Towards a Biologically Plausible Replacement for Back-Prop: Target-Prop

Yoshua Bengio
December 13, 2014
NIPS’2014 MLINI Workshop

PLUG: Deep Learning, MIT Press book in preparation, draft chapters online for feedback
Issues with Back-Prop

- Over very deep nets or recurrent nets with many steps, non-linearities compose and yield sharp non-linearity $\rightarrow$ gradients vanish or explode
- Training deeper nets: harder optimization
- In the extreme of non-linearity: discrete functions, can’t use back-prop
- *Biological plausibility?*
Biological Implausibility of Backprop vs Targetprop

- Not quite local, and not quite neural backprop operations:
  - Purely linear operation
  - Needs precise knowledge of derivative of fprop nonlinearity
  - Spikes?
  - Non-local requirement: “weight transport”
    - The $w_{ij}$ used in fprop network must match the $w_{ij}$ used in the bprop (feedback) network, i.e., symmetric weights
- Where is the target coming from?
- Needs a clock to alternate between fprop and bprop
- BPTT is even worse (storing sequence of all activations and running gradients backwards in time)
Previous work: Boltzmann machine

(Hinton et al, Salakhutdinov et al)

- Not yet successful but biologically reasonable algorithm
  - Clamp $x$ and $y$
  - Stochastic relaxation
  - Measure and add pre $x$ post
  - Release $y$ (or both $x$ and $y$)
  - Stochastic relaxation
  - Measure and subtract pre $x$ post
- Sleep phase: both $x$ and $y$ released.
- Needs symmetric weights
Issues with Undirected Graphical Models & Boltzmann Machines

- Sampling from the MCMC of the model is required in the inner loop of training
- As the model gets sharper, mixing between well-separated modes stalls

Training updates

\textit{vicious circle}

Mixing
REINFORCE & Stochastic Perturbation

(Williams 1992, “Simple Statistical Gradient-Following Algorithms Connectionist Reinforcement Learning”)
(Fiete & Seung 2006, “Gradient learning in spiking neural networks by dynamic perturbations of conductances”)

- Correlating reward (or log-likelihood) with stochastic perturbations gives rise to a gradient estimator and a learning algorithm that can be applied to spiking neural networks

- Very inefficient (even with variance reduction) and does not scale well with size of the network, whereas backprop does
Deterministic Relaxation

(Xie & Seung 2003, “Equivalence of backpropagation and contrastive Hebbian learning in a layered network”)

• **Contrastive Hebbian learning**
  • Similar to deep Boltzmann machine but deterministic
  • Also needs symmetric weights

• Equivalence to back-prop shown in the case where a fixed point of the relaxation is reached (for both cases of free Y and clamped Y) and the feedback weights are weak (to obtain derivatives by Taylor expansion)

• Is a single step or a few steps enough?
**Temporal Derivative = Loss Gradient**

*(Hinton, NIPS DL Workshop 2007 talk, “How to do backpropagation in a brain”)*

- **Weak feedback weights are symmetric** and are used to propagate gradients.
- **Non-linear activation**: \( h_i = \sigma(a_i) = \sigma(b_i + \sum W_{ij} h_j) \)
- **Temporal derivative of spiking rate** represents gradient on potential: \( \frac{\partial h_i}{\partial t} = \frac{\partial C}{\partial a_i} \)
- **Weak feedback** influence goes through \( \sigma \) and becomes amplified by \( W'_{ji} \sigma'(a_j) \) to yield temporal change in \( h_j \)
- **STDP update** = delta-rule (pre \( x \) d/dt post)
Feedback Alignment

- (Lillicrap et al arxiv 2014, “Random feedback weights support learning in deep networks”)
- Feedback weights $B$ (instead of $W^T$ in backprop) are fixed
- As good as back-prop on not too deep nets: angle($W, B$) decreases during training (but does not reach 0)
Generalizing gradients to non-differentiable credit assignment: Target Propagation

- Gradient = in which direction each unit should make infinitesimal change towards reducing loss function

  *not infinitesimal*

- Target = nearby value which would yield a smaller loss

- When target - value is small → equivalent to backprop via Lagrange multipliers (LeCun thesis, 1986, 1987)

- Principle of target propagation can work for discrete values
Target Prop
(Bengio 2014, arXiv 1407.7906)

• Instead of propagating the effect of an infinitesimal change, propagate a target that would be
  • Near the original value
  • Yielding to a lower loss
• Can be obtained by maintaining each layer as an auto-encoder:
  good target $\hat{h}_{l-1}$ s.t.
  
