Accelerated deep learning with TensorFlow
Lecture 5 Tricks of the trade

Ole Winther

Dept for Applied Mathematics and Computer Science
Technical University of Denmark (DTU)

January 23, 2017

Objectives of lecture

- Tricks of the trade!
- P1: Faster convergence
  - Advanced stochastic gradient descent
  - Initialization
  - Gradient clipping
  - Batch normalization
- P2: Better performance - regularization
  - Weight decay/priors
  - Early stopping
  - Dropout
- P3: A few final words

Part 1:
Faster training convergence

Adam algorithm by Kingma and Ba

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1$^{\text{st}}$ moment vector)
  $v_0 \leftarrow 0$ (Initialize 2$^{\text{nd}}$ moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

Adam algorithm by Kingma and Ba

- $\mathbb{E}[(\theta)]$ is the cost function we want to optimize
- $f(\theta_t)$ stochastic estimate of $\mathbb{E}[(\theta)]$, e.g. training cost on mini-batch.
- Momentum:

$$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1)\nabla_\theta f(\theta_{t-1})$$

- Correct bias from $m_0 = 0$.
- Normalize stepsize with $\sqrt{\text{estimate of gradient}^2}$ so that

$$|\Delta\theta| \lesssim \alpha$$

- Never take step larger than α
- Take roughly equal steps in all directions, but $\beta_2 > \beta_1$ and $\epsilon > 0$ ensures that we will stop at minimum.
- Demo

Initialization

- Initialize weights $W_{ij} \sim \sqrt{\frac{4}{n_i+n_j}}\mathcal{N}(0,1)$
- where size of $\mathbf{W}$ is $n_i \times n_j$.

Initialization

- Initialize weights $W_{ij} \sim \sqrt{\frac{4}{n_i + n_j}} \mathcal{N}(0, 1)$
- where size of $\mathbf{W}$ is $n_i \times n_j$.
- Outline of derivation:
- Assume independent signals.

Initialization

- Initialize weights $W_{ij} \sim \sqrt{\frac{4}{n_i+n_j}}\mathcal{N}(0,1)$
- where size of $\mathbf{W}$ is $n_i \times n_j$.
- Outline of derivation:
- Assume independent signals.
- Retain variance forward: $\sqrt{\frac{2}{n_j}}$
- Retain variance backward: $\sqrt{\frac{2}{n_i}}$
- (2 is from relu indicators being on half the time.)

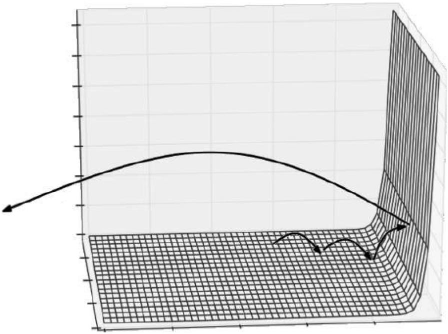
Initialization

- Initialize weights $W_{ij} \sim \sqrt{\frac{4}{n_i+n_j}}\mathcal{N}(0,1)$
- where size of $\mathbf{W}$ is $n_i \times n_j$.
- Outline of derivation:
- Assume independent signals.
- Retain variance forward: $\sqrt{\frac{2}{n_j}}$
- Retain variance backward: $\sqrt{\frac{2}{n_i}}$
- (2 is from relu indicators being on half the time.)
- Strike a balance between the two (Glorot and Bengio, 2010).

Initialization

- Initialize weights $W_{ij} \sim \sqrt{\frac{4}{n_i+n_j}}\mathcal{N}(0,1)$
- where size of $\mathbf{W}$ is $n_i \times n_j$.
- Outline of derivation:
- Assume independent signals.
- Retain variance forward: $\sqrt{\frac{2}{n_j}}$
- Retain variance backward: $\sqrt{\frac{2}{n_i}}$
- (2 is from relu indicators being on half the time.)
- Strike a balance between the two (Glorot and Bengio, 2010).
- (Other ideas, relevant for RNNs: sparse initialization (Martens, 2010), orthogonal initialization (Saxe et al., 2014))

Gradient clipping



- Highly nonlinear model: Gradient update can catapult parameters very far.
- Heuristic: Clip the magnitude of the gradient.
- Figure from (Pascanu, 2014)

