Deep Learning

Learning High-degree Non-linear Models: Enabling factors, tips and tricks

Lecturer: [illegible] (Cybercom)

Problems

- Overlearning
- Getting stuck in local minima
- Exploding/Vanishing gradients

Solutions

1. Weight sharing - less parameters
2. Unsupervised pre-training
   - lots of data
3. Non-linear activation functions
4. Drop-out
5. Gradient clipping

Parallel/distributed computing

Reservoir Computing

[illegible]

[illegible]

[illegible]

Deep Learning

Learning High-degree Non-linear Models: Enabling factors, tips and tricks

Lecturer: AI Generalist, Tero Keski-Valkama (Cybercom)

Problems

– Overlearning
– Getting stuck in local minima
– Exploding/diminishing gradients

Solutions

1. Weight-sharing – less parameters
2. Unsupervised pre-training
    – lots of data
3. Near-linear activation functions
4. Drop-out
5. Gradient clipping

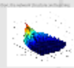
Bonus: Reservoir computing

Reservoir Computing

– A common name for methods that utilize a large, pre-initialized network, and then train a simple linear model with this large reservoir.
– For example: Echo State Networks, Extreme Learning Machine.
– Fast to train, work very well in domains where semantic depth is not required.

What Did We Learn?

- To make deep neural networks actually do anything useful, you need a bag of tricks:
    - [illegible]
    - Pre-training with [illegible]
    - Nonlinear activation functions
    - Drop-out
    - Gradient clipping
- [illegible]
- [illegible]
    - [illegible]
    - [illegible]



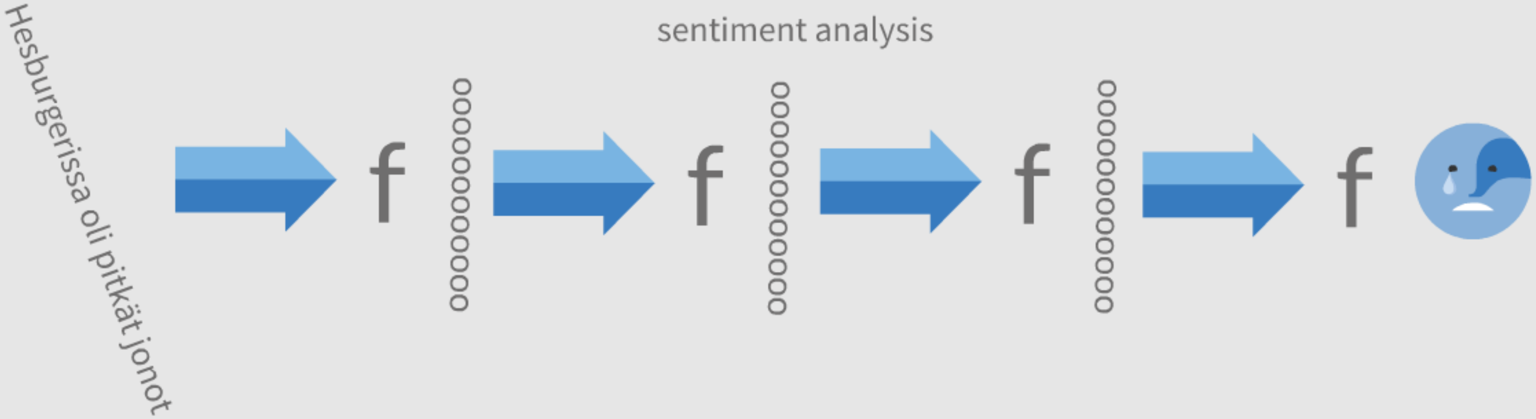
Deep Learning

Learning High-degree Non-linear Models: Enabling factors, tips and tricks

Lecturer: AI Generalist, Tero Keski-Valkama (Cybercom)