  $f_l(\hat{h}_{l-1}) = f_l(g_l(\hat{h}_l)) = \hat{h}_l$

• Preliminary experiments: works with correction for imperfect inverse:
  
  $\hat{h}_{l-1} = h_{l-1} + g_l(\hat{h}_l) - g_l(h_l)$
How to train an auto-encoder without backprop

- To learn a shallow auto-encoder **without backprop**: could be potentially applied to discrete units, biologically more plausible
- If you observe the output $y=f(x)$ for some $x$, that gives you a training example
  
  (input = $y$, target = $x$)

  for training a function $g$ that tries to invert $f$
- Here we want both the encoder $f$ and the decoder $g$ to invert each other
Recirculation & Backprop-free AEs
(Hinton & McClelland 1988, Lee & Bengio 2014)

Also: (O’Reilly 1996, generalized recirculation)

- Experiments with back-prop free auto-encoder training similar to Recirculation algorithm.
- Minimizes 2-way reconstruction losses
**Back-Step BFAE**

*(Lee & Bengio, NIPS 2014 Deep Learning workshop)*

- By symmetry, we minimize both encode/decode and decode/encode reconstruction errors, unlike in ordinary auto-encoders.
- To make the learning of encoder more relevant, we consider the decode/encode step where the input of the encoder is approximately $x$, using a local approximation:
Denoising Auto-Encoders (Minimizing Reconstruction Error) Learn to Model the Input Distribution

- *(Alain & Bengio ICLR 2013)*: reconstruction-input = dlogp(x)/dx
- *(Alain & Bengio ICLR 2013; Bengio et al, arxiv 2013)*

*(Bengio et al NIPS’2013; Bengio et al ICML’2014)*

- Encode-Decode iterations without noise = local MAP
- Encode-Decode iterations with noise = MCMC samples from estimated generative model
- Clamped Encode-Decode iterations fill-in missing values
- GSNs generalize this to arbitrary recurrent net with injected noise
Denoising Auto-Encoders Learn a Small Move Towards Higher Probability

(Alain & Bengio ICLR 2013)

- Reconstruction \( \hat{x} \) points in direction of higher probability

\[
\hat{x} - x \propto \frac{\partial \log P(x)}{\partial x}
\]

- Trained with input/target pair = (corrupted \( \tilde{x} \rightarrow \) clean data \( x \))

- DAE \( \rightarrow \) Score matching
  (Vincent 2011)

Reconstruction is towards more probable configuration according to AE
Regularized Auto-Encoders Learn a Vector Field or a Markov Chain Transition Distribution

- (Bengio, Vincent & Courville, TPAMI 2013) review paper
- (Alain & Bengio ICLR 2013; Bengio et al, arxiv 2013)
- (Bengio et al NIPS’2013; Bengio et al ICML’2014)
Generative Stochastic Networks

- Generalizes the denoising auto-encoder training scheme
  - Introduce latent variables in the Markov chain (over X,H)
  - Instead of a fixed corruption process, have a deterministic function with parameters $\theta_1$ and a noise source $Z$ as input

$$H_{t+1} = f_{\theta_1}(X_t, Z_t, H_t)$$

- DAE special case of GSN, both generate a Markov chain whose stationary distribution is a consistent estimator of the data generating distribution \textit{(Bengio et al, NIPS’2013; ICML’2014)}

\[
\begin{align*}
H_{t+1} &\sim P_{\theta_1}(H|H_t, X_t) \\
X_{t+1} &\sim P_{\theta_2}(X|H_{t+1})
\end{align*}
\]
Ancestral Sampling with Learned Approximate Inference

- Helmholtz machine & Wake-Sleep algorithm
  - (Dayan, Hinton, Neal, Zemel 1995)
- Variational Auto-Encoders
  - (Kingma & Welling 2013, ICLR 2014)
  - (Gregor et al ICML 2014)
  - (Rezende et al ICML 2014)
  - (Mnih & Gregor ICML 2014)
- Reweighted Wake-Sleep (Bornschein & Bengio 2014)
- Target Propagation (Bengio 2014)
- Deep Directed Generative Auto-Encoders (Ozair & Bengio 2014)
- NICE (Dinh et al 2014)
Variational Auto-Encoder Training

Objective

- Two distributions:
  - data $Q(x) \rightarrow h \sim Q(h|x): Q(x,h)$
  - model $P(h) \times P(x|h): P(x,h)$

- Variational bound is equivalent to the following natural objective:

\[
\min KL(Q(x,h) \mid \mid P(x,h))
\]

Can consistently be applied at every layer $h$ of a deep net
All the pieces are tractable

\[ KL(Q(x, h) \| P(x, h)) \]

