CPSC 540: Machine Learning

Deep Structured Models

Mark Schmidt

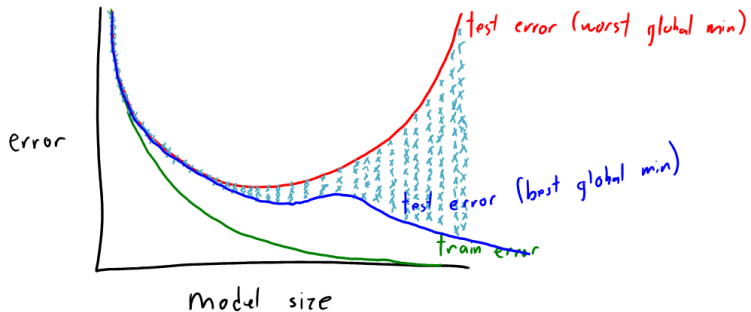
University of British Columbia

Winter 2020

Last Time: Neural Network Tricks and Double Descent Curves

- We overviewed some standard methods to improve training of neural networks.
  - Data transformations, ReLU, batch normalization, CNNs.

- We discussed some explanations for why large neural networks generalized.
  - Implicit regularization of using SGD to choose between global optima.

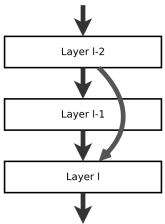


- Today: some more tricks, connection to CRFs, and deep structured predicction.

"Residual" Networks (ResNets)

- Suppose we fit a deep neural network to a linearly-separable dataset.
  - Original features x are sufficient to perfectly classify training data.
  - For a deep neural network to work, each layer needs to preserve information in x.
    - You might be "wasting" parameters just re-representing data from previous layers.

- Consider residual networks:



https://en.wikipedia.org/wiki/Residual_neural_network

- Take a previous (non-transformed) layer as input to current layer.
  - Also called "skip connections" or "highway networks".

"Residual" Networks (ResNets)

- ResNets seemingly make learning easier:
  - You can "default" to just copying the previous layer.
  - The non-linear transform is only learning how to modify the input.
    - "Fitting the residual".
- This was a key idea behind first methods that used 100+ layers.
  - Easy for information about x to reach y through huge number of layers.
  - Won all tasks in ImageNet 2015 competition.
  - Evidence that biological networks have skip connections like this.
- Dense networks (DenseNets): connect to many previous layers.
  - Basically gets rid of vanishing gradient issue.

DenseNets

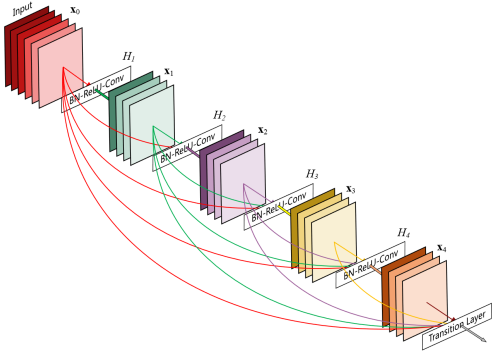


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

https://arxiv.org/pdf/1608.06993.pdf

Pre-Training

- Suppose you want to solve a new object detection task.
  - Recognize a particular abnormality in radiology images.
- You only have a few labeled images, so is deep learning useless?

- An important concept in many computer vision applications is pre-training.
  - Learn new concepts faster by modifying networks trained on millions of images.
  - Uses that many "features" are common between tasks (edges, corners, shapes,...).

- Typical setup:
  - Take a network trained on ImageNet (typically VGG or ResNet).
  - Re-train the last layer to solve your problem (convex with usual losses).

Outline

Backpropagation as Message-Passing

- Computing the gradient in neural networks is called backpropagation.
  - Derived from the chain rule and memoization of repeated quantities.
- We're going to view backpropagation as a message-passing algorithm.
- Key advantages of this view:
  - It's easy to handle different graph structures.
  - It's easy to handle different non-linear transformations.
  - It's easy to handle multiple outputs (as in structured prediction).
  - It's easy to add non-deterministic parts and combine with other graphical models.

Backpropagation Forward Pass

- Consider computing the output of a neural network for an example i,

$$\begin{aligned} y^i &= v^T h(W^3 h(W^2 h(W^1 x^i))) \\ &= \sum_{c=1}^{k} v_c h\left(\sum_{c'=1}^{k} W^3_{c'c} h\left(\sum_{c''=1}^{k} W^2_{c''c'} h\left(\sum_{j=1}^{d} W^1_{c''j} x^i_j\right)\right)\right). \end{aligned}$$

  where we've assume that all hidden layers have k values.
  - In the second line, the h functions are single-input single-output.