A quick re-cap

- Deep learning refers to deep neural networks. It took about 10 years for neural network technology to surpass the issues they stalled with in the 1990s. In 00s, if an academic paper mentioned neural networks, it was less likely to be published than if not.
- 1990s networks were shallow, and hence relatively useless: It requires a bunch of non-trivial tricks to make deeper networks possible. This presentation will present the most important of these tricks.
- Neural networks are just complex non-linear models with lots of parameters. They are formed by layers of neurons, successive operations with an a linear combination of inputs from the previous layer and an activation function.
- In general, you always need a training set, a test set and a validation set. Typically you would divide your data into three parts, train the system with the training set, test if the system overfits with the test set, and finally for the final system you can validate that your metaparameters didn't just learn the test set using the validation set.
- Underfitting = high bias, overfitting = high variance



Prezi

Training Neural Networks

- Networks can be trained in a supervised fashion, with target outputs(labels), using backpropagation of error. The output error is known, and it is propagated backwards, layer by layer, estimating an error for each weight parameter. The weights are then updated using this error multiplied by the learning factor (<<1). Lots of learning steps makes the network learn the target function (input -> target output)
- Non-linearities (activation functions) between the layers make the model interesting, compared to other statistical models.
- When the layers are getting smaller than the previous layer, the information is being compressed (or filtered), and the layer activations represent a higher abstraction level, a higher semantic level information about the signal.
- Unsupervised learning (no target output) can be done using backpropagation and autoassociativity (input -> input), (or prediction: delayed input -> input), or using Deep Belief Nets and Contrastive Divergence (minimizes the energy for generating real data-related distributions while maximizing the energy for "confabulations") for pairs of subsequent layers one by one.
- Metaoptimizing the learning parameters and neural network structure is always required.
- A good platform to use for neural network experimentation is Google's TensorFlow, based on Python.

Backpropagation Supervised Non-energy-based Methods	+	Contrastive Divergence Unsupervised Energy-based Methods

http://deeplearning.cs.cmu.edu/notes/yuxiong_CD.pdf

Problems

– Overlearning
– Getting stuck in local minima
– Exploding/diminishing gradients

Solutions

1. Weight-sharing – less parameters
2. Unsupervised pre-training
   – lots of data
3. Near-linear activation functions
4. Drop-out
5. Gradient clipping

Bonus: Reservoir computing

Weight sharing / Regularization

- Reducing the number of parameters or degrees of freedom of the model is regularization. Regularization prevents overfitting.
- In Convolutional Neural Networks, the weights are identical for each window, and window outputs are max-pooled to the next layer. This exploits the translational symmetry of the domain, and is often used for images.
- Regularization can also be done by introducing a loss term for weights, L2 norm is often used.

Problems

- Overlearning
- Getting stuck in local minima
- Exploding/diminishing gradients

Solutions

1. Weight-sharing – less parameters
2. Unsupervised pre-training
   - lots of data
3. Near-linear activation functions
4. Drop-out
5. Gradient clipping

Bonus: Reservoir computing

Unsupervised pre-training

- Unsupervised pre-training makes use of a lot of unlabeled data, learning the overall structure of the domain, which can be fine-tuned with backpropagation for specific labels at a later stage. More data, less labels.
- This could be thought as "pre-training as regularization", or approaching neural reinforcement learning.
- Free pre-trained models are available from Google and Oxford University. Mostly convolutional nets trained on images and video.
- The models are taught using for example greedy DBN Contrastive Divergence or other autoencoding methods.

http://www.jmlr.org/papers/volume11/erhan10a/erhan10a.pdf
http://www.vlfeat.org/matconvnet/pretrained/

Problems

- – Overlearning
- – Getting stuck in local minima
- – Exploding/diminishing gradients

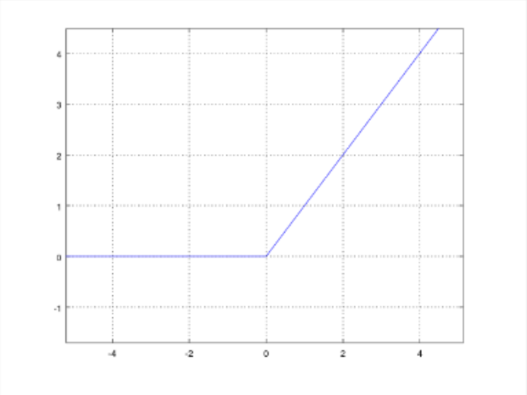
Solutions

1. Weight-sharing – less parameters
2. Unsupervised pre-training
   – lots of data
3. Near-linear activation functions
4. Drop-out
5. Gradient clipping

