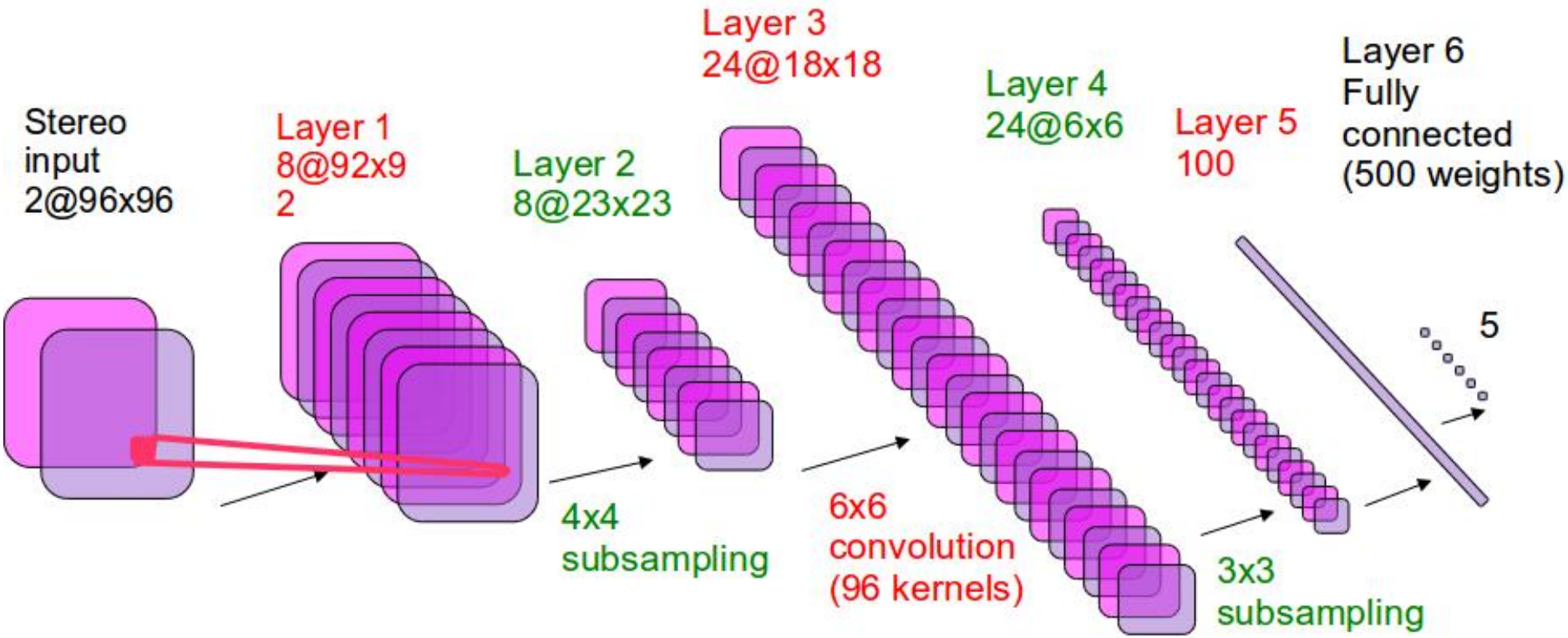
A convolutional network




C1
S2
C3
S4
C5
4
4
4
Output
F6
4
3
8
4
3
3

Face detection

Face detection



- 90,857 free parameters, 3,901,162 connections.
- The entire network is trained end-to-end (all the layers are trained simultaneously).

ConvNet summary

- With complex problems, it is hard to design features by hand.
- Neural networks circumvent this problem.
- They can be hard to train (again...).
- Convolutional neural networks use knowledge about locality in images.
- They are much easier than standard networks.
- And they are FAST (again...).

What has not been covered

- In some cases, we have lots of data, but without the labels.
- *Unsupervised* learning.
- There are techniques to use these data to get better performance.