Decomposes into

- \( -H(Q) \)
- Reconstruction error:
  \[ E_{Q(h, x)}[- \log P(x|h)] \]
- Bottom-up / Top-down match:
  \[ E_{Q(h, x)}[- \log P(h)] \]

\( \to \) upper model likelihood

\( \to \) lower encoder tries to reduce upper error

tractable if top \( P(h) \) is an auto-encoder:

\[ \hat{h} - h \propto \frac{\partial \log P(h)}{\partial h} \]
Extracting Structure By Gradual Disentangling and Manifold Unfolding (Bengio 2014, arXiv 1407.7906)

Each level transforms the data into a representation in which it is easier to model, unfolding it more, contracting the noise dimensions and mapping the signal dimensions to a factorized (uniform-like) distribution.

$$\min KL(Q(x, h) \| P(x, h))$$

for each intermediate level $h$
Disentangling with Deep Nets

- With sparse auto-encoders, the pressure to make $P(h)$ simple comes from the sparsity penalty.

- However, we would also like lower-level encoders to help the higher-level encoders achieve a better $P(h)$.

- Minimizing $KL(Q(h_L,x) \parallel P(h_L,x))$ at each level $h_l$ achieves that!

- However, with INSUFFICIENT DEPTH, the sparsity (or other simplicity preference) of $P(h)$ cannot be achieved without yielding a mismatch between $P(h)$ and $Q(h)$, hence a poor $P(x)$.
The Importance of Contraction

- Denoising $\rightarrow$ contractive $g$
- Max. determinant of $f'$ $\rightarrow$ contractive $g$
- Contraction $\rightarrow$ removes unnecessary directions
- The $\log P(h = f(x) + \text{noise})$ force on the encoder $f$ makes $f$ contractive, making it insensitive to directions of non-variation in $x$-space
- Making $g$ contractive helps to manage the mismatch between $P(h)$ and $Q(h)$
- Adding noise at the top-level in $Q(h|\mathbf{x})$ shows to the decoder which directions of $h$ need to be contracted out, making it contractive too
• Each layer tries to be a good denoising auto-encoder while transforming the lower-level data into a form $h$ easier to model by higher levels: higher $P(h)$
• This basically makes the long-path reconstructions (going all the way up) a target $\hat{h}$ for the original $h$, and vice-versa, while the long-path auto-encoder is trained with $h$ as data
**Target-Prop as an alternative to Back-Prop**

- Weights at each layer are trained to match targets associated with the layer output.
- How to propagate targets?
- If $g_i$ is the inverse of $f_i$, then it guarantees an improvement in the cost.

\[
\begin{align*}
\text{cost}(h_i) & > \text{cost}(\hat{h}_i) \\
\text{cost}(f_i(h_{i-1})) & > \text{cost}(f_i(\hat{h}_{i-1}) \approx \hat{h}_i)
\end{align*}
\]
Difference Target-Prop for Inexact Inverse

- Make a correction that guarantees to first order that the projection estimated target is closer to the correct target than the original value

\[ \hat{h}_{i-1} = h_{i-1} - g_i(h_i) + g_i(\hat{h}_i) \]

- Special case: feedback alignment, if \( g_i(h) = B \, h \)

\[ \| \hat{h}_i - f_i(\hat{h}_{i-1}) \|^2 < \| \hat{h}_i - h_i \|^2 \]

if \( 1 > \max \text{ eigen value } \left( (I - f'_i(h_{i-1})g'_i(h_i)) \right)^T (I - f'_i(h_{i-1})g'_i(h_i)) \)
Experimental Result

- We used hyper-parameters for the best valid error respectively.
- Test error: 1.73%: target prop with high regression,
  1.62%: difference target prop,
  1.44%: back-prop, respective learning rates.

Experimental Result

- Left graph: Hyper-parameters for the best valid error.
- Right graph: Hyper-parameters for the best training cost at 100 epoch.
- Target prop is sometimes faster than back-prop though it is usually overfitting, but it can solve under-fitting problem (ex - very deep net, highly non-linear net and discrete net).

Hyper-optimizing for validation error

- Hyper-optimizing for validation error
- Hyper-optimizing for training error

Hyper-optimizing for validation error
Targetprop can work for discrete activations

- Work in progress

Experimental Result

- We used hyper-parameters for the best valid error

  - Test error: 2.5\% (discrete networks with 3 hidden layers), 2.5\% (discrete networks with 2 hidden layers), 5-6\% (just training top classifier with 2 hidden : back-prop)
Target-Prop on Deep Nets


Tricks: noise injection, combine feedforward and feedback values, train lower layers first
From Supervised Targets to Reconstructions from Other Modalities: Multi-Modal / Structured Output

- $y$ is complex and needs its own $P(y)$ (modeled by it’s own stack of auto-encoders) and non-trivial $P(y|x)$. Model joint of $h_x(x)$ and $h_y(y)$ with another stack on top.