Batch normalization (Ioffe and Szegedy, 2015)

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

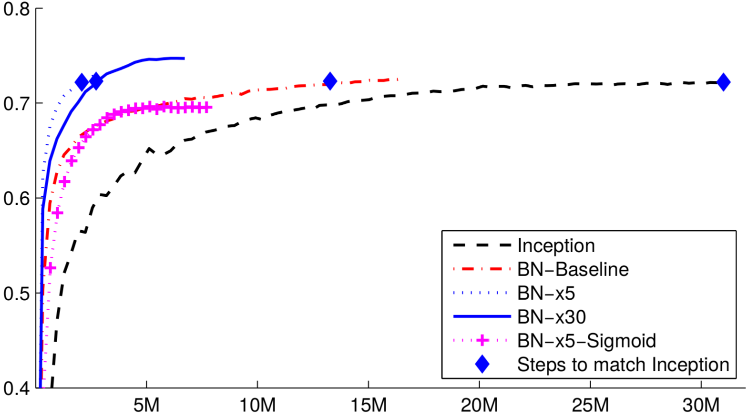
$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

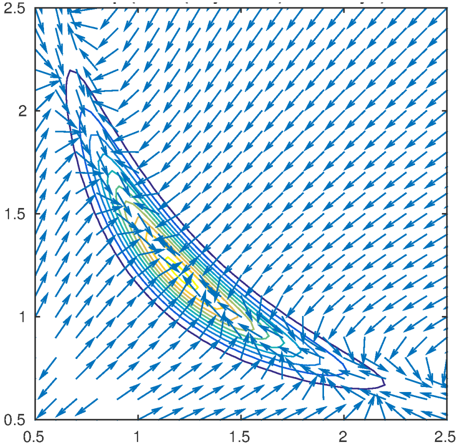
Batch normalization (Ioffe and Szegedy, 2015)

- Improves learning (BN-baseline vs. Inception)
- Allows bigger learning rates (5x and 30x)

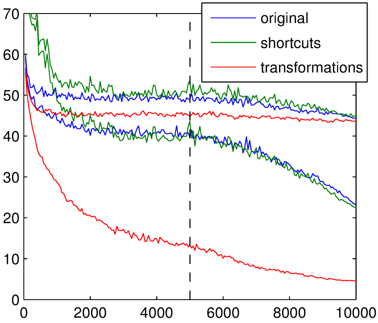


Why does it work? Recall Newton's

$$\theta_{k+1} = \theta_k - \mathbf{H}_k^{-1}\mathbf{g}_k, \quad \mathbf{H} = \begin{pmatrix} \frac{\partial^2 C}{\partial\theta_1\partial\theta_1} & \cdots & \frac{\partial^2 C}{\partial\theta_1\partial\theta_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 C}{\partial\theta_n\partial\theta_1} & \cdots & \frac{\partial^2 C}{\partial\theta_n\partial\theta_n} \end{pmatrix}$$

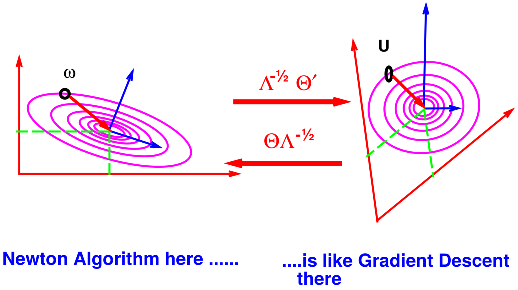


Why does it work? (Raiko et al., 2012)



- Transformations do not change the model, but the optimisation
- Hessian **H** is closer to a diagonal
- Traditional gradient is thus closer to Newton's and parameter updates are more independent

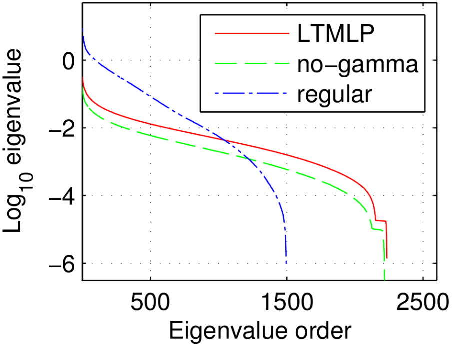
Eigenvalues of the Hessian $\mathbf{H}$ (LeCun et al., 1998)



