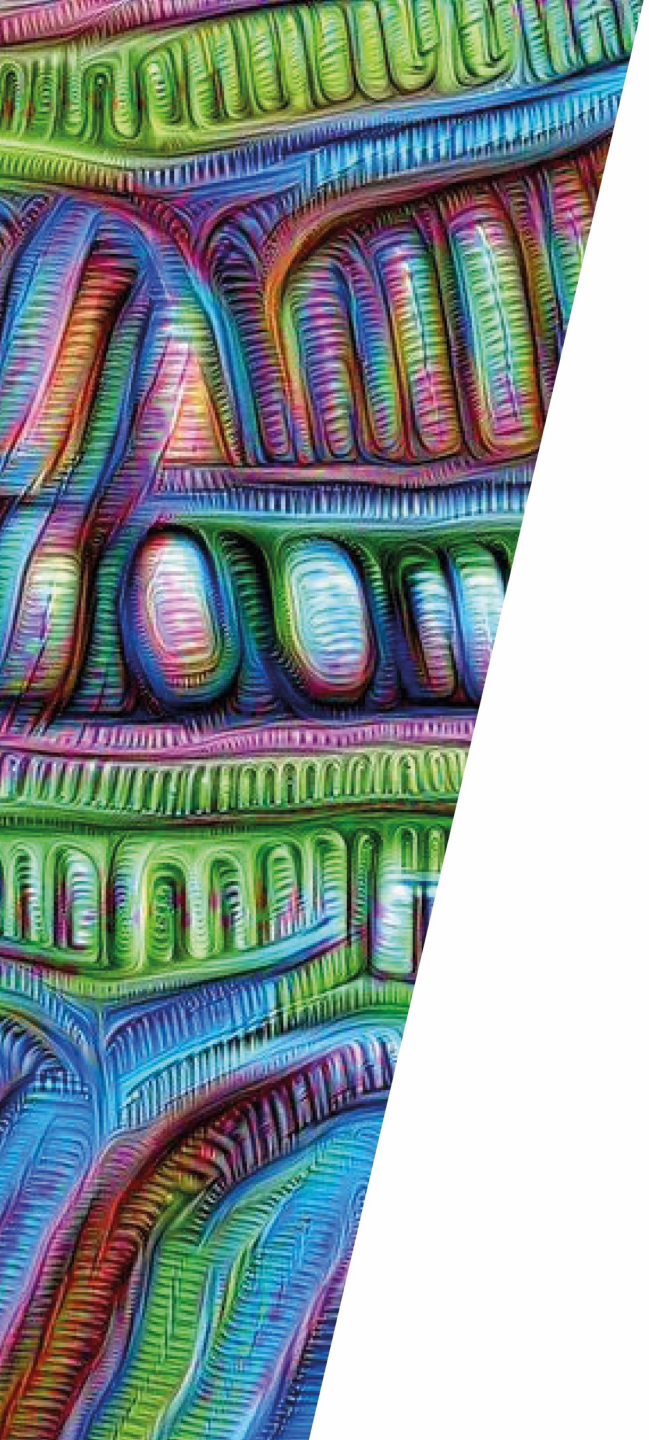


Bienvenue!

ÉCOLE D'ÉTÉ FRANCOPHONE EN APPRENTISSAGE PROFOND

21-25 août 2017

IVADO

HEC Montréal
Polytechnique Montréal
Université de Montréal

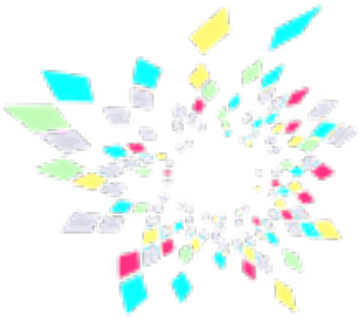
MILA

Conseils pratiques pour l'entraînement des réseaux profonds

Yoshua Bengio

23 août 2017

École d'Été en Apprentissage Profond



Hyper-paramètres

- Les paramètres sont choisis par descente de gradient sur l'ensemble d'entraînement
- Mais beaucoup de décisions de design doivent être fixées empiriquement, comme la capacité (taille des couches et profondeur), le gain du gradient (learning rate), et les coefficients de différents régularisateurs.
- On utilise normalement un ensemble de validation pour choisir les valeurs des hyper-paramètres
- Ensuite on peut utiliser l'ensemble de test pour évaluer la performance finale et comparer à un banc d'essai

Hyper-paramètres des MLPs

- **Learning rate global**
- Nombre d'époques (itérations sur l'ensemble d'entraînement)
- Nombre de neurones par couche
- Profondeur
- Choix des fonctions d'activation
- Coefficients de régularisation (L1, L2, etc.)
- Injection de bruit et dropout
- Fonction de perte et non-linéarité de sortie
- Taille des minibatch (paquet d'exemples traités en parallèle)
- Méthode de normalisation des poids (batch norm, etc.)
- Méthode de normalisation des entrées et cibles
- Méthode de déformation des données pour mieux généraliser
- Etc.

