Machine Learning

ITAM

Menu

- Complexity of a NN
  - Parameters and techniques
- Deep networks
  - Difficulties to train
  - Some solutions
  - Convolutional ANN

Complexity of a NM

- The complexity is given by
  - Number of weights (degrees of freedom)
  - Number of neurons in layeres
- The concept of deep learning referrs to the training of networks with several hidden layers (more that 2)(some say 2)

Problems in training a deep network

- Many parameters to adjust (weights)
- Layers learn at different rates
- Unstable gradient
    - As we propagate the error back the information regarding it diminishes in each layer (diminishing gradient)
    - From the backprop algorithm we can see that at each layer we multiply by the gradient of the error of the following layer (this is normally less than 1, note that the derivative of the sigmoid has a max in 0.25)
    - Also the sigmoid activation fn saturates (this reducing the info to propagate)
  - In cases where the w's are big the effect can be the opposite (expoding gradient). But this happens rearly due to regularization and because making too big makes the derivative of the sigmoid small (its bell shaped)

Some workarounds

- No definte solution but research indicates:
  - Mange the complexity of the network with regularization techniques
  - Choose the initial weights wisely
  - Use other activation functions (ReLU)
  - Modify the learning algorithm e.g.:
    - Momentum adds a fraction of the change in weight of the previous iteration to the new weight (yet another parameter)
    - Different learning rate per neuron and change with time
  - More data!, More computing power! (not practical 6 years ago)
  - Explore other network topologies

Network complexity

- We can control the complexity with :
  - Regularization
    - Lasso (L1)
    - Ridge (L2)
    - Elastic net (linear combination of the above)

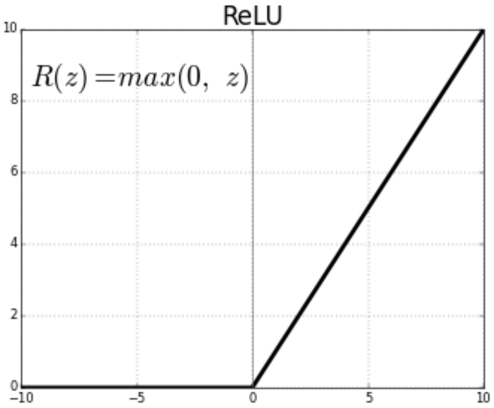
Network complexity

- Drop out
  - During training, for each example, each neuron is chosen with probability p (usually 0.5) to be ignored (the effect is aprox half participate)
  - During test, all neurons participate but ther weights are scaled with p
  - This technique can be understood as having multiple trained networks and averaging their output. The effect is to reduce variance. We will discuss this further when we talk about ensamble learning

Network complexity

- Early stopping
  - Stop iterating (training) when the validation error from one iteration to another starts to increase (preserve the weights of the previous iteration)

Other activation functions



- ReLu (Rectified linear unit)
  - Max(0,w^Tx)
  - Helps with the diminishing gradient problem. Only saturates on one side.
  - Approximately half will be deactibated at the beginning of training (using random weights)

