FNN Feed forward neural network.

- 1980- ANN 2006 Auto-encoder
2012 Relu.

Softmax layer: output a probability for each class.

- multiple classes: one-hot encoding for labels.

- preprocessing: < learn faster: SGD converges faster.
  - min-max normalization: $x' = \frac{x - x_{min}}{x_{max} - x_{min}}$ (when we know the max)
  - standardization: $z_i = \frac{x_i - \mu}{\sigma} \sim N(0,1)$

- overfitting: • Dropout
  • Regularization
  •

- parameters: - Depth: < 1000
  - Width: 5-30
  - Connectivity: usually fully connected

- Activition function: - provide non-linearity.
  - Relu (overcome vanishing gradient).

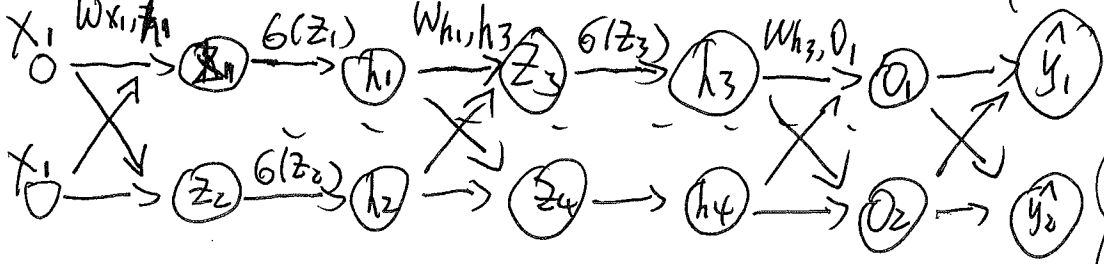
- Loss function: - Classification: cross entropy: $-\sum_{i=1}^{K} y_i \log \hat{y}_i$
  - Regression: squated loss: $\sum_{i=1}^{K} (y_i - \hat{y}_i)^2$

- Training of FNN.

- Minibatch: · Devide data into minibatches → (iterations)
  · Go through the whole sample (epoch)
  · Update weights after each iteration.
- SGD: $w_\ell \leftarrow w_\ell - \eta \cdot \frac{\partial L}{\partial w_\ell}$.
- learning rate: · Constant
  · $\eta \leftarrow \eta / \sqrt{t+1}$ (AdaGrad), t - epoch.
  · weight-based $\eta_w = \frac{\eta}{\sqrt{\sum g_t^2}}$. $g_\ell = \frac{\partial L}{\partial w}$.

- Dropout: · only in traning (p)
  · in testing, reweight $*(1-p)$
  "ensemble" models.

- Back ward propagating (update the weights using chain rule for derivatives).



$$L = -\sum_{i=1}^{2} y_i \log \hat{y}_i$$

$$\hat{y}_1 = \frac{e^{o_1}}{e^{o_1}+e^{o_2}}$$

$$\hat{y}_2 = \frac{e^{o_2}}{e^{o_1}+e^{o_2}}$$

forward pass: When doing prediction, record terms in the chain that do not depend on loss.

backward pass: Calculate the terms that depend on loss and pass them back to update weights.

* Vanishing gradient ....

RNN - sequence data.

- LSTM

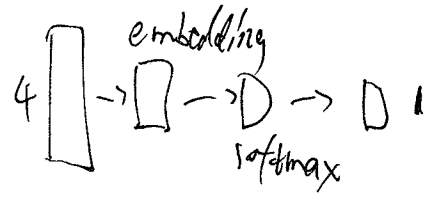
NN with memory

- Auto-encoder.
  - unsupervised learning
  - non-linear compression. (embedding).

PCA, SVD (linear)