Optimisation imbriquée des paramètres et des hyper-paramètres

- Pour chaque configuration des hyper-paramètres considérée
  - Faire un entraînement avec cette configuration, i.e., optimiser les paramètres selon une erreur calculée sur l'ensemble d'entraînement
  - Mesurer l'erreur du modèle résultant sur l'ensemble de validation
  - Si c'est la meilleure vue jusqu'à présent, sauver ce modèle et la configuration des hyper-paramètres comment étant la meilleure.
- Mesurer l'erreur du modèle obtenu avec les meilleurs hyper-paramètres trouvés, mais sur l'ensemble de test

Choisir = optimiser

- La procédure de choix des hyper-paramètres est une forme d'optimisation (ou hyper-optimisation, ou méta-optimisation)
- Quand on optimise un critère sur un jeu de donnée, on obtient généralement une mesure biaisée (optimiste) de ce critère
- Si on choisissait les hyper-paramètres selon l'ensemble d'entraînement on choisirait les valeurs qui maximise la capacité, ce qui favoriserait le sur-apprentissage (overfitting)
- C'est pour ça qu'on a besoin de 3 ensembles de données

Validation croisée

- Si l'ensemble d'entraînement est très petit (e.g. 1000 exemples), la division entraînement/validation/test donne un trop petit nombre d'exemples de test pour obtenir des comparaisons statistiquement significatives.
- D'un autre côté, chaque entraînement est rapide.
- On peut répéter l'entraînement + test de nombreuses fois avec des partitions différentes des données, de façon à ce que chaque exemple original serve une fois comme exemple de test: **k-fold cross-validation**

Validation séquentielle

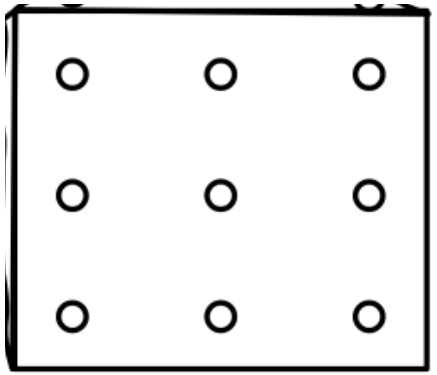
- Si les données représentent un historique, une série temporelle, et que l'on veut construire un prédicteur qui sera utilisé pour des données futures, il faut utiliser une méthode particulière d'estimation de l'erreur future
- On va simuler ce qui serait arrivé si on avait eu accès seulement aux données jusqu'au temps t pour l'entraînement et qu'on avait ensuite utilisé ce prédicteur pour la période suivante.
  - On répète cela pour beaucoup de blocs entraînement/test correspondant à beaucoup de points t dans la séquence.

Chercher dans l'espace des hyper-paramètres

- Recherche manuelle
  - Attention à ne pas faire une recherche manuelle en se fiant sur l'erreur de test, surtout si cet ensemble est petit!
  - Séquentiel et long, mais le cerveau humain peut être entraîné à faire cela… mais ça rend les expériences moins reproductibles.
- Recherche sur une grille: inefficace en haute dimension (plus que 2 hyper-paramètres)
- Recherche aléatoire *(Bergstra & Bengio, 2012, JMLR)*
  - *Simple, robuste et parallélisable*
- Bayesian optimisation (séquentielle, automatisée)