- Eigenvectors corresponds to (update) directions
- In Newton's method, each direction has its own learning rate: inverse of the eigenvalue
- Some eigenvalues can be negative: Newton's method points the wrong way

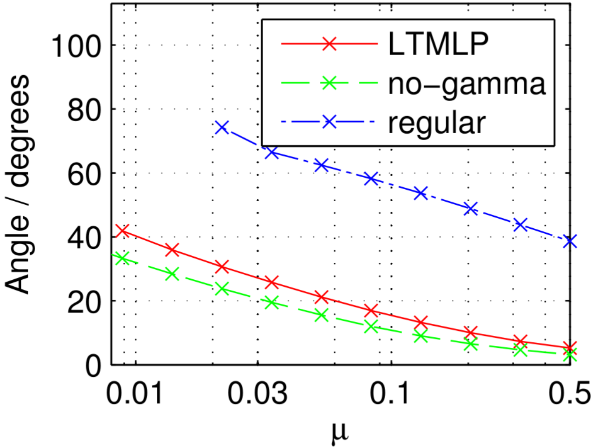
Why does it work? (Vatanen et al., 2013)

- Analysis of eigenvalues of Hessian in 2600-parameter model
- Curvature is much more even with transformations



Why does it work? (Vatanen et al., 2013)

Angle between gradient and second order update:
$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - (\mathbf{H}_k + \mu\mathbf{I})^{-1}\mathbf{g}_k$,



Part 2:

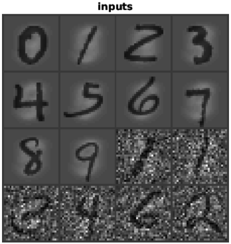
Better performance

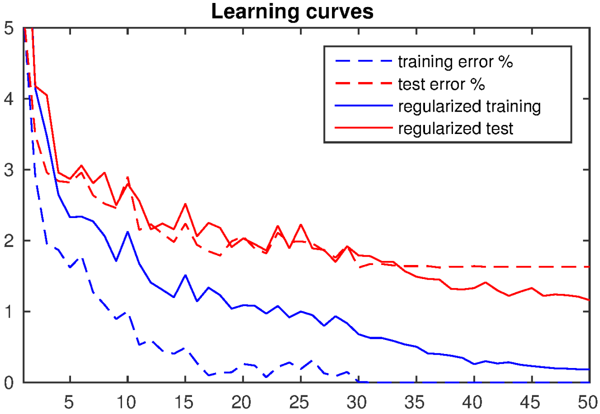
Regularization

Goal of Regularization

- Neural networks are very powerful (universal appr.).
- Easy to perform great on the training set (overfitting).
- **Regularization** improves generalization to new data at the expense of increased training error.
- Use held-out validation data to choose hyperparameters (e.g. regularization strength).
- Use held-out test data to evaluate performance.

Example

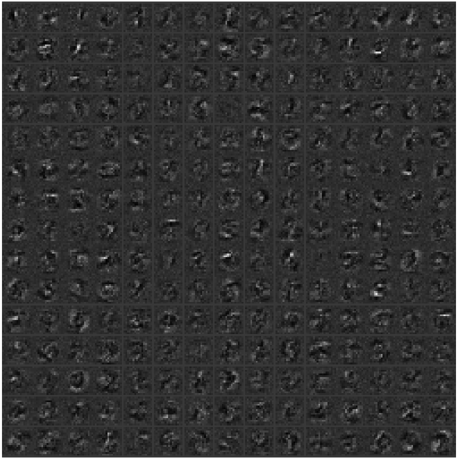




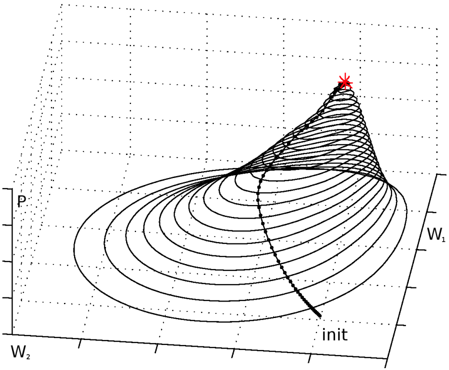
Without regularization training error goes to zero and learning stops.
With noise regularization, test error keeps dropping.