- The nested sum structure is similar to our message-passing structures.

- However, it's easier because it's deterministic: no random variables to sum over.
  - The messages will be scalars rather than functions.

Backpropagation Forward Pass

- Forward propagation through neural network as message passing:

$$
\begin{aligned}
y^i &= \sum_{c=1}^{k} v_c h\left(\sum_{c'=1}^{k} W^3_{c'c} h\left(\sum_{c''=1}^{k} W^2_{c''c'} h\left(\sum_{j=1}^{d} W^1_{c''j} x^i_j\right)\right)\right) \\
&= \sum_{c=1}^{k} v_c h\left(\sum_{c'=1}^{k} W^3_{c'c} h\left(\sum_{c''=1}^{k} W^2_{c''c'} h(M_{c''})\right)\right) \\
&= \sum_{c=1}^{k} v_c h\left(\sum_{c'=1}^{k} W^3_{c'c} h(M_{c'})\right) \\
&= \sum_{c=1}^{k} v_c h(M_c) \\
&= M_y,
\end{aligned}
$$

where intermediate messages are the z values.

Backpropagation Backward Pass

- The backpropagation backward pass computes the partial derivatives.
  - For a loss f, the partial derivatives in the last layer have the form

$$\frac{\partial f}{\partial v_c} = z_c^{i3} f'(v^T h(W^3 h(W^2 h(W^1 x^i)))),$$

where

$$z_{c'}^{i3} = h\left(\sum_{c'=1}^{k} W_{c'c}^3 h\left(\sum_{c''=1}^{k} W_{c''c'}^2 h\left(\sum_{j=1}^{d} W_{c''j}^1 x_j^i\right)\right)\right).$$

  - Written in terms of messages it simplifies to

$$\frac{\partial f}{\partial v_c} = h(M_c) f'(M_y).$$

Backpropagation Backward Pass

- In terms of forward messages, the partial derivatives have the forms:

$$\frac{\partial f}{\partial v_c} = h(M_c) f'(M_y),$$

$$\frac{\partial f}{\partial W^3_{c'c}} = h(M_{c'}) h'(M_c) w_c f'(M_y),$$

$$\frac{\partial f}{\partial W^2_{c''c'}} = h(M_{c''}) h'(M_{c'}) \sum_{c=1}^{k} W^3_{c'c} h'(M_c) w_c f'(M_y),$$

$$\frac{\partial f}{\partial W^1_{jc''}} = h(M_j) h'(M_{c''}) \sum_{c'=1}^{k} W^2_{c''c'} h'(M_{c'}) \sum_{c=1}^{k} W^3_{c'c} h'(M_c) w_c f'(M_y),$$

which are ugly but notice all the repeated calculations.

Backpropagation Backward Pass

- It's again simpler using appropriate messages

$$\frac{\partial f}{\partial v_c} = h(M_c) f'(M_y),$$

$$\frac{\partial f}{\partial W^3_{c'c}} = h(M_{c'}) h'(M_c) w_c V_y,$$

$$\frac{\partial f}{\partial W^2_{c''c'}} = h(M_{c''}) h'(M_{c'}) \sum_{c=1}^{k} W^3_{c'c} V_c,$$

$$\frac{\partial f}{\partial W^1_{jc''}} = h(M_j) h'(M_{c''}) \sum_{c'=1}^{k} W^2_{c''c'} V_{c'},$$

where $M_j = x_j$.

Backpropagation as Message-Passing

- The general forward message for child c with parents p and weights W is

$$M_c = \sum_p W_{cp} h(M_p),$$

  which computes weighted combination of non-linearly transformed parents.
  - In the first layer we don't apply h to x.
- The general backward message from child c to *all* its parents is

$$V_c = h'(M_c) \sum_{c'} W_{cc'} V_{c'},$$

  which weights the "grandchildren's gradients".
  - In the last layer we use f instead of h.
- The gradient of W_{cp} is $h(M_c)V_p$, which works for general graphs.

Automatic Differentiation

- Automatic differentiation:
  - Input is code that computes a function value.
  - Output is code computing is one or more derivatives of the function.

- Forward-mode automatic differentiation:
  - Computes a directional derivative for cost of evaluating function.
    - So computing gradient would be d-times more expensive than function.
  - Low memory requirements.
  - Most useful for evaluating Hessian-vector products, $\nabla^2 f(w)d$.