Grid Search for Hyper-Parameters

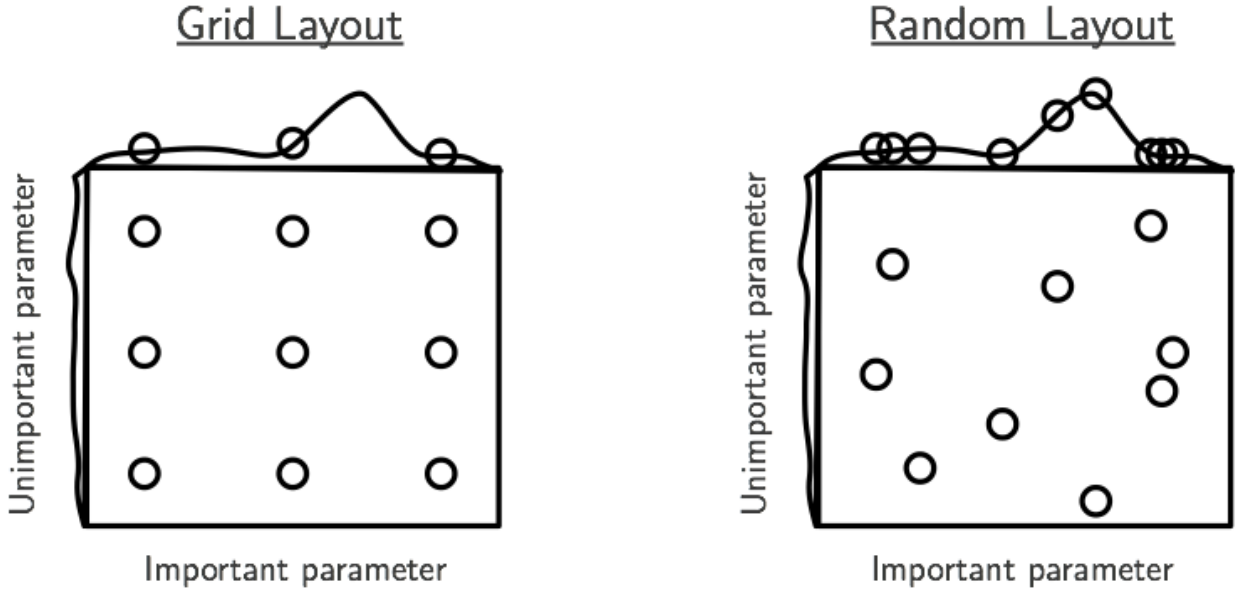
- Discretize hyper-parameter values
- Form cross-product of values across all hyper-parameters: the grid
- Launch a trial training + validation error measurement for each element of the grid
- Can be parallelized on a cluster, but may need to redo failed experiments, until all grid is filled
- Exponential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



Random Sampling of Hyperparameters

(Bergstra & Bengio 2012)

- Random search: simple & efficient
  - Independently sample each HP, e.g. l.rate~exp(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 further, etc.



Régularisation L2

On rajoute un terme quadratique en W au critère d'entraînement

erreur de prédiction + $\lambda \sum_{ij} W_{ij}^2$

On ne pénalise pas les bias, seulement les poids.

Ça indique qu'on préfère les solutions avec des ***petits poids***.

Régularisation L1

On rajoute au critère d'entraînement un terme proportionnel à la valeur absolue des poids :

erreur de prédiction + $\lambda \sum_{ij} |W_{ij}|$

On ne pénalise pas les bias, seulement les poids.

Ça indique qu'on préfère les solutions avec des ***petits poids*** et plusieurs d'entre eux vont être **poussés à être 0**.

Initialisation des poids: pas une science exacte mais fait une différence

(du cours de Hugo Larochelle)

Initialisation invariante à la taille de chaque couche!

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 $U\left[-b, b\right]$ where $b = \frac{\sqrt{6}}{\sqrt{H_k + H_{k-1}}}$