Expressivity demo: Training first layer only

No regularization, training $\mathbf{W}^{(1)}$ and $\mathbf{b}^{(1)}$ only.
0.2% error on training set, 2% error on test set.



What is overfitting?



Posterior probability mass matters

Center of gravity $\neq$ maximum

Probability theory states how we should make predictions (of $\mathbf{y}_{\text{test}}$) using a model with unknowns $\boldsymbol{\theta}$ and data $\mathbf{X} = \{\mathbf{x}_{\text{train}}, \mathbf{y}_{\text{train}}, \mathbf{x}_{\text{test}}\}$:

$$P(\mathbf{y}_{\text{test}} \mid \mathbf{X}) = \int P(\mathbf{y}_{\text{test}}, \boldsymbol{\theta} \mid \mathbf{X}) \mathrm{d}\boldsymbol{\theta}$$
$$= \int P(\mathbf{y}_{\text{test}} \mid \boldsymbol{\theta}, \mathbf{X}) P(\boldsymbol{\theta} \mid \mathbf{X}) \mathrm{d}\boldsymbol{\theta}.$$

Probability of observing $\mathbf{y}_{\text{test}}$ can be acquired by summing or integrating over all different explanations $\boldsymbol{\theta}$. The term $P(\mathbf{y}_{\text{test}} | \boldsymbol{\theta}, \mathbf{X})$ is the probability of $\mathbf{y}_{\text{test}}$ given a particular explanation $\boldsymbol{\theta}$ and it is weighted with the probability of the explanation $P(\boldsymbol{\theta} | \mathbf{X})$. However, such computation is intractable. If we want to choose a single $\boldsymbol{\theta}$ to represent all the probability mass, it is better not to overfit to the highest probability peak, but to find a good representative of the mass.

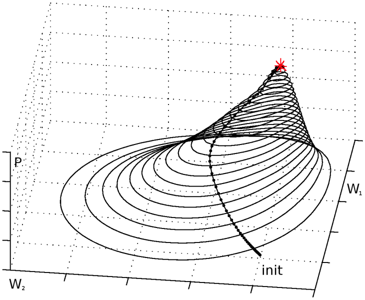
Regularization methods

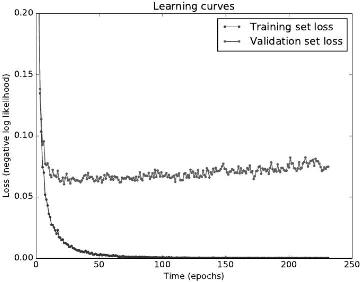
- Limited size of network
- Early stopping
- Weight decay
- Data augmentation
- Injecting noise
- Parameter sharing (e.g. convolutional)
- Sparse representations
- Ensemble methods
- Auxiliary tasks (e.g. unsupervised)
- Probabilistic treatment (e.g. variational methods)
- Adversarial training, . . .

Limited size of network

- Rule of thumb:
  When #parameters is ten times less than #outputs $\times$ #examples, overfitting will not be severe.
- Reducing input dimensionality (e.g. by PCA) helps in reducing parameters
- Easy. Low computational complexity
- Other methods give better accuracy
- *Data augmentation* increases #examples
  Parameter sharing decreases #parameters
  Auxiliary tasks increases #outputs

Early stopping



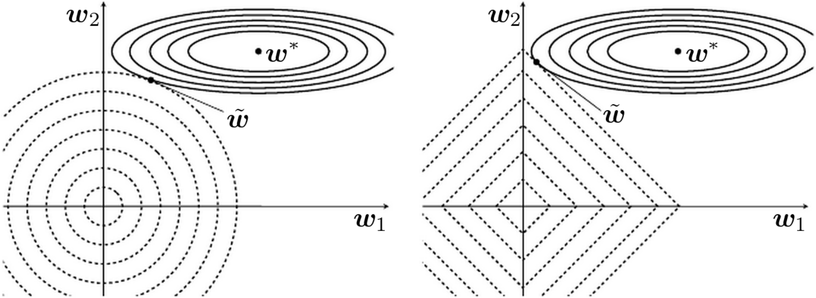


- Monitor validation performance during training
- Stop when it starts to deteriorate
- With other regularization, it might never start
- Keeps solution close to the initialization