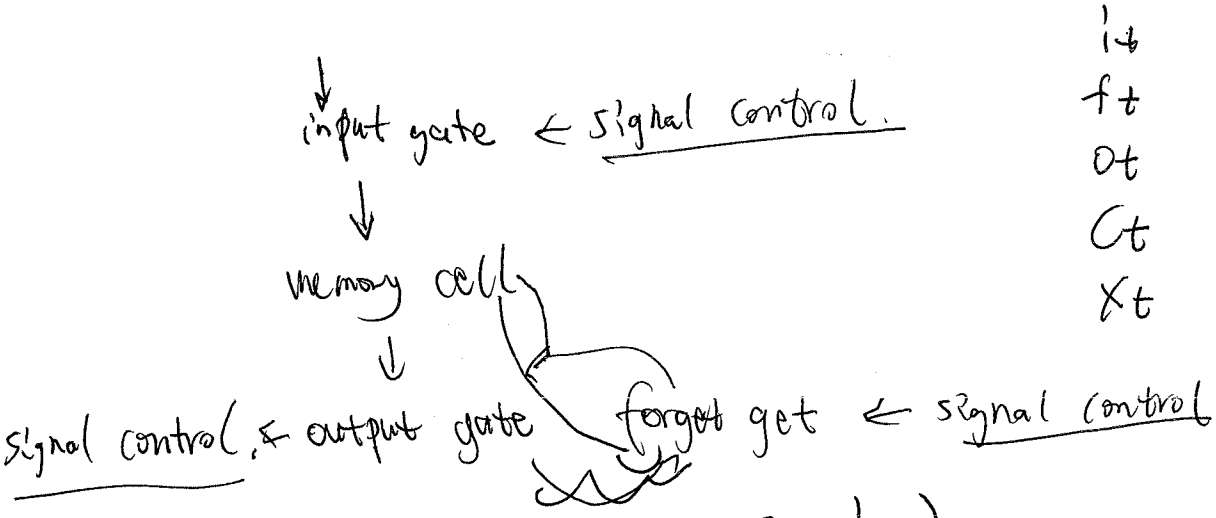
- used to initialize parameters for NN. (Early time).
- Trained using RBM: $e^{E(v,h)} = -\sum_i b_i v_i - \sum_j b_j h_j - \sum_{i,j} v_i h_j w_{ij}$.

- Word embedding.
  - New representation (vector) of the original data with certain meaning (semantics).
  - CBOW : 5 words, with the middle as label.
  - (more popular) Skip-gram : 5 words, with the middle word as feature, others as labels



LSTM.

— LSTM. memory to store previous NN output.



i_t
f_t
o_t
C_t
X_t

$$\begin{cases} i_t = \sigma(W_i[h_{t-1}, X_t] + b_i) \\ f_t = \sigma(W_f[h_{t-1}, X_t] + b_f) \\ o_t = \sigma(W_o[h_{t-1}, X_t] + b_o) \end{cases}$$

$$\hat{C}_t = \tanh[W_c[h_{t-1}, X_t] + b_c], \quad C_t = i_t \cdot \hat{C}_t + f_t \cdot C_{t-1}$$

$$\boxed{h_t = o_t \tanh(C_t)} \rightarrow \text{final output.}$$

CNN

- capture local patterns - subsampling
- less parameters (weight sharing).

- Structure: data → convolution → max-pooling (repeat) → flatten → FFNN

① local patterns are smaller
2nd convolution ② patterns in different regions.

subsampling
does not change the object.

- Data augmentation (generate more samples, reduce overfitting).
  - rotation
  - random cropping
  - add random noises.

  objects are still there.

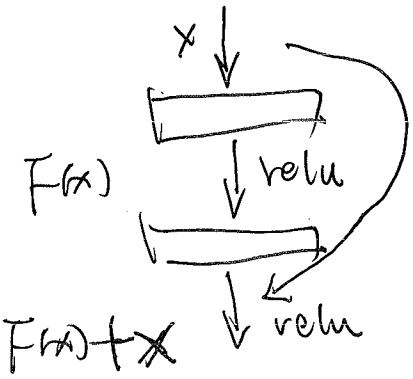
- Convolutional layer:

  filter ("matrix dot product") → feature map
  - filter size: 5-13
  - #: sample/para.
  - stride: 1
  - zero padding: feature map size = original data size.

- Max-pooling: new representation (image), smaller input data size.

CNN variants (~~original~~ normal is called AlexaNet).

- Inception Net : irregular filter sizes.
  branches
- Residual Net: deeper (overcome vanishing gradient).



highway
skip connection

(LSTM similar if $C_t = 1 \cdot C_{t-1} + 0 \cdot \hat{C}_t$)

- Style transferring (styles are also patterns)

$$L_{total} = \alpha L_{content} + \beta L_{style}.$$

- Devise (Multi-model model).

source word —skip-gram→ embedding vector

image —CNN→ embedding vector
repeat { convolution, max-pooling

> Combine using similar metric.