size of $\mathbf{h}^{(k)}(\mathbf{x})$

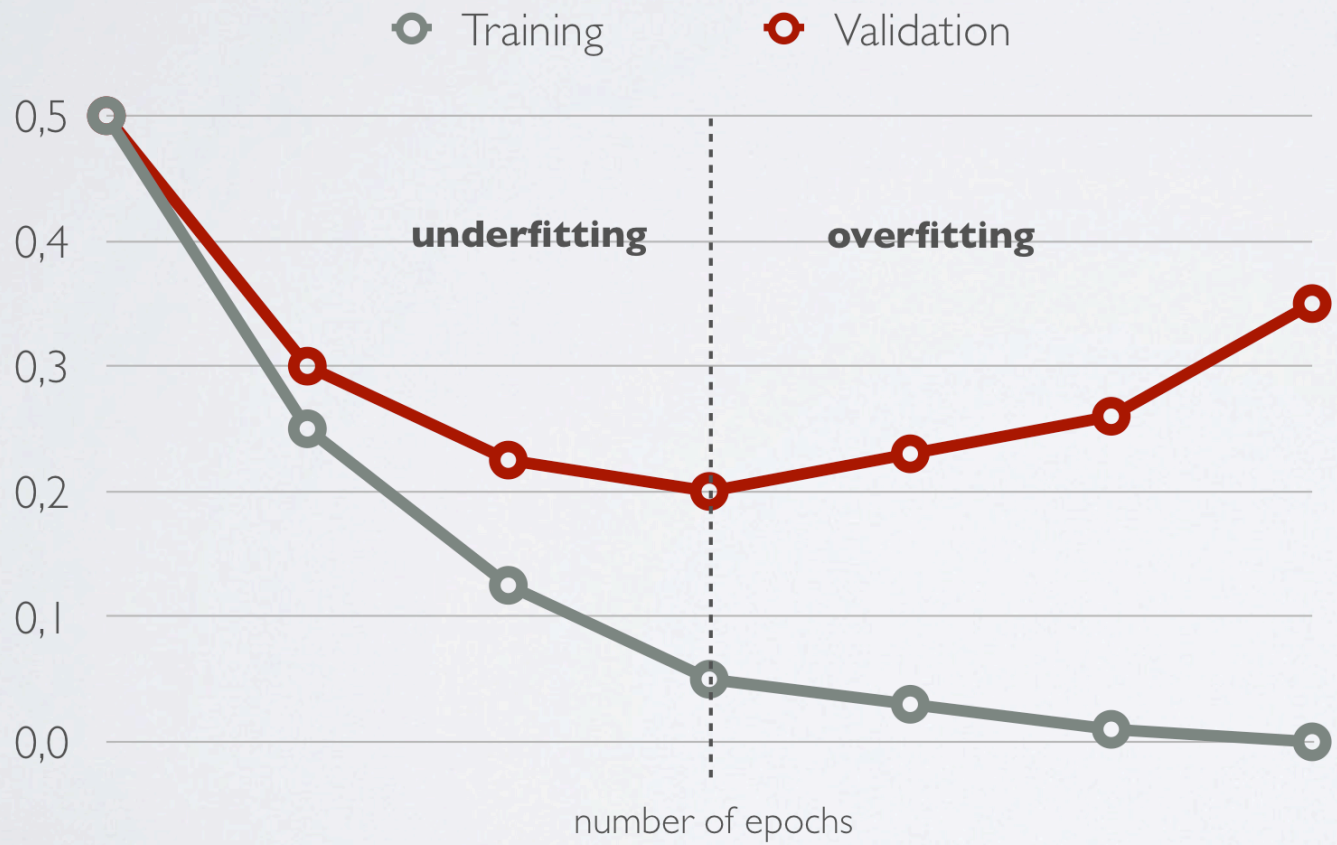
    - the idea is to sample around 0 but break symmetry
    - other values of ***b*** could work well (not an exact science) (see Glorot & Bengio, 2010)

Early Stopping (Arrêt prématuré): free lunch (↑ expériences pour le prix de 1)

(du cours de Hugo Larochelle)