Weight decay (Tikhonov, 1943)

- Add a penalty term to the training cost $C = \cdots + \Omega(\boldsymbol{\theta})$
  Note: only a function of parameters $\boldsymbol{\theta}$, not data.
- L^2 regularization: $\Omega(\boldsymbol{\theta}) = \frac{\lambda}{2}\|\boldsymbol{\theta}\|^2$
  hyperparameter λ for strength.
  Gradient: $\frac{\partial \Omega(\boldsymbol{\theta})}{\partial \theta_i} = \lambda \theta_i$.
- L^1 regularization: $\Omega(\boldsymbol{\theta}) = \lambda/2\|\boldsymbol{\theta}\|_1$
  Gradient: $\frac{\partial \Omega(\boldsymbol{\theta})}{\partial \theta_i} = \lambda \mathrm{sign}(\theta_i)$.
  Induces sparsity: Often many params become zero.
- Max-norm: Constrain row vectors $\mathbf{w}_i$ of weight matrices to $\|\mathbf{w}_i\|^2 \leq c$.

Weight decay



- L2 (left) and L1 (right).
- w^* unregularised solution, $\tilde{w}$ regularised solution.
- Note: L1 pushes small parameters more towards zero - sparsity!

Weight decay as Bayesian prior

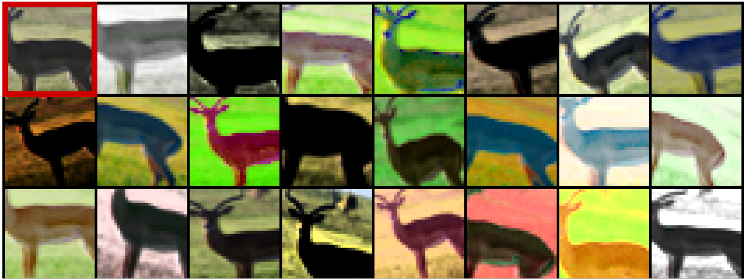
- Consider the maximum a posteriori solution
- Bayes rule: $P(\boldsymbol{\theta} \mid \mathbf{X}) = P(\mathbf{X}|\boldsymbol{\theta})P(\boldsymbol{\theta})$
  written on -log scale: $C = -\log P(\mathbf{X}|\boldsymbol{\theta}) - \log P(\boldsymbol{\theta})$
- Assuming Gaussian prior $P(\boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, \lambda^{-1}\mathbf{I})$
  we get $\Omega(\boldsymbol{\theta}) = \sum_i -\log\exp\frac{-\theta_i^2}{2\lambda^{-1}} = \frac{\lambda}{2}\|\boldsymbol{\theta}\|^2$
- L^2 regularization $\Leftrightarrow$ Gaussian prior
- L^1 regularization $\Leftrightarrow$ Laplace prior
- Max-norm regularization $\Leftrightarrow$ Uniform prior with finite support
- $\Omega = 0 \Leftrightarrow$ Maximum likelihood

How to set hyperparameter λ?

- In general, difficult to set strength of regularization
- Split data into training, validation, and test sets
- Choose a number of settings λ, train separately
- Use validation performance to pick the best λ
- (Retrain using both training and validation sets)
- Evaluate final performance on test data
- Ongoing work on adjusting hyperparameters on the fly (Luketina et al., 2016)

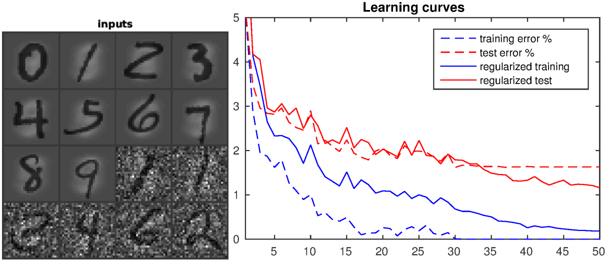
Data augmentation

Image from (Dosovitskiy et al., 2014)



