

Deep Nets: Tricks and Tips

Yoshua Bengio

August 29th, 2017 @ DS3

Data Science Summer School

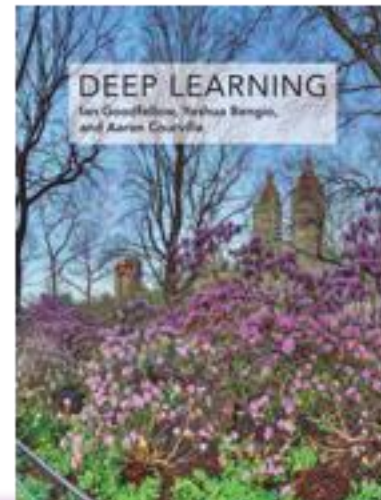


CIFAR
CANADIAN
INSTITUTE
FOR
ADVANCED
RESEARCH

Université 
de Montréal

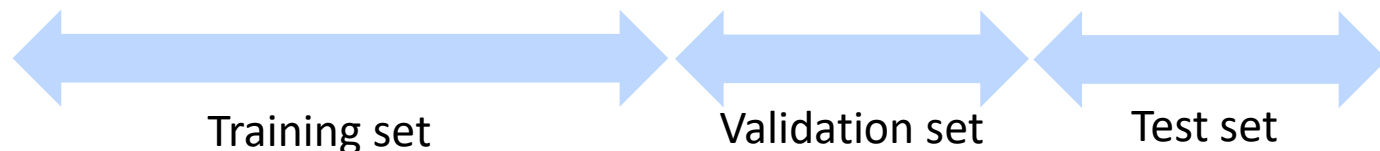


PLUG: **Deep Learning**, MIT Press book is out,
chapters will remain online



Hyper-parameters

- Parameters: optimized by gradient-based optimization on the training set
- Hyper-parameters: design decisions and settings of the optimization procedure
 - Optimized based on performance on a validation set disjoint from training set.
- A disjoint test set is used to obtain final unbiased estimation of generalization performance.
- Training, validation and test sets are subsets of randomized (shuffled) data, to mimic iid assumption



Hyper-parameters of MLPs

- **Global learning rate**
- Number of training epochs (passes over training set)
- Number of neurons per layer
- Depth (number of layers)
- Choice of activation function(s)
- Regularisation coefficients (L1, L2, etc.)
- Noise injection & dropout
- Loss function and output non-linearity
- Minibatch size (with parallel computation within minibatch)
- Weight normalization method (e.g. batch normalization)
- Input and targets normalization
- Data deformations
- Etc.

Nested optimisation of parameters and hyper-parameters

- For each considered configuration of hyper-parameters
 - Train parameters with this configuration (optimize train loss)
 - Measure resulting model's validation error
 - Keep this configuration if it's the best seen up to now
- Optionally: Retrain with training+validation set
- Measure resulting model's test error

Choosing the best = optimizing

- Picking the best hyper-parameter configuration in the given set is a form of optimization (or hyper-optimization)
- Choosing hyper-parameters based on training set would lead to high-capacity choices with overfitting (hence need a validation set)
- Optimizing a particular average taken over a dataset yields a biased (optimistic) value of that average (hence need a test set to obtain an unbiased estimator of generalization ability)
- ➔ We need 3 datasets: training, validation, test

Cross-Validation

- If your training set is very small (e.g. 1000 examples), the training/validation/test split gives too few validation or test examples to obtain statistically significant comparisons
- But each training run is very fast!
- We can repeat the training+test run many times with different subsets of the data, allowing to use more (or all) of the examples as part of the test set (at least once).
- Once such technique is **k-fold cross-validation**

T: part of Training set
V: part of Validation set

Run 1: T T T T V
Run 2: T T T V T
Run 3: T T V T T
Run 4: T V T T T
Run 5: V T T T T

5 subsets
5-fold XV

Sequential Validation

- If the data come from a time-series, and we want to use the predictor from past data to act in the future, we need a different estimator of generalization performance called sequential validation
- We simulate what would have happened if we had had data only up to time t , acting on that trained model over a block of following examples (acting as test), then repeat with increased t and average the test performances over all these blocks.