Topics: early stopping

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

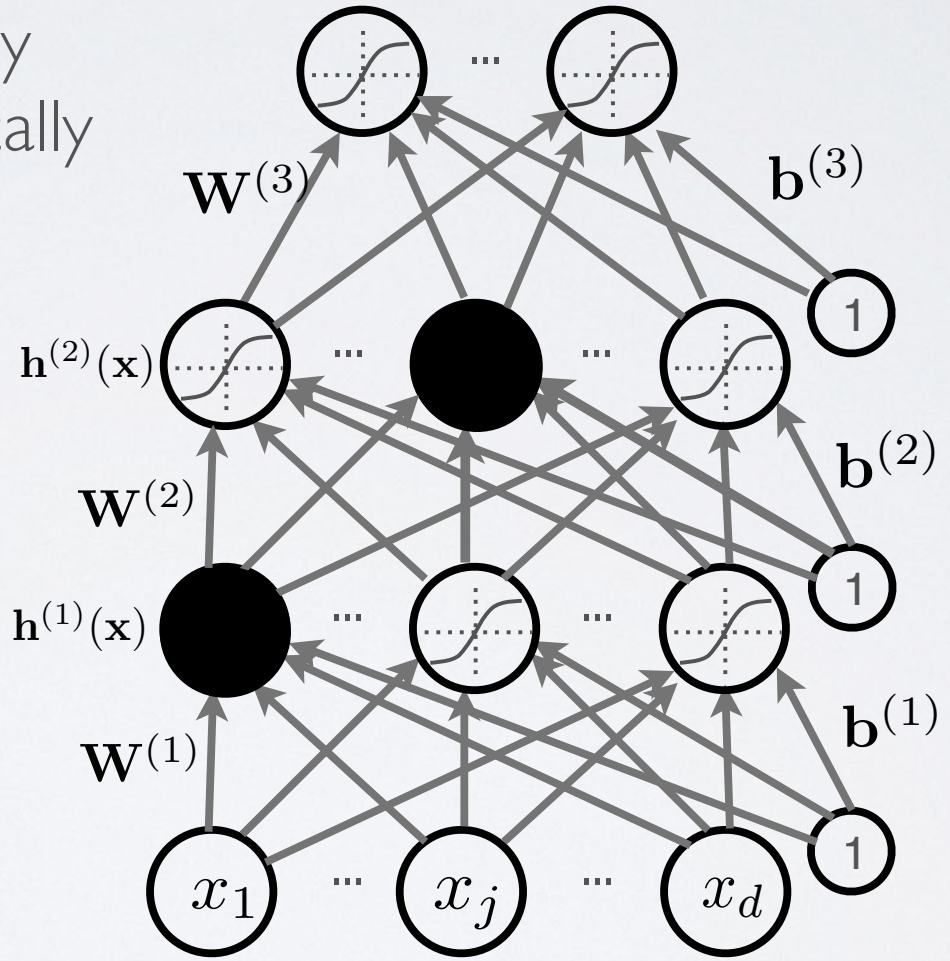


Régulariser en injectant du bruit: dropout

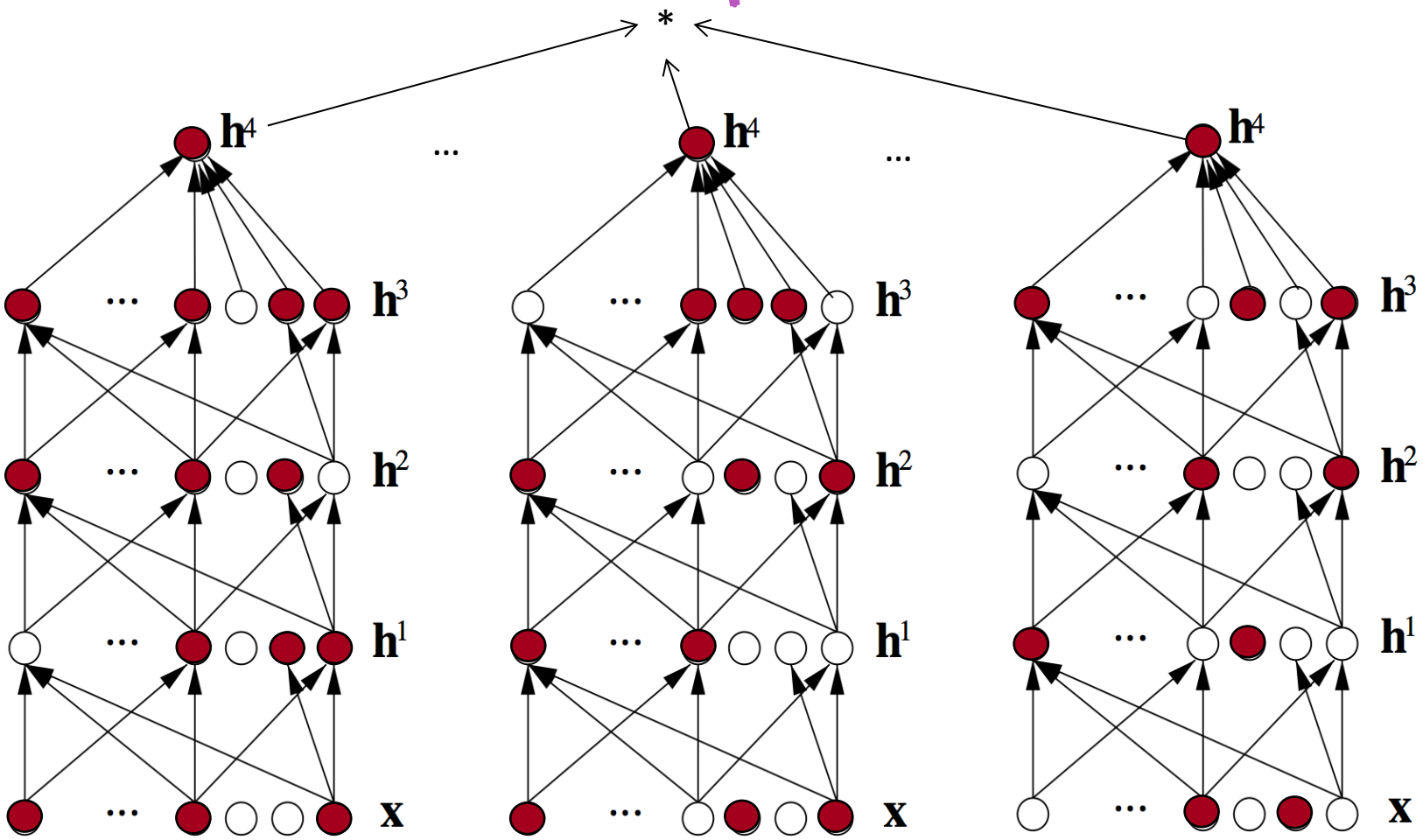
(du cours de Hugo Larochelle)

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



Dropout: entraînement vs test

- Pendant l'entraînement avec dropout, on peut appliquer la rétropropagation comme d'habitude. Le bruit est vu comme une entrée supplémentaire (et implanté comme ça!).