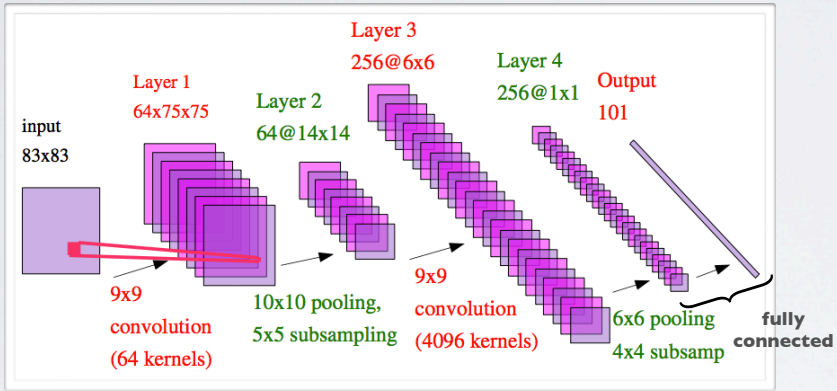
Additional techniques currently used

- Local response normalization
- Data augmentation
- Batch gradient normalization

CONVOLUTIONAL NETWORK

Topics: convolutional network

- This architecture works well for handwritten character recognition



(from Yann Lecun)

- It performs poorly on object recognition in general
  - we need to introduce other operations between

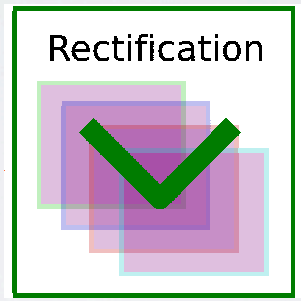
Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

CONVOLUTIONAL NETWORK

Topics: rectification layer

Jarret et al. 2009

- Rectification layer: $y_{ijk} = |x_{ijk}|$
  - introduces invariance to the sign of the unit in the previous layer
    - for instance, lose information of whether an edge is black-to-white or white-to-black



Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

CONVOLUTIONAL NETWORK

Topics: local contrast normalization layer

Jarret et al. 2009

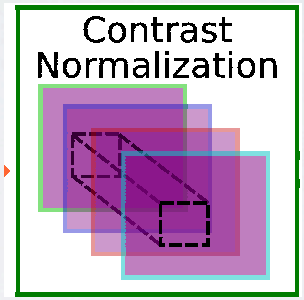
- Local contrast normalization:

$$v_{ijk} = x_{ijk} - \sum_{ipq} w_{pq} x_{i,j+p,k+q}$$

$$y_{ijk} = v_{ijk} / \max(c, \sigma_{jk})$$

$$\sigma_{jk} = (\sum_{ipq} w_{pq} v^2_{i,j+p,k+q})^{1/2}$$

$$\sum_{pq} w_{pq} = 1$$



where c is a small constant to prevent division by 0

- reduces unit's activation if neighbors are also active
- creates competition between feature maps

Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

CONVOLUTIONAL NETWORK

Topics: local contrast normalization layer

Jarret et al. 2009

- Local contrast normalization:

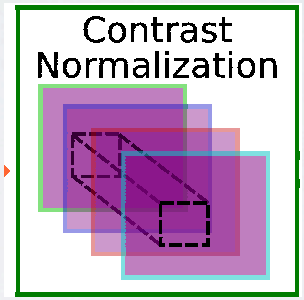
$$v_{ijk} = x_{ijk} - \sum_{ipq} w_{pq} x_{i,j+p,k+q}$$

local average

$$y_{ijk} = v_{ijk} / \max(c, \sigma_{jk})$$

$$\sigma_{jk} = (\sum_{ipq} w_{pq} v^2_{i,j+p,k+q})^{1/2}$$

$$\sum_{pq} w_{pq} = 1$$



where c is a small constant to prevent division by 0

- reduces unit's activation if neighbors are also active
- creates competition between feature maps

Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

CONVOLUTIONAL NETWORK

Topics: local contrast normalization layer

Jarret et al. 2009

- Local contrast normalization:

$$v_{ijk} = x_{ijk} - \sum_{ipq} w_{pq} x_{i,j+p,k+q}$$

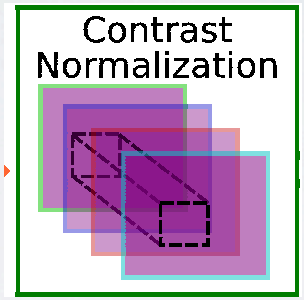
local average

$$y_{ijk} = v_{ijk} / \max(c, \sigma_{jk})$$

$$\sigma_{jk} = \left(\sum_{ipq} w_{pq} v^2_{i,j+p,k+q}\right)^{1/2}$$

local std dev.