T V t

T: part of Training set

T T V t

V: part of Validation set

T T T V t

t: part of test set

T T T T V t

Use all the past Vs to select models

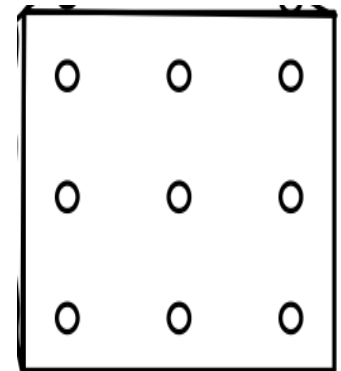
Use all the test blocks to evaluate overall performance

Hyper-Optimization

- Manual search
 - Don't use test error!
 - Slow and sequential, but some trained humans do this better than any machine, for now.
 - Not systematic, harder to reproduce
- Grid search: inefficient with more than 2 hyper-parameters
- Random search (*Bergstra & Bengio, 2012, JMLR*)
 - *Simple, robust & parallelizable*
- Bayesian optimisation (sequential, automated), especially good for non-experts, but still need to set intervals

Grid Search for Hyper-Parameters

- Discretize hyper-parameter values
- Form cross-product of values across all hyper-parameters: the grid
- Launch a trial training + validation error measurement for each element of the grid
- Can be parallelized on a cluster, but may need to redo failed experiments, until all grid is filled
- Exponential in # of hyper-parameters!

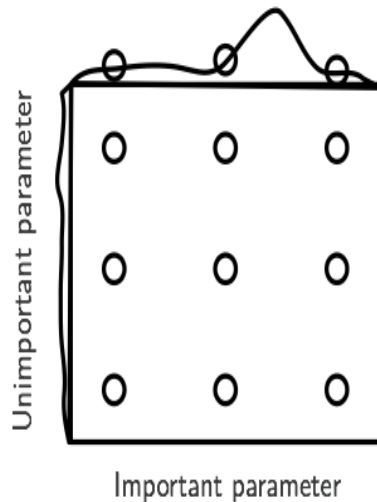


Random Sampling of Hyperparameters (Bergstra & Bengio 2012)

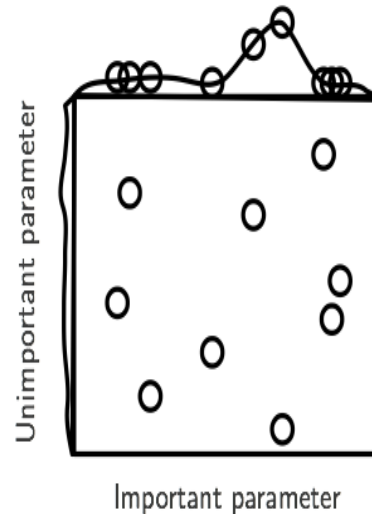


- Random search: simple & efficient
 - Independently sample each HP, e.g.
 $\text{l.rate} \sim \exp(\text{U}[\log(.1), \log(.0001)])$
 - Each training trial is iid
 - If a HP is irrelevant grid search is wasteful
 - More convenient: ok to early-stop, continue

Grid Layout



Random Layout



L1 regularisation to remove weights and inputs

Add a term that pushes weights or groups of weights to 0

$$\text{prediction error} + \lambda \sum_{ij} |W_{ij}|$$

pushes individual weights to 0, whereas

$$\text{prediction error} + \lambda \sum_i \sqrt{\sum_j W_{ij}^2}$$

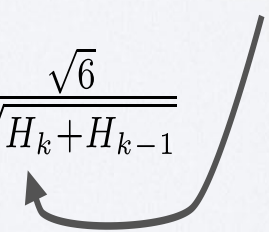
is trying to make all the weights in the group $W_{i,.}$ go to 0

Weight Initialisation

(from
Hugo Larochelle)