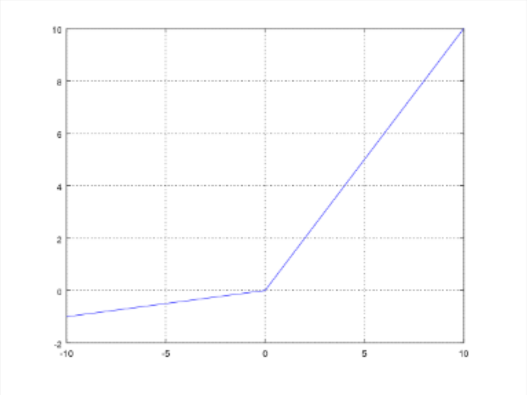
Bonus: Reservoir computing

Near-linear Activation Functions

- Deep neural networks are effective because of non-linear activation functions
  – Stacking linear layers only reduces to a single linear layer.
- However, non-linearities are challenging to learn. Minimizing the non-linearity has been shown to be really useful. Deep neural networks can be trained without unsupervised pre-training if rectifier activation functions are used. (otherwise cannot)



Rectifier: max(0,x)



Leaky rectifier: max(0.1*x,x)

Problems

– Overlearning
– Getting stuck in local minima
– Exploding/diminishing gradients

Solutions

1. Weight-sharing – less parameters
2. Unsupervised pre-training
   – lots of data
3. Near-linear activation functions
4. Drop-out
5. Gradient clipping

Bonus: Reservoir computing

Dropout

- Dropout is a very effective strategy for preventing overfitting.
- In practice it means randomly dropping out for example half of the internal representation neurons and their output weights for each training step, and scaling the outputs accordingly.
- This prevents neurons from co-adapting with each other, and learn features more independently. I.e. one neuron learns one feature, not multiple neurons learning one feature.
- It is a very strong method against overfitting, and should be almost always used.
- Analogous to Random Forests.

https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf

Problems

– Overlearning

– Getting stuck in local minima

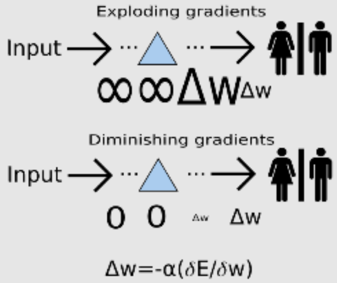
– Exploding/diminishing gradients

Solutions

1. Weight-sharing – less parameters
2. Unsupervised pre-training
   – lots of data
3. Near-linear activation functions
4. Drop-out
5. Gradient clipping