Augmented data by image-specific transformations.
E.g. cropping just 2 pixels gets you 9 times the data! Infinite MNIST: `http://leon.bottou.org/projects/infimnist`

Injecting noise (Sietsma and Dow, 1991)



- Inject random noise during training
  separately in each epoch
- Can be applied to input data, to hidden activations, or to weights
- Can be seen as data augmentation
- Simple end effective

Injecting noise to inputs (analysis)

- Inject small additive Gaussian noise at inputs
- Assume least squares error at output $\mathbf{y}$
- Taylor series expansion around $\mathbf{x}$
- $\Rightarrow$ Corresponds to penalizing the Jacobian $\|\mathbf{J}\|^2$

$$\mathbf{J} = \frac{d\mathbf{y}}{d\mathbf{x}} = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_c}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial x_d} & \cdots & \frac{\partial y_c}{\partial x_d} \end{pmatrix}$$

- For linear networks, this reduces to L^2 penalty
- Rifai et al. (2011) penalize the Jacobian directly

Parameter sharing

- Force sets of parameters to be equal
- Reduces the number of (unique) parameters
- Important in convolutional networks (CNNs, this lecture)
- Auto-encoders sometimes share weights between encoder and decoder (Unsupervised learning lecture)

Sparse representations

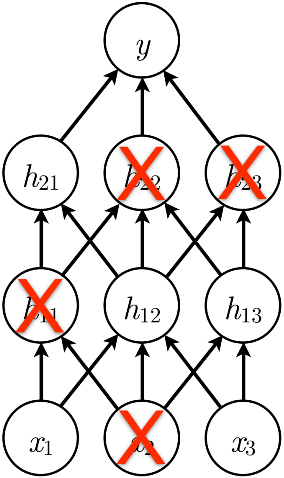
- Penalize representation $\mathbf{h}$ using $\Omega(\mathbf{h})$ to make it sparse
- L^1 penalty on weights makes $\mathbf{W}$ sparse
- Similarly L^1 penalty can make $\mathbf{h}$ sparse
- Also possible to set a desired sparsity level
- *Sparse coding* is common in image processing

Ensemble methods

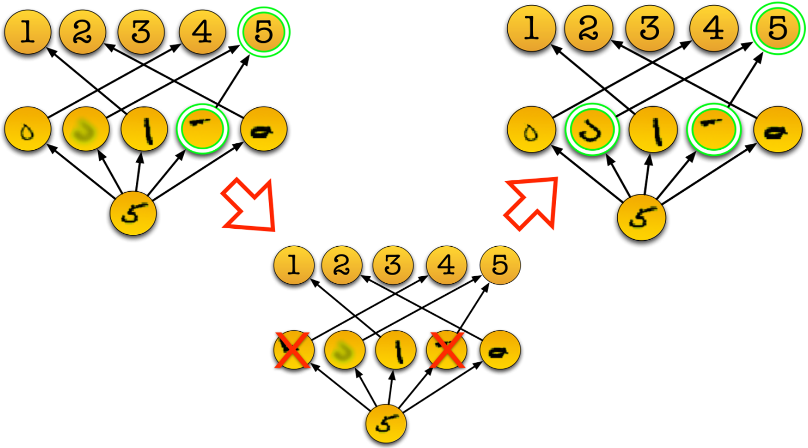
- Train several models and take average of their outputs
  Instead of one point representing $P(\theta|\mathbf{X})$, use several
- Also known as *bagging* or *model averaging*
- It helps to make individual models different by
  - varying models or algorithms
  - varying hyperparameters
  - varying data (dropping examples or dimensions)
  - varying random seed
- It is possible to train a single final model to mimick the performance of the ensemble, for test-time computational efficiency (Hinton et al., 2015)

Dropout (Srivastava et al., 2014)