Attempts to be
invariant to the
size of the
layers

Topics: initialization

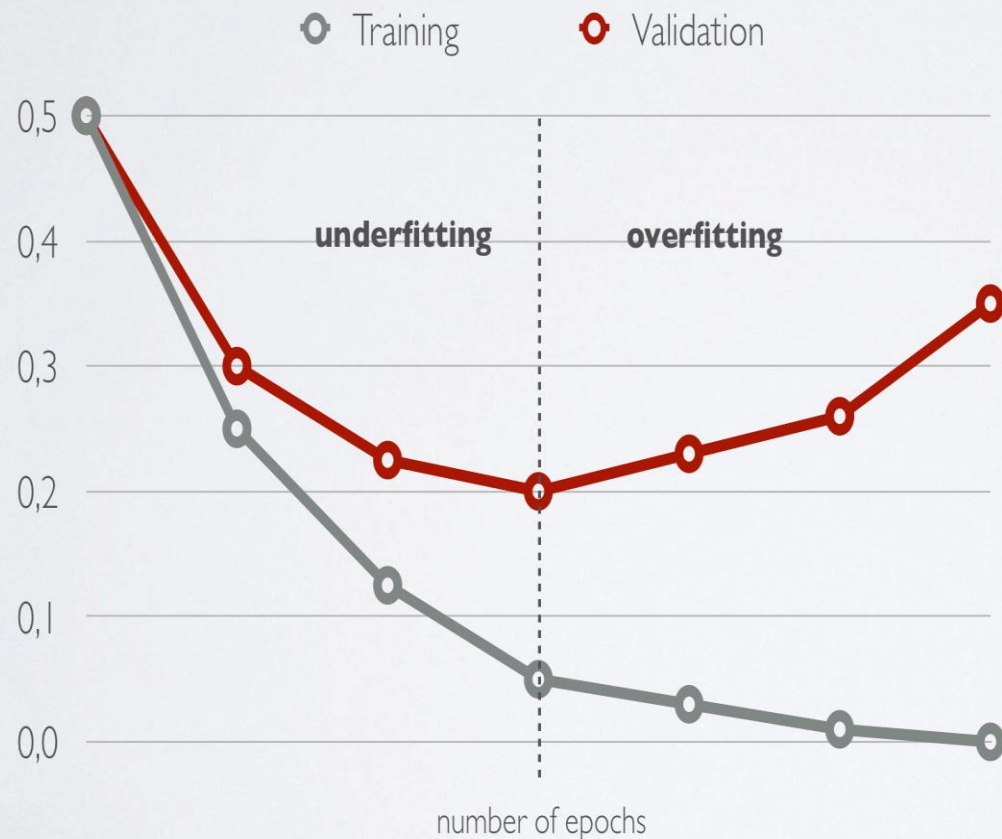
- For biases
 - initialize all to 0
 - For weights
 - Can't initialize weights to 0 with tanh activation
 - we can show that all gradients would then be 0 (saddle point)
 - Can't initialize all weights to the same value
 - we can show that all hidden units in a layer will always behave the same
 - need to break symmetry
 - Recipe: sample $\mathbf{W}_{i,j}^{(k)}$ from $U[-b, b]$ where $b = \frac{\sqrt{6}}{\sqrt{H_k + H_{k-1}}}$
 - the idea is to sample around 0 but break symmetry
 - other values of b could work well (not an exact science) (see Glorot & Bengio, 2010)
- size of $\mathbf{h}^{(k)}(\mathbf{x})$
- 

Early Stopping : free lunch (T jobs for the price of 1)

(from
Hugo
Larochelle)

Topics: early stopping

- To select the number of epochs, stop training when validation set error increases (with some look ahead)