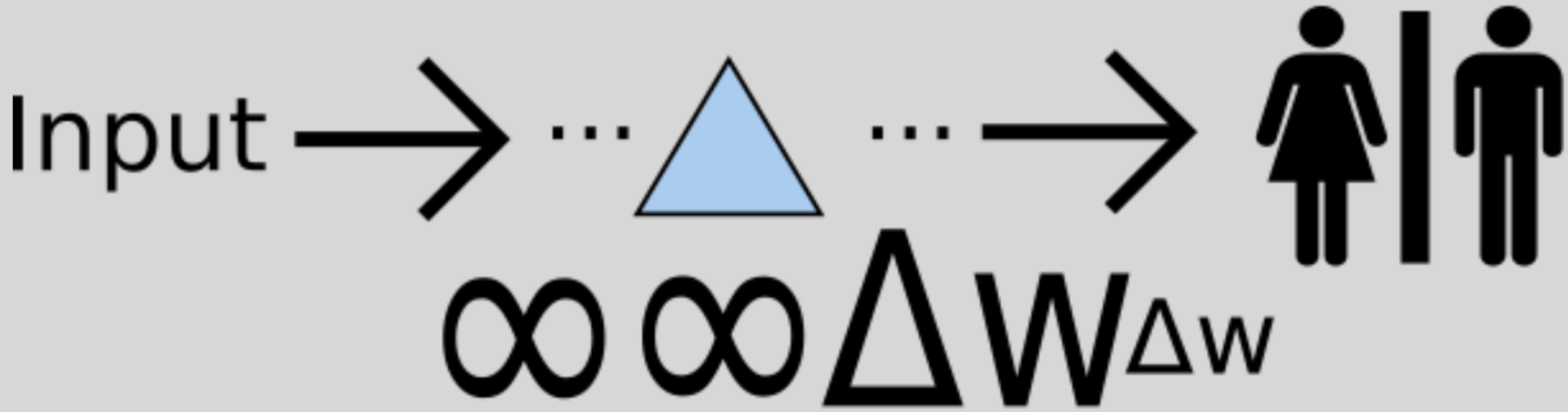
Bonus: Reservoir computing

Gradient Clipping

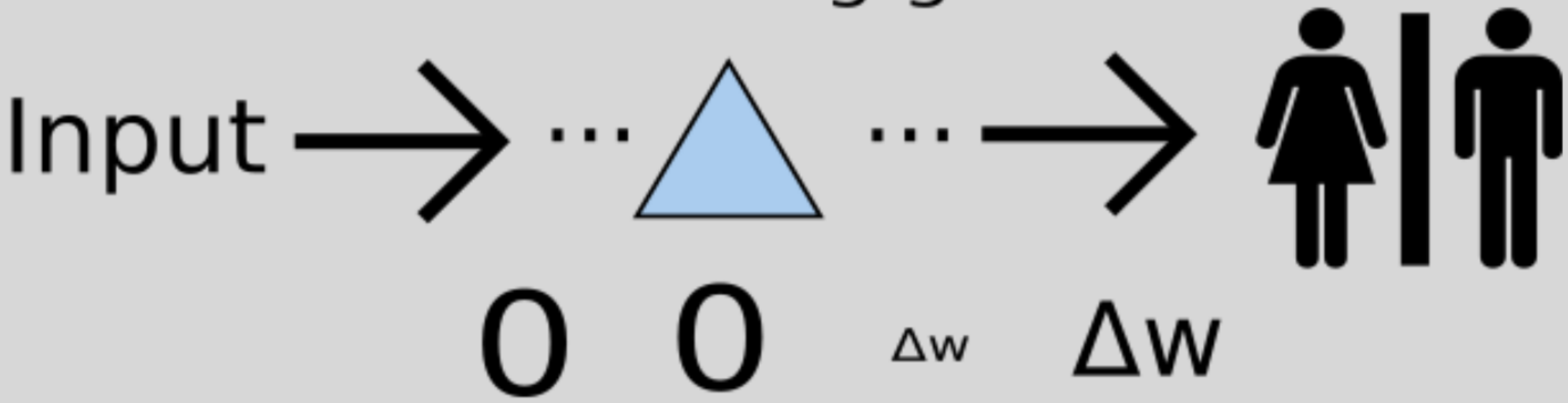


- Gradient explosion happens in backpropagation when the squared error gradients are amplified through activation functions of multiple layers. This typically causes infinities and divergence. Limited activation functions (sigmoid, tanh) are prone to this behavior.
- Gradients are often clipped to some maximum absolute value to prevent infinities.
- Gradients can also diminish to zero, when a small change in internal representation means a too large change in output. If a gradient goes to zero, it means it continues to be zero in backpropagation towards input. Adding noise might help the system in escaping local plateaus.
- Microsoft uses special one-bit gradients only signifying the direction of change, which is always non-zero.