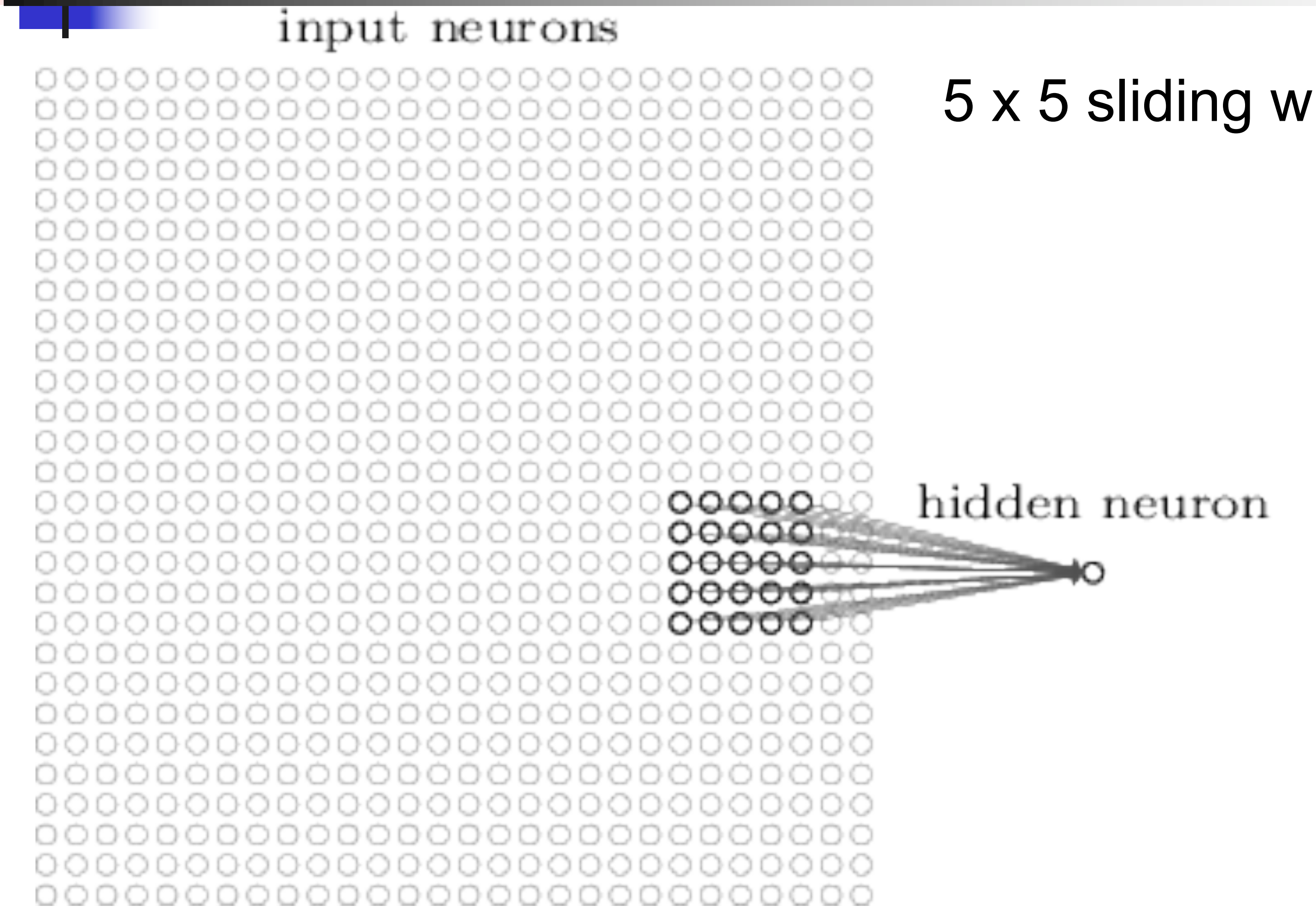
Explore other topologies

- Convolutional ANN
  - Particularly good for images
  - They take into account the spacial structure of an image
    - One less thing to learn!
  - Three important concepts
    - Localization(Local receptive field)
    - Shared weights
    - Pooling (aggregation)