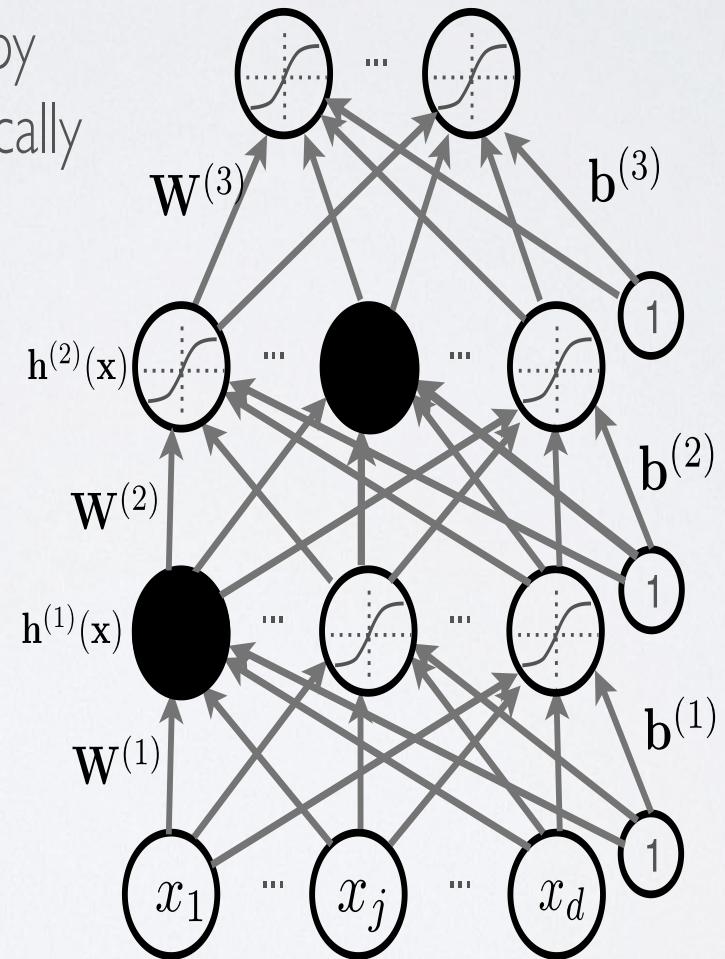
Regularizing by injecting noise: dropout

(from
Hugo
Larochelle)

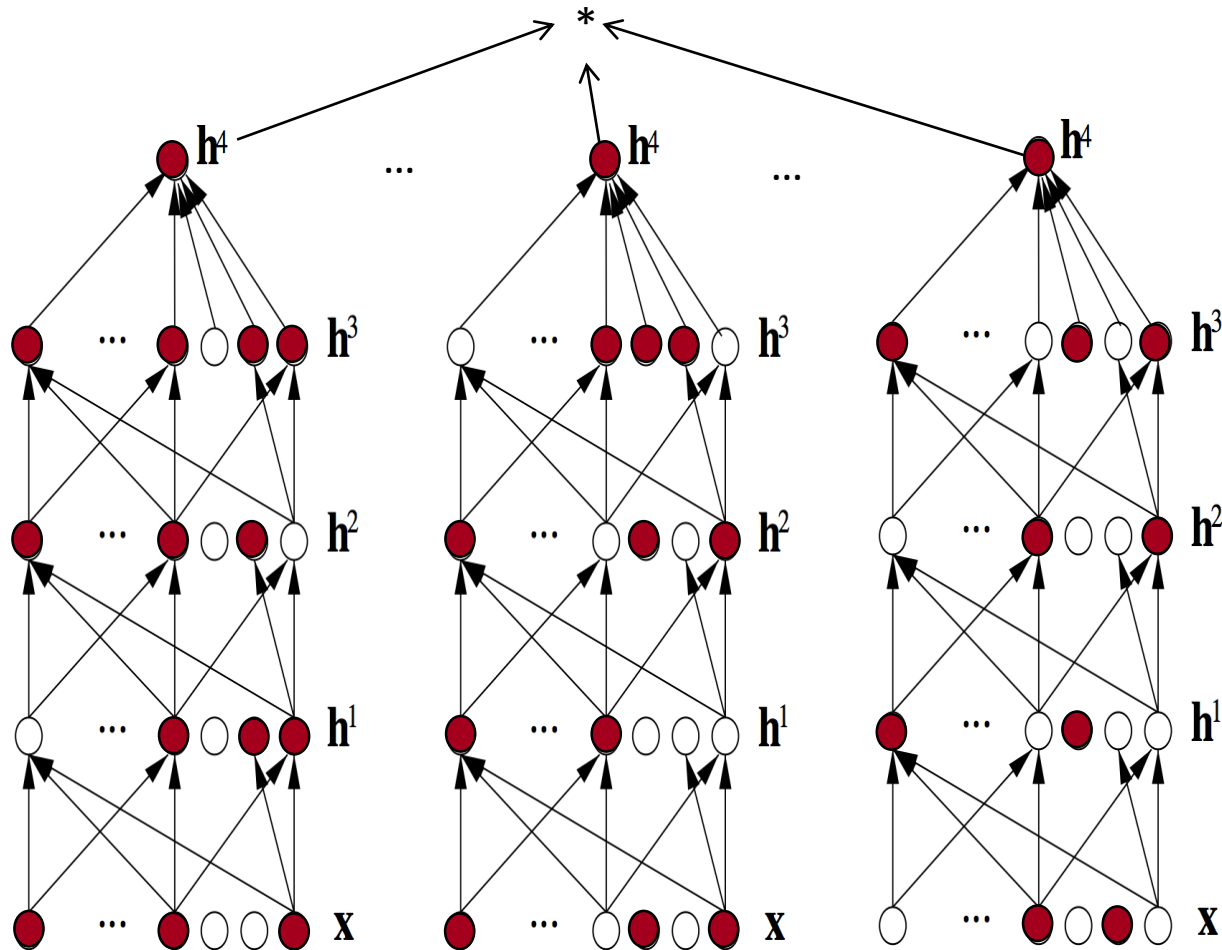
No noise
at test
time.