- Inference (MAP or MCMC) is done with the top stack, then projected back in the $x$ or $y$ space.

- Top level is a DAE or GSN
(Conditional) Sampling & MAP

• Two things we want from our models:
  • Probabilistic inference:
    • Sample some variables given others (or none)
  • MAP inference:
    • Choose likely values for some variables given others
• Both can be done here:
  • Unconditional sampling by ancestral sampling from P
  • Conditional sampling by GSN-like MCMC, clamping the given variables and resampling others:
    • Iteratively encode/decode with noise injected (top level stack)
  • Local ascent for approximate MAP:
    • Iteratively encode/decode with no noise injected (top level stack)
Ambiguous Multi-Modal Posteriors on Latent Variables

• The simple stack model aims at a factorized posterior $Q(h|x)$
• However, plausible latent variables must have a **multi-modal posterior**
• Latent variables can be thought of as unobserved labels $\phi$
• Stick an AE/GSN on top with the latent variables $\phi$ and $h$ as input
Multiple Iterations of Target-Prop

Set $h$ to opt. reconstruction (et al & LeCun 200x) or prediction error (Perpinan & Wang AISTATS 2014)

- Want time-invariant computation/update without f/bprop phases
- If target = new value of $h$, target-prop update
  $$\hat{h}^l = h^l + g(\hat{h}^{l+1}) - g(h^{l+1})$$
  can be interpreted as
  $$\text{new } h^l - \text{old } h^l = g(\text{new } h^{l+1}) - g(\text{old } h^{l+1})$$
  $$\Delta h = \Delta r$$
- Where $r$ = output of reconstruction (feedback path) units
  $$r = g(\text{upper level } h)$$
- If we do several iterations, $h$ continues to move towards better matching the top-level target but may become unrealistic for realization by feedforward path
- Iterations stop when target is reached as well as possible
Reconstructing Both $x$ and $y$

- Feedback path = target for feedforward path (better predict $y$)
- Feedforward path = target for feedback path (better predict $x$)
- Good 2-way auto-encoders give that for free
- Iterative targetprop is run both ways
Multiple Iterations of Target-Prop → MAP or EM training

- One option is to force \( h \) to remain close to its feedforward value
- More appealing: look for \( h \) that is compatible with both \( x \) and \( y \)
- Achievable by moving \( h \) towards better reconstruction of both \( x \) and \( y \)
- If we add a sparsity prior, this corresponds to
  \[
  \arg\max_h \log P(h|x, y)
  \]
- And if we inject noise, this corresponds a GSN MCMC towards
  \[
  h \sim P(h|x, y)
  \]
- And we can then do a layer-wise update (delta-rule = STDP) which corresponds to an EM update of \( P(x,y,h) \)
**Delta-Rule Updates in Cortex**

- *(Urbanczik & Senn, Neuron Report 2014, “Learning by the dendritic prediction of somatic spiking”)*
- Dendritic synapses learn by matching the target imposed by soma-level synapses, as in Delta-Rule
- Spiking rate at soma depends on both, dominated by soma-synapses

![Diagram of neuron showing dendritic tree, soma, axon, input x: adaptive synapse into dendritic tree, and target y: overriding synapse into soma.](image)
How Brains Might Learn Without Backprop

• Two kinds of nodes: feedforward & feedback
• Feedback nodes = targets for feedforward ones and vice-versa
• Initially clamp $x$, fprop, then clamp $y$, then settle, delta-rule update along the way (new $x,y$ can arrive at any time)

• Does not depend on the form of activation function, tied symmetric weights, differentiability of anything, using rates vs spikes, etc.

*Extends Hinton’s proposal made at the first DL workshop NIPS’2007 based on encoding gradients as temporal derivatives*
Conclusion: Beautiful Coincidences

• Same basic mechanism (auto-encoders) basically does it all

• Local training signals provided only by using activity at the neurons and synapses (like in the Boltzmann machine, but no need to achieve mixing in inner loop of training)

• Not requiring the differentiability of neuronal activation and noise helps to achieve to needed contraction (could be used to train spiking neurons?)

• Same mechanism for supervised, unsupervised, and multi-modal learning, and structured-output sampling or MAP

• Handling sequences: relation to Jaeger’s Conceptors...
PLUG: Deep Learning, MIT Press book in preparation, draft chapters online for feedback

LISA team: Merci! Questions?