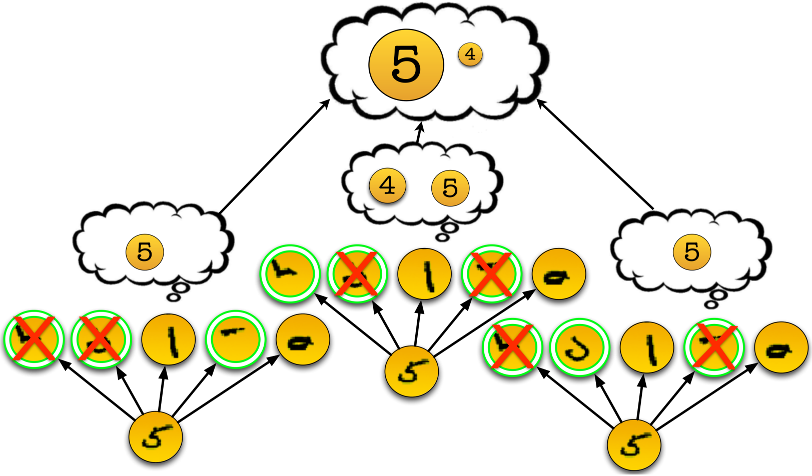
- Each time we present data example **x**, randomly delete each hidden node with 0.5 probability
- Can be seen as *injecting noise* or as *ensemble*:
  - Multiplicative binary noise
  - Training an ensemble of $2^{|h|}$ networks with weight sharing
- At test time, use all nodes but divide weights by 2



Dropout training



Dropout as bagging



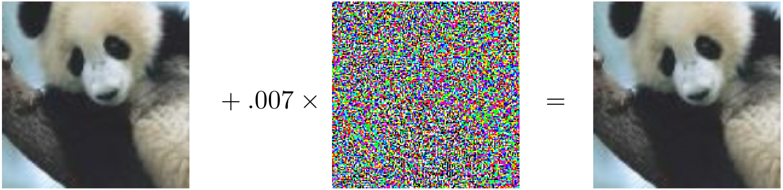
Auxiliary tasks

- Multi-task learning: *Parameter sharing* between multiple tasks
- E.g. speech recognition and speaker identification could share low-level representations
- Layer-wise pretraining (Hinton and Salakhutdinov, 2006) can be seen as using unsupervised learning as an auxiliary task

Probabilistic treatment

- Probabilistic modelling is strong but complex form of regularization
- Some keywords:
  - Variational methods: E.g. model means and variances of signals
  - Sampling = Markov chain Monte Carlo (MCMC)
  - Boltzmann machines

Adversarial training (Szegedy et al., 2014)



$\boldsymbol{x}$ $\quad$ $\mathrm{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$ $\quad$ $\boldsymbol{x} + \epsilon\, \mathrm{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

$y =$ "panda" $\quad$ "gibbon"

- Search for an input $\mathbf{x}'$ near a datapoint $\mathbf{x}$ that would have very different output $\mathbf{y}'$ from $\mathbf{y}$
- Adversaries can be found surprisingly close!
- Miyato et al. (2016) build a very effective regulariser

Part 3:
A few final words

Where humans excel & reinforcement learning gets nowhere

AI versus IA

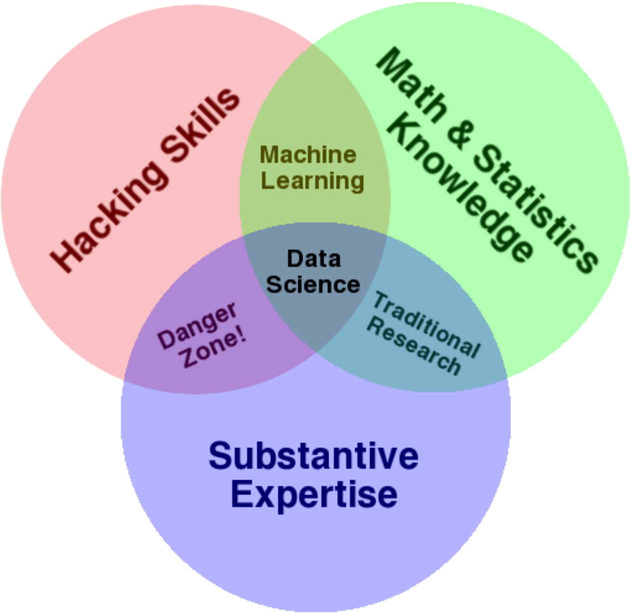
IA = Intelligence augmentation

Does it scale?