- sReverse-mode automatic differentiation:
  - Computes gradient for cost of evaluating function.
  - But high memory requirements: need to store intermediate calculations.
    - Backpropagation is (essentially) a special case.
- Reverse-mode is replacing "gradient by hand" (less time-consuming/bug-prone).

Combining Neural Networks and CRFs

- Last time we saw conditional random fields like

$$p(y \mid x) \propto \exp\left(\sum_{c=1}^{k} y_c v^T x_c + \sum_{(c,c') \in E} y_c y_{c'} w\right),$$

  which can use logistic regression at each location c and Ising dependence on y_c.

- Instead of logistic regression, you could put a neural network in there:

$$p(y \mid x) \propto \exp\left(\sum_{c=1}^{k} y_c v^T h(W^3 h(W^2(W^1 x_c))) + \sum_{(c,c') \in E} y_c y_{c'} w\right).$$

- Sometimes called a conditional neural field or deep structured model.
- Backprop generalizes:
  1. Forward pass through neural network to get $\hat{y}_c$ predictions.
  2. Belief propagation to get marginals of y_c (or Gibbs sampling if high treewidth).
  3. Backwards pass through neural network to get all gradients.

Multi-Label Classification

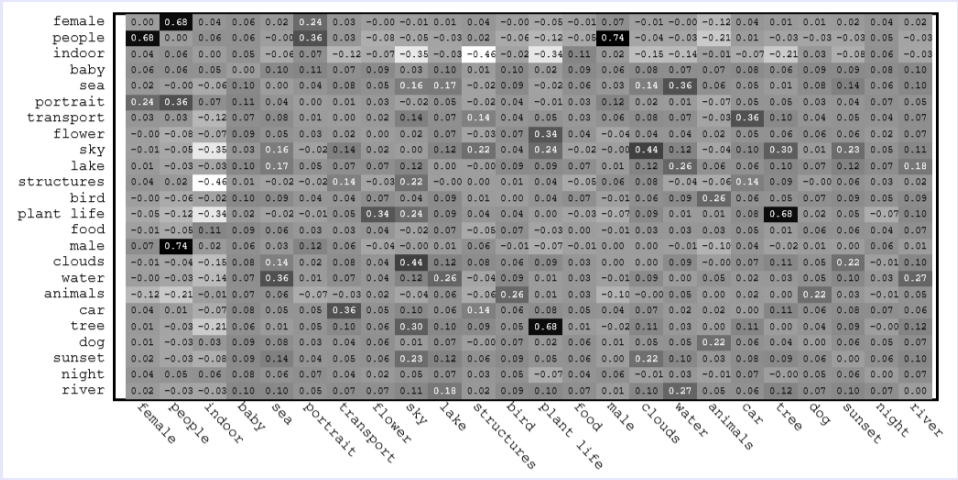
- Consider multi-label classification:

http://proceedings.mlr.press/v37/chenb15.pdf

- Flickr dataset: each image can have multiple labels (out of 38 possibilities).
- Use neural networks to generate "factors" in an undirected model.
  - Decoding undirected model makes predictions accounting for label correlations.

Multi-Label Classification

- Learned correlation matrix:



Automatic Differentiation (AD) vs. Inference

- If you use exact inference methods, automatic differentiation will give gradient.
  - You write message-passing code to compute Z.
  - AD modifies your code to compute expectations in gradient.

- With approximate inference, AD may or may not work:
  - AD will work for iterative variational inference methods (which we'll cover late).
  - AD will not tend to work for Monte Carlo methods.
    - Can't AD through sampling (but there exist tricks like "common random numbers").

- Recent trend: run iterative variational method for a fixed number of iterations.
  - AD can give gradient of result after this fixed number of iterations.
  - "Train the inference you will use at test time".

Motivation: Gesture Recognition

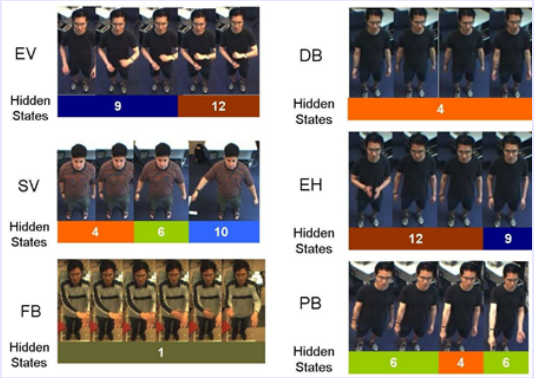
- Want to recognize gestures from video:

http://groups.csail.mit.edu/vision/vip/papers/wang06cvpr.pdf

- A gesture is composed of a sequence of parts:
  - And some parts appear in different gestures.