- Pour utiliser ou tester un réseau entraîné par injection, on enlève le bruit. Pour le dropout, en remplace le masque binaire 0-1 par son espérance (habituellement 0.5).

Diagnostic: overfitting vs underfitting?

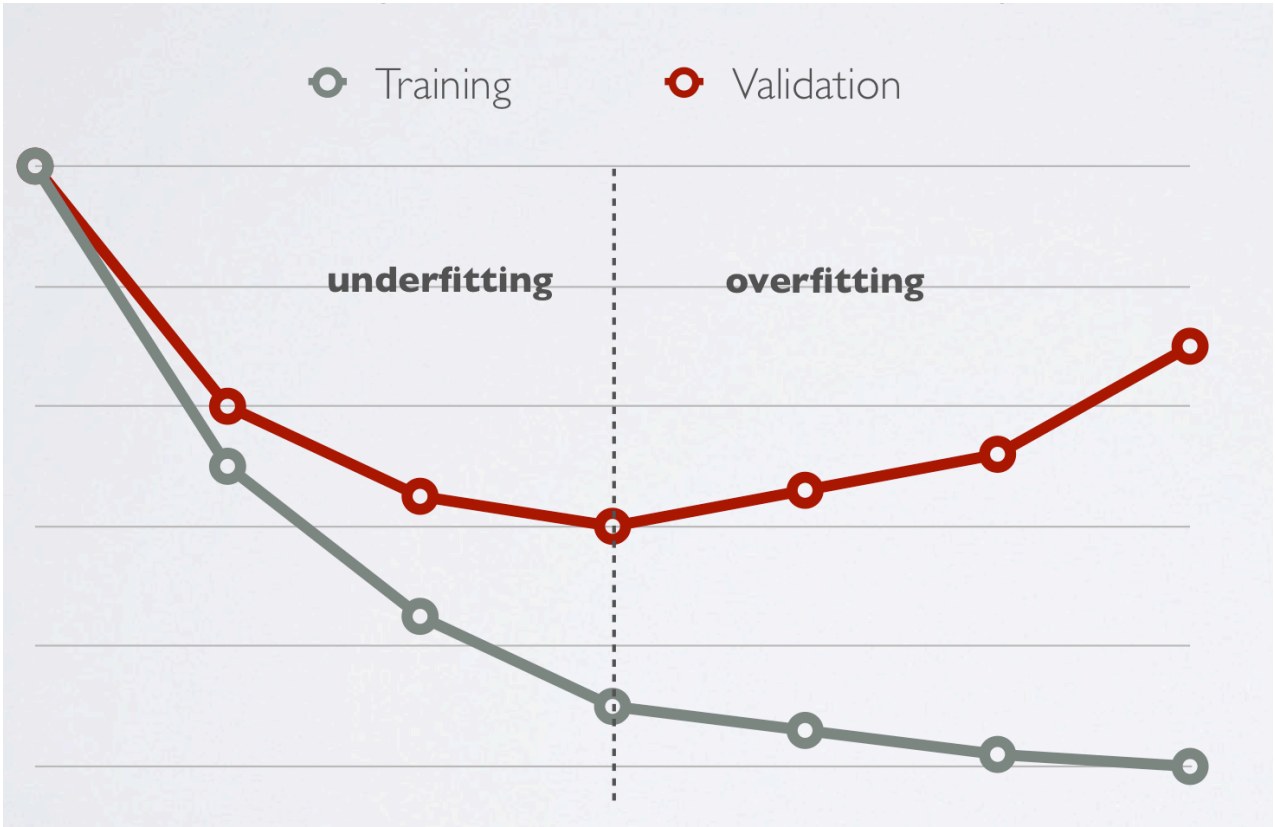
(du cours de Hugo Larochelle)

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

Comment savoir si on est en overfitting ou underfitting?

Overfitting: si on augmente la capacité (nombre de paramètres, temps d'entraînement, moins de régularisation, meilleur optimiseur, etc.), l'erreur de test ou validation augmente.