Joaquin Quiñonero Candela, Director of Applied Machine Learning at Facebook on Quora:

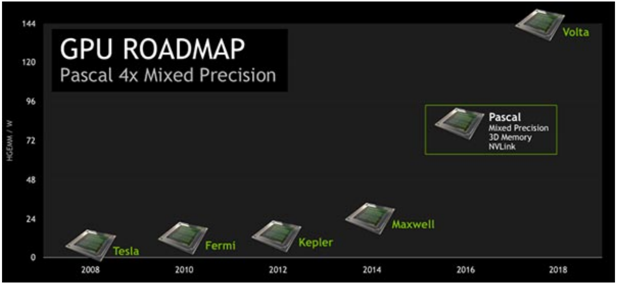
- In computer vision we have a system that processes every single image and video. . . well over 1B items per day.
- . . . predict the content. . . . . . to generate captions for the blind,. . . detect and take down offensive content, improve media search results, automate visual captcha. . . .
- In language technology,. . . translate over 2B posts every single day, with over 1800 language directions representing more than 40 unique languages.
- . . . these models are used to rank news feed stories (1B users every day, 1.5K stories per user per day on average), ads, search results (1B+ queries a day), trending news, friend recommendations and even rank notifications that a user receives, or rank the comments on a post.

Data science - we are getting there



Deep learning prediction about the future

- Expect rapid progress in coming years!
- Machine perception will be at human level
- As you know by now it is still hard to make it work!



References convergence

- Glorot, Xavier, and Yoshua Bengio. ”Understanding the difficulty of training deep feedforward neural networks.” International conference on artificial intelligence and statistics. 2010.
- Martens, J. Deep learning via Hessian-free optimization. In ICML, 2010.
- Saxe, Andrew M., James L. McClelland, and Surya Ganguli. ”Exact solutions to the nonlinear dynamics of learning in deep linear neural networks.” In ICLR 2014.
- Pascanu, R., ”On Recurrent and Deep Neural Networks”, PhD thesis, University of Montreal, 2014.
- Ioffe, Sergey, and Christian Szegedy. ”Batch normalization: Accelerating deep network training by reducing internal covariate shift.” Proc. of ICML, 2015.
- T. Raiko, H. Valpola, and Y. LeCun. Deep Learning Made Easier by Linear Transformations in Perceptrons. Proc. of AISTATS, 2012.
- Y. LeCun, L. Bottou, G. Orr and K. Muller: Efficient BackProp, in Orr, G. and Muller K. (Eds), Neural Networks: Tricks of the trade, Springer, 1998.
- T. Vatanen, T. Raiko, H. Valpola, and Y. LeCun. Pushing Stochastic Gradient towards Second-Order Methods - Backpropagation Learning with Transformations in Nonlinearities. In Proc. ICONIP, 2013.

References regularisation

- Tikhonov, Andrey Nikolayevich (1943). On the stability of inverse problems (in Russian). Doklady Akademii Nauk SSSR 39 (5): 195–198.
- J. Luketina, M. Berglund, and T. Raiko (2016). Scalable Gradient-Based Tuning of Continuous Regularization Hyperparameters. To be presented at the ICLR workshop track. Preprint available as arXiv:1511.06727 [cs.LG].
- Dosovitskiy, A., Springenberg, J. T., Riedmiller, M., & Brox, T. (2014). Discriminative unsupervised feature learning with convolutional neural networks. In Advances in Neural Information Processing Systems (pp. 766-774).
- Sietsma, J. and Dow, R. J. (1991). Creating artificial neural networks that generalize. Neural networks, 4(1), 67–79.
- Rifai, S., Vincent, P., Muller, X., Glorot, X., & Bengio, Y. Contractive auto-encoders: Explicit invariance during feature extraction. In ICML, pp. 833-840, 2011.
- Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean. ”Distilling the knowledge in a neural network.” NIPS 2014 Deep Learning Workshop.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1), 1929-1958.
- Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Science, 313(5786), 504-507.
- Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, Rob Fergus. Intriguing properties of neural networks. International Conference on Learning Representations (2014)
- Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, Shin Ishii. Distributional Smoothing with Virtual Adversarial Training. Under review as a conference paper at ICLR 2016.

Thanks!
Ole Winther