Motivation: Gesture Recognition

- We may not know the set of "parts" that make up gestures.

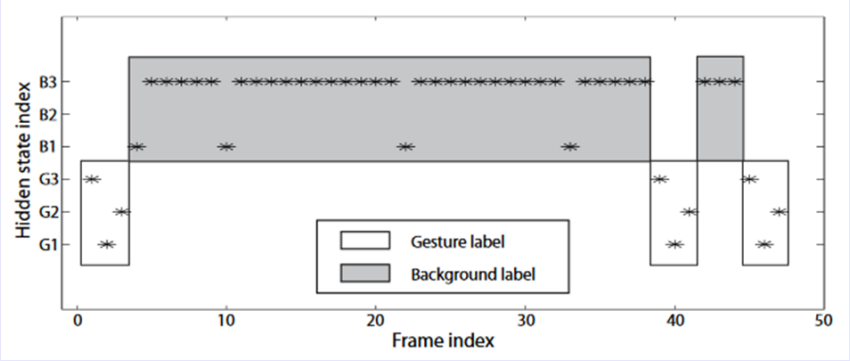


http://groups.csail.mit.edu/vision/vip/papers/wang06cvpr.pdf

- We can consider learn the "parts" and their latent dynamics (transitions).

Motivation: Gesture Recognition

- We're given a labeled video sequence, but don't observe "parts":

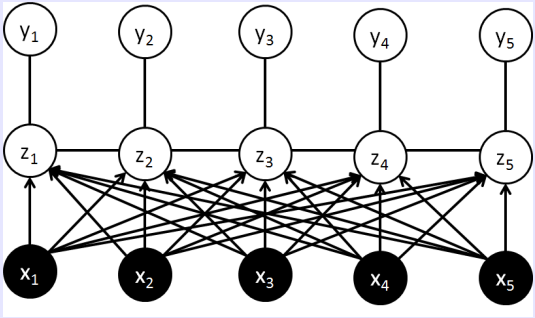


http://www.lsi.upc.edu/~aquattoni/AllMyPapers/cvpr_07_L.pdf

- Our videos are labeled with "gesture" and "background" frames,
  - But we don't know the parts (G1, G2, G3, B1, B2, B3) that define the labels.

Latent-Dynamic Conditional Random Field

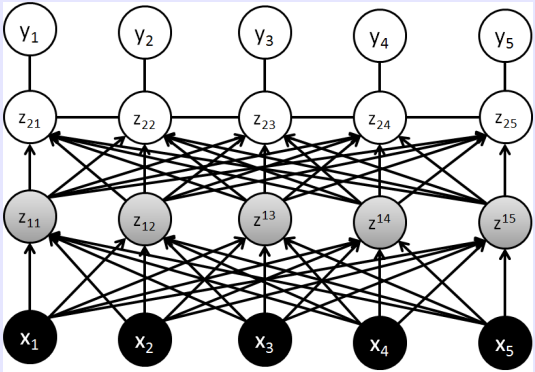
- Here we could use a latent-dynamic conditional random field



- Observed variable x_j is the image at time j (in this case x_j is a video frame).
- The gesture y is defined by sequence of parts z_j.
  - We're learning what the parts should be.
  - We're learning "latent dynamics": how the hidden parts change over time.
- Notice in the above case that the conditional UGM is a tree.

Neural Networks with Latent-Dynamics

- Neural networks with latent dynamics:

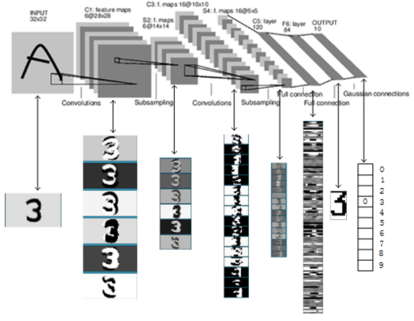


- Combines deep learning, mixture models, and graphical models.
  - Achieved among state of the art in several applications.