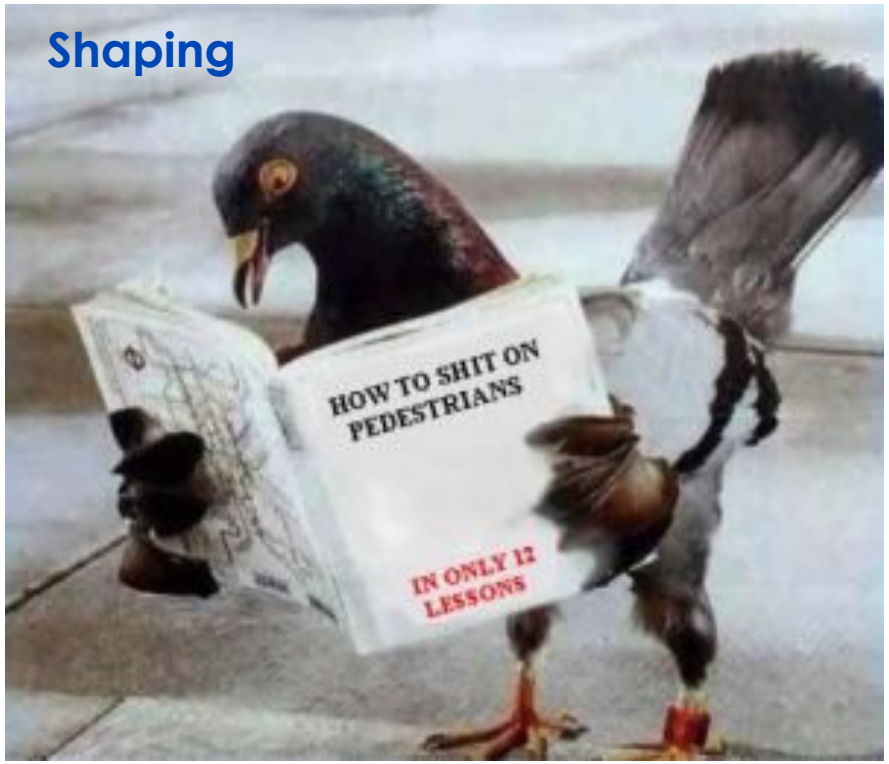
Capacité optimale

- Pour un nombre donné d'exemples, il y a une capacité optimale qui minimise l'erreur de test
- Plus on augmente le nombre d'exemples d'entraînement et plus la capacité optimale augmente

- Sur un très petit jeu de données, on est limité à un réseau très petit **ou très régularisé** qui apprend une fonction très simple