Localization

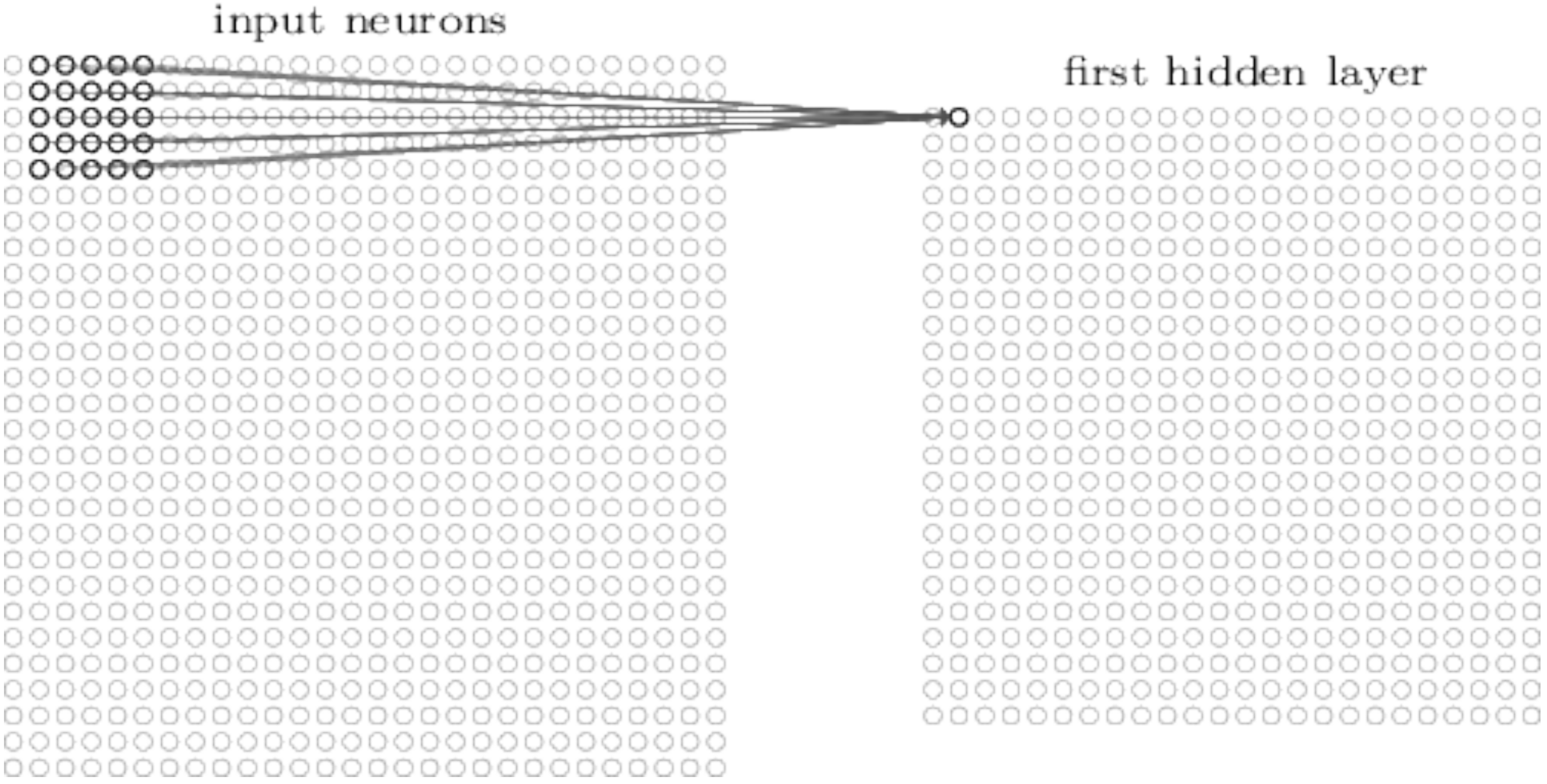
- Only some neurons of a layer are connect to some neurons of the next. No longer an all to all relationship
  - We make groups or related neurons
    - For images we can organize them in 3D groups: height width and channel (color). The idea is that neurons that have inputs that are close together in space should be connected together in the next layer.
  - The output of this layer is the input of the next

Localization (images from Michael Nielsen http://neuralnetworksanddeeplearning.com/index.html)



5 x 5 sliding window

Localization

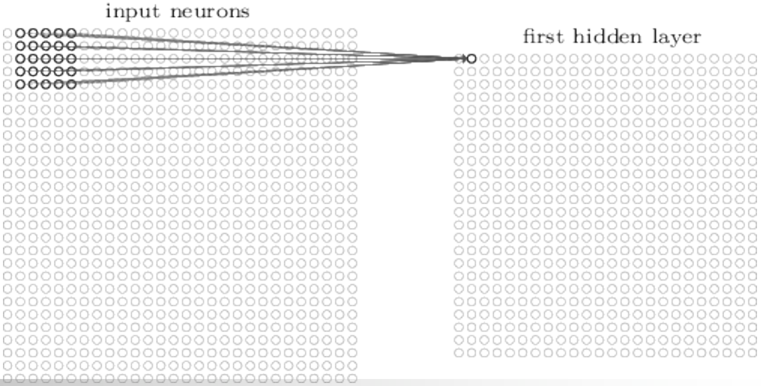


The size of the slide step is called *stride length*

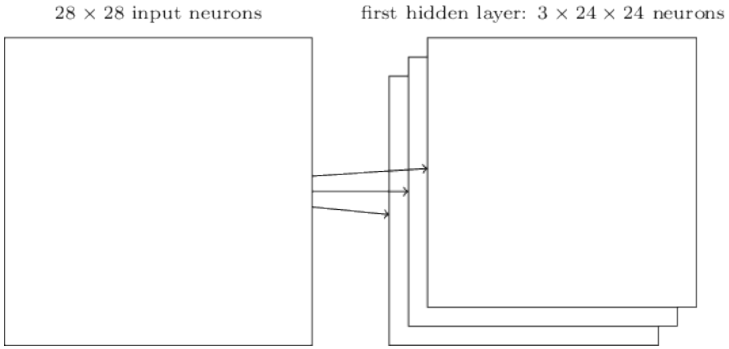
Localization

- If we have n * n neurons in the input layer and a window of m*m with stride length s
  - How many neurons in the next layer?
    - k horizontal and k vertical windows
    - (1 + ceil((n-m)/s)) x (1 + ceil((n-m)/s))
    - For12 x 12, m=3, stride s= 2
      - 1+ ceil(9/2)= 6
    - If you go over you can zero-pad