Outline

Convolutional Neural Networks

- In 340 we discussed convolutional neural networks (CNNs):

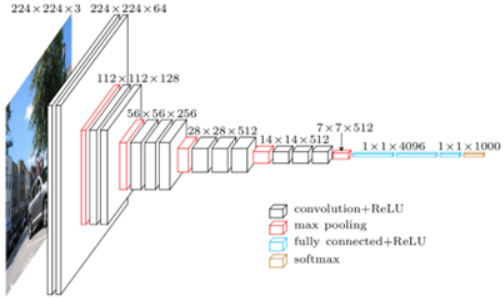


http://blog.csdn.net/strint/article/details/44163869

- Convolutional layers where W acts like a convolution (sparse with tied parameters).
- Pooling layers that usually take maximum among a small spatial neighbourhood.
- Fully-connected layers that use an unrestricted W.

Motivation: Beyond Classification

- Convolutional structure simplifies the learning task:
  - Parameter tieing means we have more data to estimate each parameter.
  - Sparsity drastically reduces number of parameters.

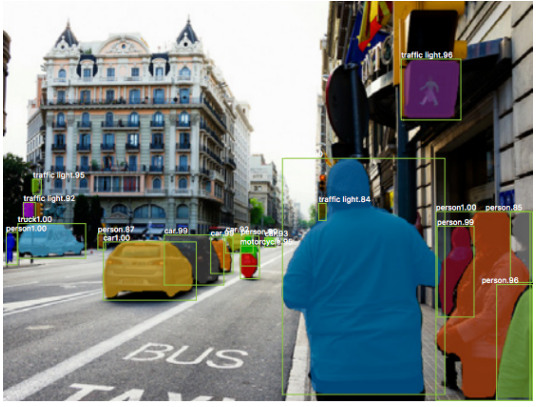


https://www.cs.toronto.edu/~frossard/post/vgg16

- We discussed CNNs for image classification: "is this an image of a cat?".
  - But many vision tasks are not image classification tasks.

Object Localization

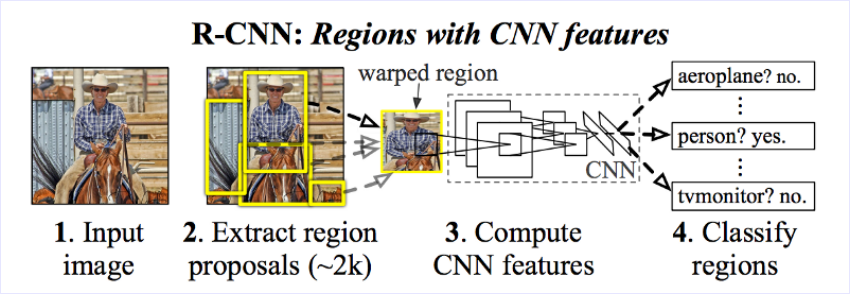
- Object localization is task of finding locations of objects:
  - Need to find *where* in the image the object is.
  - May need to recognize *more than one* object.



https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4

Region Convolutional Neural Networks: "Pipeline" Approach

- Early approach (region CNN):
  1. Propose a bunch of potential boxes.
  2. Compute features of box using a CNN.
  3. Classify each box based on an SVM.
  4. Refine each box using linear regression.

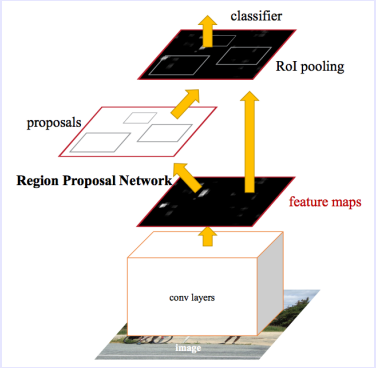


https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4

- Improved on state of the art, but not very elegant with its 4 steps.

Region Convolutional Neural Networks: "End to End" Approach

- Modern approaches try to do the whole task with one neural network.
  - The network extracts features, proposes boxes, and classifies boxes.



https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4

- This is called an end-to-end model.

End-to-End Computer Vision Models

- Key ideas behind end-to-end systems:
  1. Write each step as a differentiable operator.
  2. Train all steps using backpropagation and stochastic gradient.

- There now exist end-to-end models for all the standard vision tasks.
  - Depth estimation, pose estimation, optical flow, tracking, 3D geometry, and so on.
  - A bit hard to track the progress at the moment.
  - A survey of ≈ 200 papers from 2016:
    - `http://www.themtank.org/a-year-in-computer-vision`

  .

- We'l focus on the task of pixel labeling...

Summary

- Backpropagation can be viewed as a message passing algorithm.
- Combining CRFs with deep learning.
  - You can learn the features and the label dependency at the same time.
- End to end models: use a neural network to do all steps.
  - Computer vision can now actually work!
- Next time: generating poetry, music, and dance moves.