$$\sum_{pq} w_{pq} = 1$$



where c is a small constant to prevent division by 0

- reduces unit's activation if neighbors are also active
- creates competition between feature maps

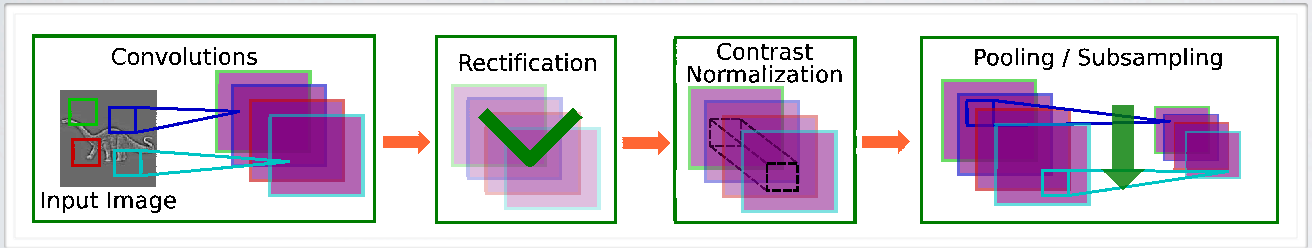
Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

CONVOLUTIONAL NETWORK

Topics: convolutional network

Jarret et al. 2009

- These operations are inserted after the convolutions and before the pooling



- Images should also be preprocessed by
  - converting to grayscale (if appropriate)
  - resizing images to 150 x 150 pixels (use zero padding for non-square images)
  - removing (intra image) mean and dividing by standard deviation of the image
  - applying local contrast normalization

Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

CONVOLUTIONAL NETWORK

Topics: initialization of parameters

Jarret et al. 2009

- Initialization of parameters:
  - can do as in regular neural network and initialize them randomly
  - can also use unsupervised pretraining approach
    - to use unsupervised neural networks we've seen so far, we have to convert pretraining as a patch-wise learning problem
      - ✓ extract patches of the same as the receptive fields of the hidden units, at random positions
      - ✓ train an unsupervised neural network (RBM, autoencoder, sparse coding) on those patches
      - ✓ use weights connecting an input patch to each hidden unit to initialize each feature map parameters
      - ✓ map images through all feature maps and repeat previous steps, for as many layers as desired
- We will compare:
  - using random initialization (R) or unsupervised pretraining (U)
  - using fine-tuning of whole network ($^+$) or only training output layer (no $^+$)

Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

CONVOLUTIONAL NETWORK

Topics: convolutional network

Jarret et al. 2009

- Results on Caltech: F_{CSG} = convolution layer
  R = rectification layer
  N = local contrast normalization layer
  P_M = max pooling layer, P_A = average pooling layer

Single Stage System: $[64.\mathbf{F}^{9\times9}_{\mathbf{CSG}} - \mathbf{R/N/P}^{5\times5}]$ **- log_reg**					
	$\mathbf{R_{abs} - N - P_A}$	$\mathbf{R_{abs} - P_A}$	$\mathbf{N - P_M}$	$\mathbf{N - P_A}$	$\mathbf{P_A}$
$\mathbf{U^+}$	54.2%	50.0%	44.3%	18.5%	14.5%
$\mathbf{R^+}$	54.8%	47.0%	38.0%	16.3%	14.3%
$\mathbf{U}$	52.2%	43.3%($\pm$1.6)	44.0%	17.2%	13.4%
$\mathbf{R}$	53.3%	31.7%	32.1%	15.3%	12.1%($\pm$2.2)
$\mathbf{G}$	52.3%				
Two Stage System: $[64.\mathbf{F}^{9\times9}_{\mathbf{CSG}} - \mathbf{R/N/P}^{5\times5}] - [256.\mathbf{F}^{9\times9}_{\mathbf{CSG}} - \mathbf{R/N/P}^{4\times4}]$ **- log_reg**					
	$\mathbf{R_{abs} - N - P_A}$	$\mathbf{R_{abs} - P_A}$	$\mathbf{N - P_M}$	$\mathbf{N - P_A}$	$\mathbf{P_A}$
$\mathbf{U^+U^+}$	65.5%	60.5%	61.0%	34.0%	32.0%
$\mathbf{R^+R^+}$	64.7%	59.5%	60.0%	31.0%	29.7%
$\mathbf{UU}$	63.7%	46.7%	56.0%	23.1%	9.1%
$\mathbf{RR}$	62.9%	33.7%($\pm$1.5)	37.6%($\pm$1.9)	19.6%	8.8%
$\mathbf{GT}$	55.8%				

Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

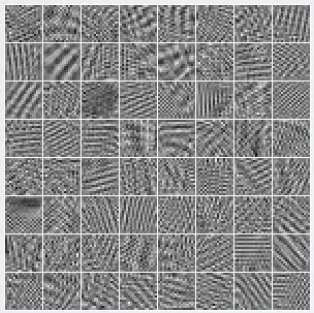
CONVOLUTIONAL NETWORK

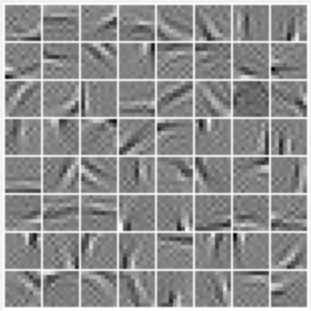
Topics: random filters

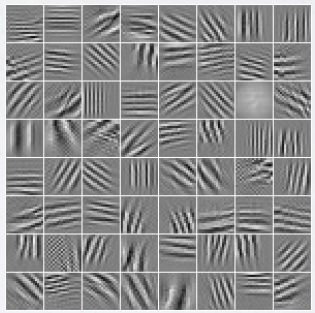
Jarret et al. 2009

- Results on Caltech:
  - random filters are surprisingly good
  - turns out that random filters give units that are still sensitive to a particular frequency
    - can analyze this by finding input which maximizes the activation of a given hidden unit (with gradient ascent applied in input space)


random filters


optimal input


learned filters


optimal input

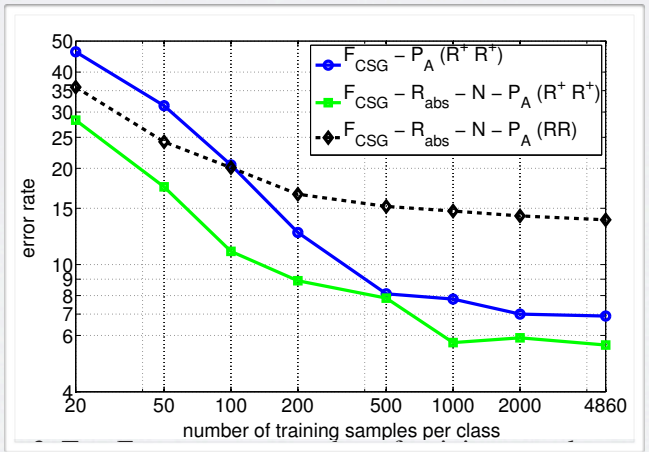
Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

CONVOLUTIONAL NETWORK

Topics: importance of architecture

Jarret et al. 2009

- Results on Caltech:
  - choice of right architecture can be more important than learning algorithm
    - the use of rectification and local contrast normalization layers is important
    - this is particularly true if little training data
- Results on NORB:
  - architecture makes less of a difference with lots of training data per class
  - random filters are also not as good



Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

INVARIANCE BY DATA SET EXPANSION

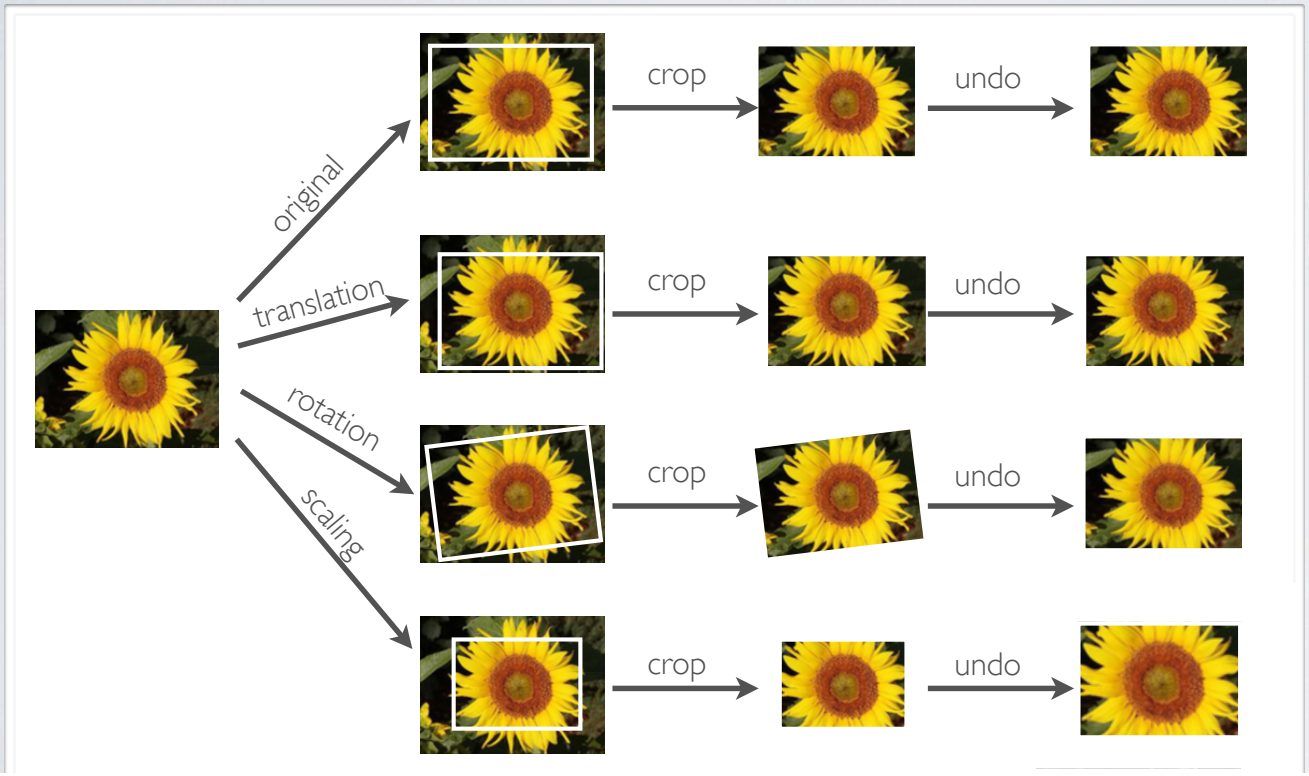
Topics: generating additional examples

- Invariances built-in in convolutional network:
  - small translations: due to convolution and max pooling
  - small illumination changes: due to local contrast normalization
- It is not invariant to other important variations such as rotations and scale changes
- However, it's easy to artificially generate data with such transformations
  - could use such data as additional training data
  - neural network will learn to be invariant to such transformations

Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

INVARIANCE BY DATA SET EXPANSION

Topics: generating additional examples

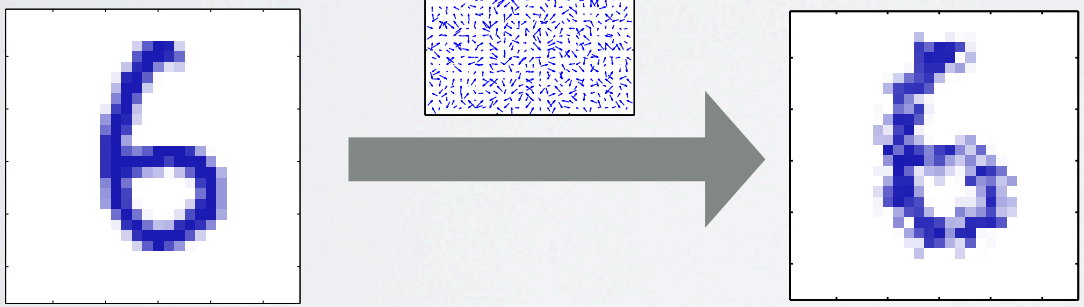


Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

INVARIANCE BY DATA SET EXPANSION

Topics: generating additional examples, distortion field

- Can add "elastic" deformations (useful in character recognition)
- We do this by applying a "distortion field" to the image
  - a distortion field specifies where to displace each pixel value