Topics: dropout

- Idea: «cripple» neural network by removing hidden units stochastically
 - each hidden unit is set to 0 with probability 0.5
 - hidden units cannot co-adapt to other units
 - hidden units must be more generally useful
- Could use a different dropout probability, but 0.5 usually works well



Dropout Regularizer: Super-Efficient Bagging



Diagnostic: overfitting vs underfitting?

(from
Hugo
Larochelle)

Topics: why training is hard

- Depending on the problem, one or the other situation will tend to prevail
- If first hypothesis (underfitting): use better optimization
 - this is an active area of research
- If second hypothesis (overfitting): use better regularization or collect more data
 - unsupervised learning or semi-supervised
 - stochastic «dropout» training

How to know if you are overfitting or underfitting?

Overfitting: if you increase capacity (number of parameters, training time, better optimizer, smaller regularization coefficient, etc.), test or validation error increase



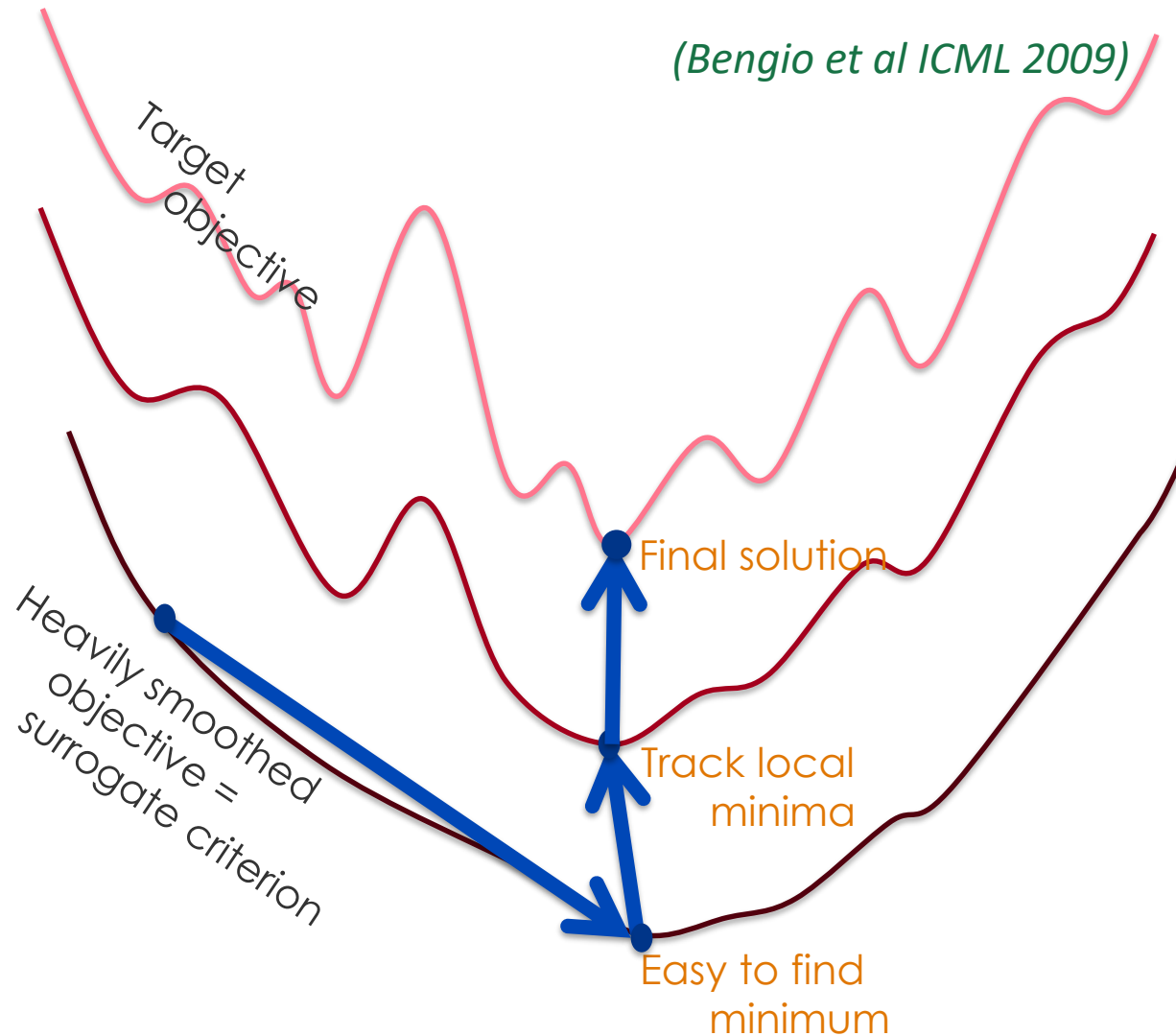
Curriculum Learning

Guided learning helps training humans
and animals



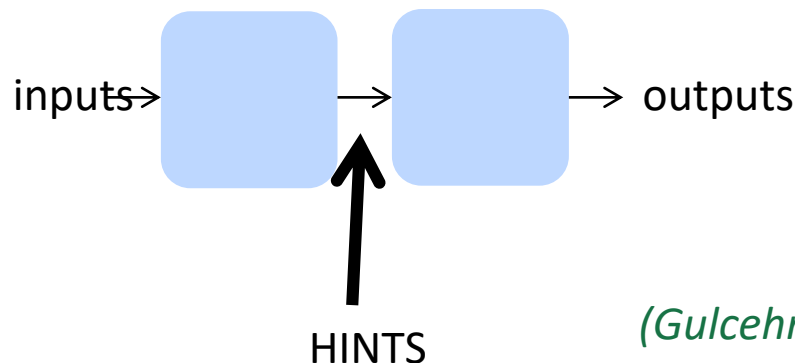
Start from simpler examples / easier tasks
(Piaget 1952, Skinner 1958)
(Bengio et al ICML 2009)

Curriculum learning as a Continuation Method



Guided Training, Intermediate Concepts

- Breaking the problem in two sub-problems and pre-training each module separately, then fine-tuning, nails it
- *Need prior knowledge to decompose the task*
- **Guided pre-training** allows to find much better solutions, escape effective local minima



(Gulcehre & Bengio ICLR'2013)

Debugging

- Instrument the code to make experiments reproducible
- Use tools to verify gradients (finite differences)
- **Train on a small dataset:** verify can reach 0 training error
- Track error curves during training (training error, validation error); training error should roughly go down
- Track distribution statistics of weights and gradients during training

Validate and Analyze

- Vary capacity and observe error curves to identify if the system is rather overfitting or rather underfitting
- Compare with simpler reference models (logistic regression, SVMs, random forests)
- Track several relevant metrics
- Look at the training and validation examples which give the worse error (input, output and target)
- Measure the input of changing the number of training examples
- Make sure you have enough test examples to be able to conclude with statistical significance