Shared weights



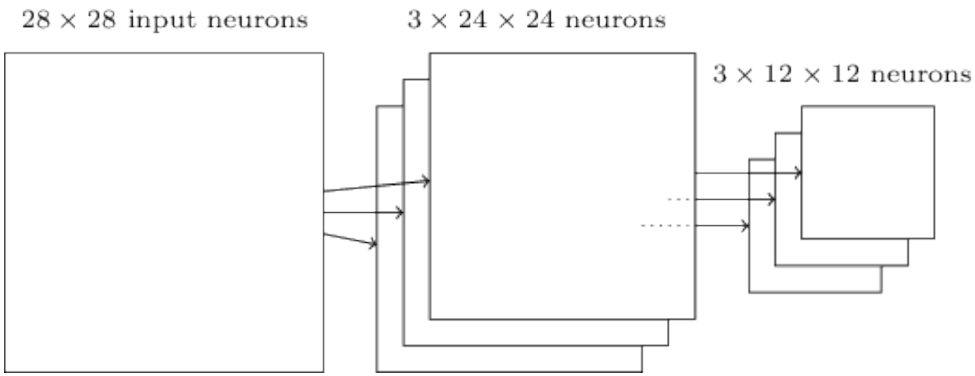
- Each neuron in an intermediate layer is connected to m x m of the previous layer
  - It has m x m weights plus the bias (w_0)
- Each of the k x k neurons in this layer is going to share the same weights!
  - They all will detect the same pattern but regardes of its location (thats the idea, which makes sense in images and time series)
  - This is known as a kernel or filter



Parallel layers

- Each one of this layers (filter) detects specializes in some feature
- We need to include several of these, all connected to previous layer

Pooling

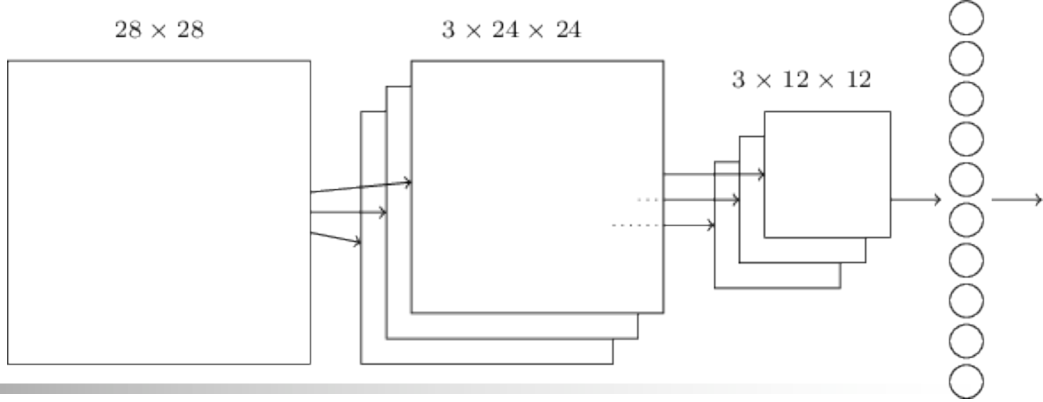


- This layer is placed right after the filters
  - Sililarly to localization it takes a sliding window over the previous layer and produces a result
  - It reduces the number of outputs further
    - If the window is 2 x 2 with strind 2 the number of neurons is reduced in half
- The operation it performs is some sort of aggregation or summary over its window
  - Max value (max-pooling)
  - L2 norm
  - Average

Topology of Convolutional ANN

- Input layer
- Convolutional layers (several)
  - Filters and max pooling
- Output layer

Output layer



- After the last convolutional layer we usually have a traditional fully conected network.
- The resulting network has a feed forward strucuture and we can basically use the backpropagation algorithm we saw in class
  - Need to work out the discontinuity of ReLu (if you use it)
  - And how to differentiate the pooling layers among other details

Other architectures

- Boltzman machines
- Deep belief networks
- LTSM (long short-term memory)
- Check out google’s inception network
- https://github.com/google/inception

Exercises

- Design an experiment to visualize the learning slowdown problem in deep networks. Experiment with different transfer functions
- Deep Minst for experts in TensorFlow

Exercises

- Create an account on Kaggle
- Look for a interesting image data set. Try a convolutional ANN