Curriculum Learning

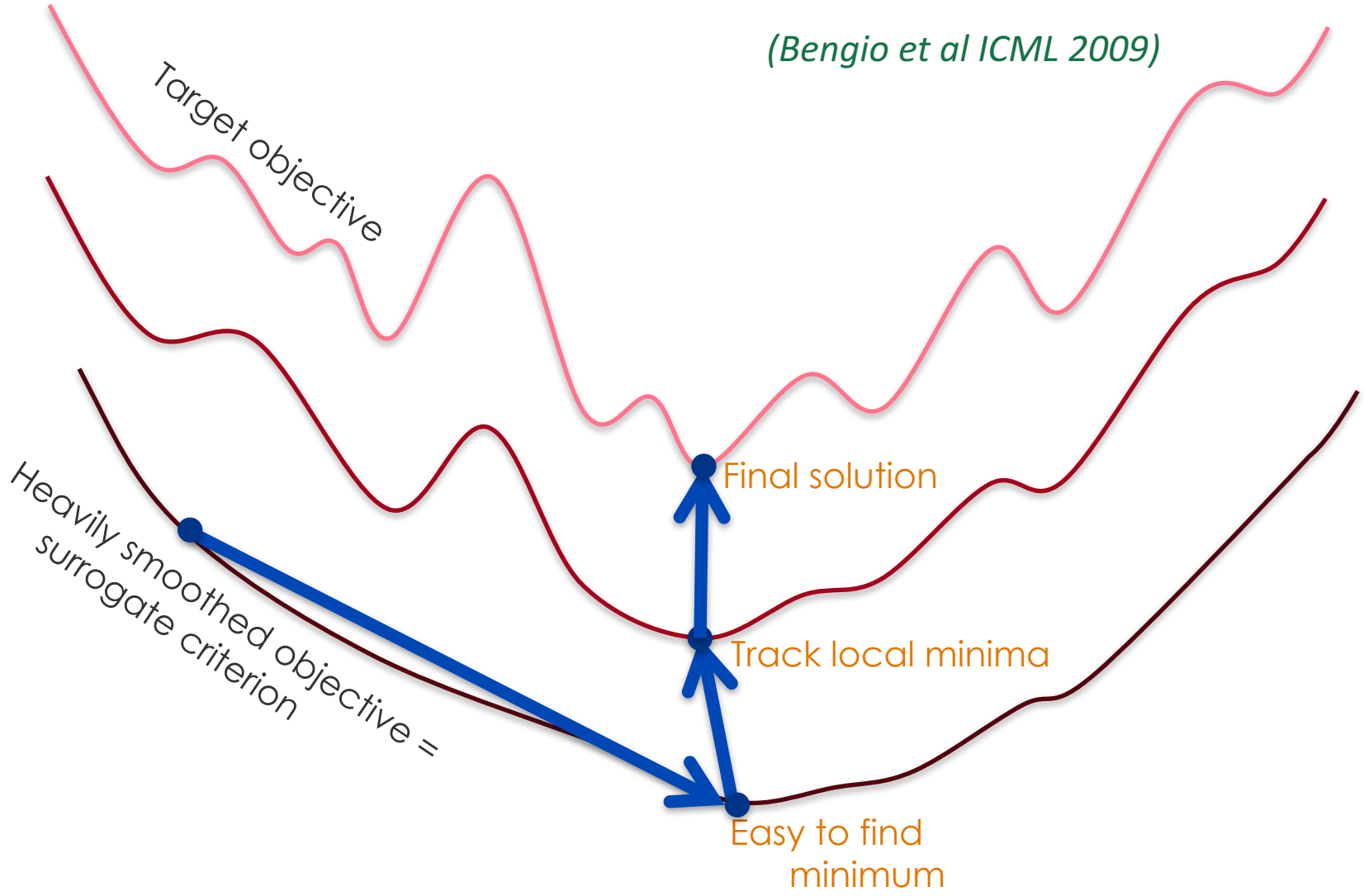
Guided learning helps training humans and animals



Start from simpler examples / easier tasks (Piaget 1952, Skinner 1958)

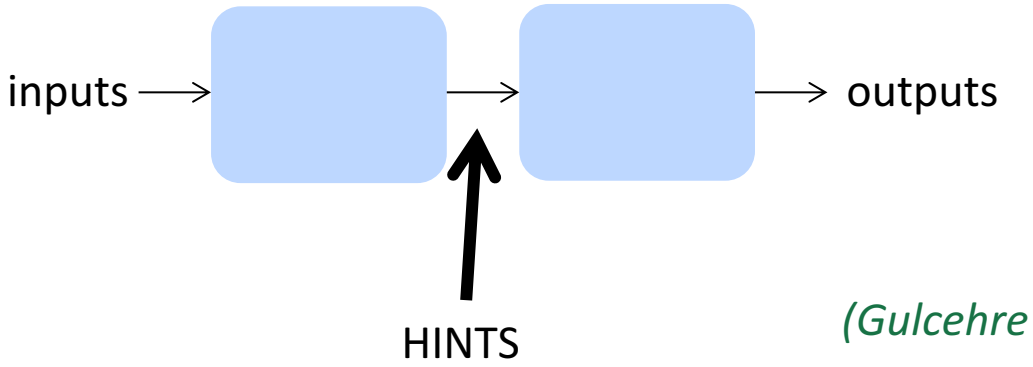
(Bengio et al ICML 2009)

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

- Instrumenter le code pour qu'il soit reproductible (seeds contrôlés)
- Outils pour vérifier numériquement le gradient rétropropagé (par différences finies)
- Entraîner sur un petit jeu de données et vérifier qu'on peut obtenir 0 erreur d'entraînement
- Traquer les courbes d'erreur pendant l'apprentissage (entraînement, validation); l'erreur d'entraînement devrait grosso-modo descendre!
- Traquer les distributions des poids et des gradients pendant l'apprentissage

Valider et Analyser

- Varier la capacité et observer les courbes d'erreur pour identifier si le système est plutôt en overfitting ou underfitting
- Comparer avec un modèle de référence simple (régression logistique, SVM, etc.)
- Traquer plusieurs métriques de performance importantes
- Regarder les exemples qui donnent des erreurs les plus grandes (entrées, sorties, et cibles)
- Mesurer l'impact d'une variation de la quantité de données d'entraînement
- S'assurer d'avoir assez de données de test pour pouvoir conclure statistiquement de l'avantage apporté par un nouveau modèle