Regularization measures

Large number of parameters tend to overfit
$\Rightarrow$ need to prevent over fitting

* Dropout:
  - at each training stage drop out units in fully connected layers with probability of $(1-p)$, where p is hyperparameter
  - Removed nodes are reinstated with original weights in the subsequent stage.

Regularization measures

* Data augmentation:
  - increase variability in training data
  - Perturb existing data (e.g. crop images in different ways)

* Early stopping:
  - stop training before learning completes

Batch normalization

- Makes sure that activations are not saturated
- Normalization values are computed for each batch in training
- Normalization is differentiable (suitable for backpropagation)
- In addition to normalization allow for some shift and scaling to support some saturation and end training
- Batch normalization is not needed after each layer

batch outputs for current layer $\{z^{(i)}\}_{i=1}^{q} \rightarrow \{\hat{z}^{(i)}\}_{i=1}^{q}$

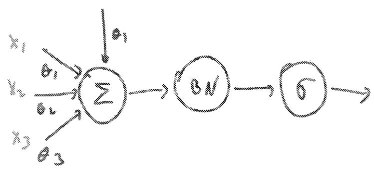
$$\hat{z}_j^{(i)} = \frac{z_j^{(i)} - \mu_j}{\sigma_j} \qquad \mu_j = \frac{1}{q}\sum_{i=1}^{m} z_j^{(i)}$$

$$\sigma_i = \left(\frac{1}{q}\sum_{i=1}^{m}\left(z_j^{(i)} - \mu_j\right)^2\right)^{1/2}$$

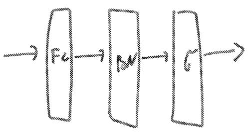
output of j-th unit for i-th batch example

Batch normalization

* Usually normalize before activation:



* Layers:



FC = Fully connected or convolutional layer without activation
BN = Batch normalization
σ = NL activation

* sometimes applied after activation.

Ensemble classifiers

* Ensemble classifier:
  - Train multiple independent models
  - Use majority vote or average during testing
* Reduces overfitting
* To obtain multiple models:
  - change data
  - change parameters
  - Record multiple snapshots of the model during training.

Summary

* start without regularization
* add Batch normalization
* add dropout if needed
* add decay if needed
* augment data if needed
* create ensemble if needed

if observing overfitting

Hyperparameter optimization

* principle:
  - select a set of parameters
  - train on training data and test on validation data using a few epochs
  - modify parameters in search of better validation error.
  - If loss is increasing (e.g. x2) stop

Hyperparameter optimization

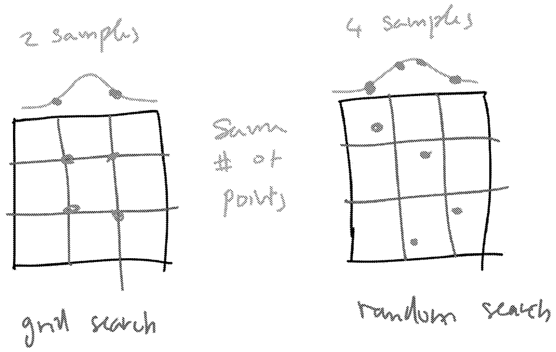
* Methods:
  - Coordinate search
  - Grid search
  - Random search
  - Bayesian model-based optimization
  - Evolutionary

Hyperparameter optimization

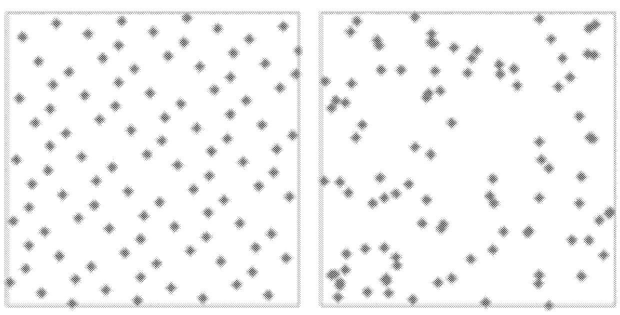
* Search procedure:
  - hold all parameters fixed and change one.
  - change parameter on a logarithmic scale to cover both small and large values (e.g. x2 or x10 values)
  - once identifying best parameter search around it (coarse-to-fine search) using more epochs
  - If best parameters are found at end of range extend range.
  - Repeat several passes on all parameters

Hyperparameter optimization

* Grid and random search
  - Search combinations of parameters!



Hyperparameter optimization



quasi-random sampling (low discrepancy sequence) | random sampling

=> Better coverage

Hyperparameter optimization

* Optimize parameters in this order:
  - learning rate (value, decay, update procedure)
  - regularization (batch norm, dropout, L2)
  - Network architecture and network size
  - other:
    - optimizer
    - optimizer parameters
    - activation
    - initialization

GPU frameworks

* Training is time consuming ⇒ use GPUs
* Frameworks:

    Tensorflow

    Keras

    Pytorch