Exploding gradients

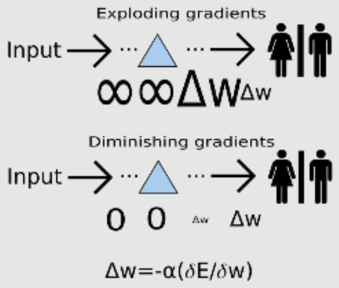


Diminishing gradients



$$\Delta w = -\alpha(\delta E / \delta w)$$

Gradient Clipping



- Gradient explosion happens in backpropagation when the squared error gradients are amplified through activation functions of multiple layers. This typically causes infinities and divergence. Limited activation functions (sigmoid, tanh) are prone to this behavior.
- Gradients are often clipped to some maximum absolute value to prevent infinities.
- Gradients can also diminish to zero, when a small change in internal representation means a too large change in output. If a gradient goes to zero, it means it continues to be zero in backpropagation towards input. Adding noise might help the system in escaping local plateaus.
- Microsoft uses special one-bit gradients only signifying the direction of change, which is always non-zero.

Reservoir Computing

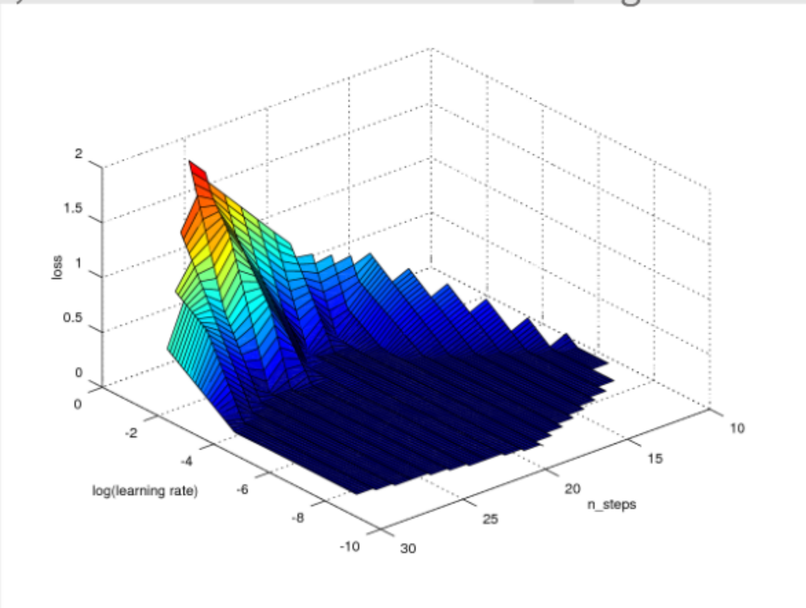
– A common name for methods that utilize a large, pre-initialized network, and then train a simple linear model with this large reservoir.

– For example: Echo-State Networks, Extreme Learning Machine.

– Fast to train, work very well in domains where semantic depth is not required.

What Did We Learn?

- To make deep neural networks actually do anything useful, you need a bag of tricks:
    - Regularization: Managing the number of parameters
    - Pre-training with massive datasets
    - Rectifier activation functions
    - Dropout
    - Gradient clipping
- These should be always used where relevant. In addition, the network structure and learning parameters should always be meta-optimized.
- The next lectures will be about:
    - Data encoding / representation
    - TensorFlow and meta-optimization



Deep Learning

Learning High-degree Non-linear Models: Enabling factors, tips and tricks

Lecturer: AI Generalist, Tero Keski-Valkama (Cybercom)

Problems

– Overlearning
– Getting stuck in local minima
– Exploding/diminishing gradients

Solutions

1. Weight-sharing – less parameters
2. Unsupervised pre-training
   – lots of data
3. Near-linear activation functions
4. Drop-out
5. Gradient clipping

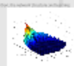
Bonus: Reservoir computing

Reservoir Computing

– A common name for methods that utilize a large, pre-initialized network, and then train a simple linear model with this large reservoir.
– For example: Echo State Networks, Extreme Learning Machine.
– Fast to train, work very well in domains where semantic depth is not required.

What Did We Learn?

- To make deep neural networks actually do anything useful, you need a bag of tricks:
  - [illegible]
  - Pre-training with [illegible]
  - Near-linear activation functions
  - Drop-out
  - Gradient clipping
- [illegible]
- [illegible]
  - [illegible]
  - [illegible]



Prezi