See Simard et al. for more detail

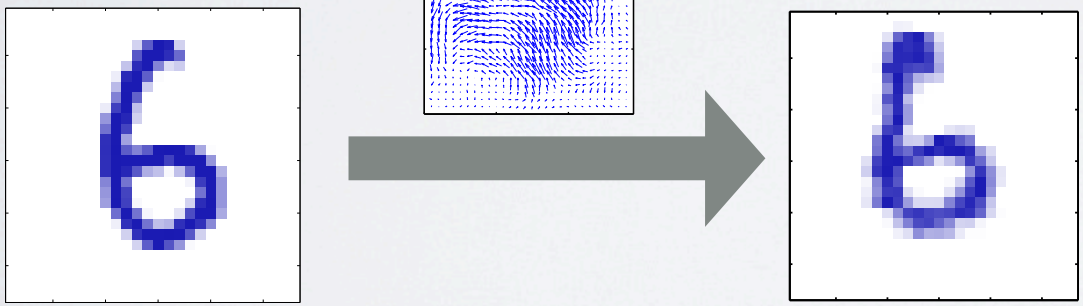
(from Bishop's book)

Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

INVARIANCE BY DATA SET EXPANSION

Topics: generating additional examples, distortion field

- Can add "elastic" deformations (useful in character recognition)
- We do this by applying a "distortion field" to the image
  - a distortion field specifies where to displace each pixel value



See Simard et al. for more detail

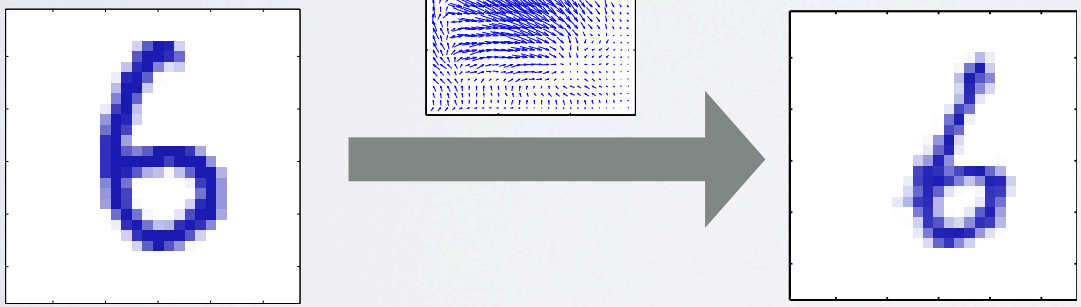
(from Bishop's book)

Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

INVARIANCE BY DATA SET EXPANSION

Topics: generating additional examples, distortion field

- Can add "elastic" deformations (useful in character recognition)
- We do this by applying a "distortion field" to the image
  - a distortion field specifies where to displace each pixel value



See Simard et al. for more detail

(from Bishop's book)

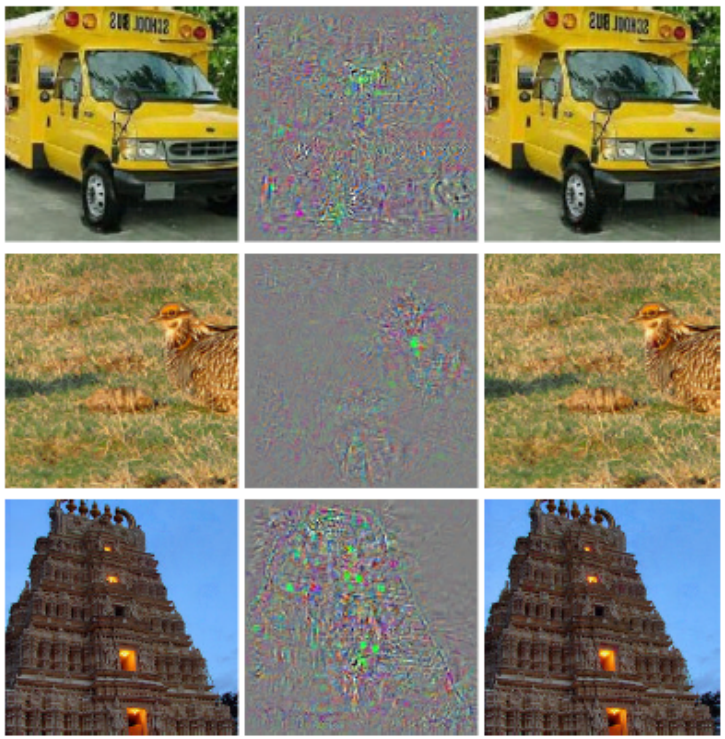
Slide taken from Hugo Larochelle : `http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html`

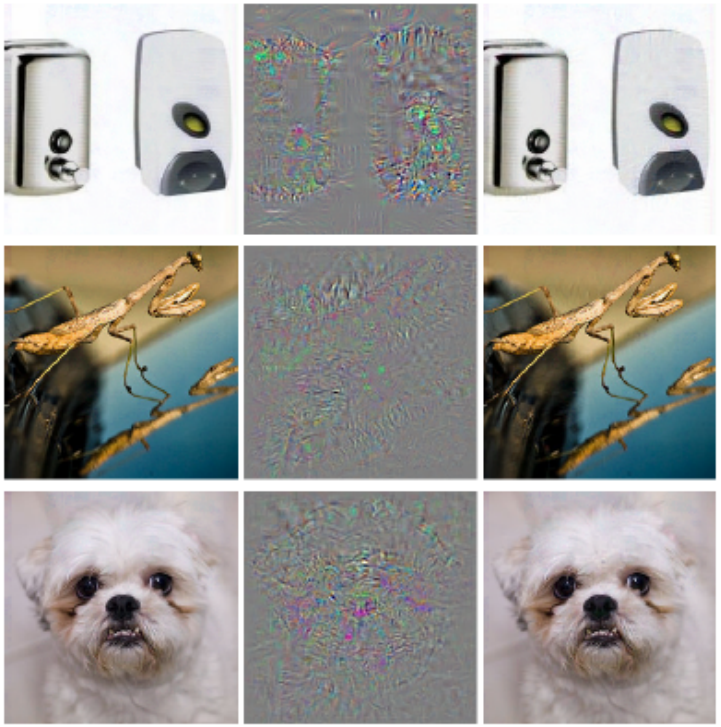
Adversarial examples

- ConvNets achieve stellar performance in object recognition
- Do they really understand what's going on in an image ?

Adversarial examples

- ConvNets achieve stellar performance in object recognition
- Do they really understand what's going on in an image ?





What I did not cover

- Algorithms for pretraining (RBMs, autoencoders)
- Generative models (DBMs)

Conclusions

- Neural networks are complex non-linear models
- Optimizing them is hard
- It is as much about theory as about engineering
- When properly tuned, they can perform wonders

Bibliography

- *Deep Learning book*, http://www-labs.iro.umontreal.ca/~bengioy/DLbook/
- *Dropout : A Simple Way to Prevent Neural Networks from Overfitting* by Srivistava et al.
- *Intriguing properties of neural networks* by Szegedy et al.
- *ImageNet Classification with Deep Convolutional Neural Networks* by Krizhevsky, Sutskever and Hinton
- *Gradient-based learning applied to document recognition* by LeCun et al.
- *Qualitatively characterizing neural network optimization problems* by Goodfellow and Vinyals
- *Sequence to Sequence Learning with Neural Networks* by Sutskever, Vinyals and Le
- *Caffe*, http://caffe.berkeleyvision.org/
- *Hugo Larochelle's MOOC*, https://www.youtube.com/playlist?list=PL6Xpj9I5qXYEcOhn7TqghAJ6NAPrNmUBH