CutSize = J = 1/4 $\sum_{i,j}(q_1 - q_j)^2 w_{ij}$

$$\sum_{i=1}^{n} q_i^2 = n$$

$$d_i \equiv \sum_j w_{i,j}$$

$$D \equiv \left[d_i \delta_{i,j}\right]$$

$$J = \frac{1}{4}\sum_{i,j}(q_i - q_j)^2 w_{i,j} = \frac{1}{4}\sum_{i,j}\left(q_i^2 + q_j^2 - 2q_i q_j\right) w_{i,j}$$

$$= \frac{1}{4}\sum_i 2q_i^2\left(\sum_j w_{i,j}\right) - \frac{1}{4}\sum_{i,j} 2q_i q_j w_{i,j}$$

$$= \frac{1}{2}\sum_i q_i^2 d_i - \frac{1}{2}\sum_{i,j} q_i q_j w_{i,j} = \frac{1}{2}\sum_i q_i\left(d_i\delta_{i,j} - w_{i,j}\right) q_j$$

$$J = 1/2\,\mathbf{q}^T(\mathbf{D} - \mathbf{W})\mathbf{q}$$

1xn 向量 nxn 矩阵

$$f'Lf = f'Df - f'Wf$$

把矩阵运算写成数值运算的形式，f_i w_{ij} 都是对应的entry

$$= \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} f_i f_j w_{ij}$$

相等

$$= \frac{1}{2}\left(\sum_{i=1}^{n} d_i f_i^2 - 2\sum_{i,j=1}^{n} f_i f_j w_{ij} + \sum_{j=1}^{n} d_j f_j^2\right)$$

神展开，因为 $d_i = \sum_j w_{ij}$ 替换d_i 后合并多项式

$$= \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2 \qquad (1)$$

$$\geq 0$$

取值只有0，1 大于等于0

MinCut: bipartite graphs with minimal number of cut edges.

- Objective #1: minimize inter-cluster connections
  - Min cut (A, B)
- Objective #2: maximize intra-cluster connections
  - Max vol (A, A) and vol (B, B)
- Overall objective, to minimize $J_{NCut}(A,B) = Cut(A,B)\left(\frac{1}{vol(A)} + \frac{1}{Vol(B)}\right)$
- Solution: the 2nd smallest eigenvector of

$$(D - W)y = \lambda Dy$$

$$L_{NCut} = D^{-1/2}(D - W)D^{-1/2}$$

- Variants
  - RatioCut $J_{RatioCut}(A,B) = Cut(A,B)\left(\frac{1}{|A|} + \frac{1}{|B|}\right)$
  - MinMaxCut $J_{MinMaxCut}(A,B) = Cut(A,B)\left(\frac{1}{Cut(A,A)} + \frac{1}{Cut(B,B)}\right)$

what the disadvantages of bi-partitioning:

always 2^n, inefficient, unstable, unbalanced issue, part of clusters very large, part of cluster very small.

FFN: Why multiple layers

-Automatic feature/representation learning

-Learn complicate (nonlinear) mapping function

Conv net, input: N X N X D, L number of K X K X D kernels, Stride S.

feature map shape:

floor(N-K) / S + 1, floor(N-K) / S + 1, L

Key Difference: CNNs: localized (by convolution)

RNNs: (in principle) capture long-term dependence (by recurrence)

CNN layer order: nonlinear activation --> pooling

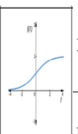
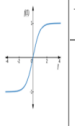
Pool hidden units in the same neighborhood:

Introduces invariance to local translations

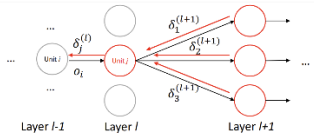
Reduces the number of hidden units in hidden layer

why can't we initialize weight with constant?

This means that the all neurons will output the same value, even after being updated. Therefore initializing every column in your weight matrices with a constant effectively reduces the effective number of neurons in a each layer to 1,

Sigmoid	$\frac{1}{1+e^{-I}}$		ReLU	$\begin{cases} 0, I \leq 0 \\ I, I > 0 \end{cases}$	
Tanh	$\frac{e^I - e^{-I}}{e^I + e^{-I}}$		Leaky ReLU	$\begin{cases} 0.01 \times I, I < 0 \\ I, I \geq 0 \end{cases}$	
			ELU	$\begin{cases} \alpha(e^I - 1), I \leq 0 \\ I, I > 0 \end{cases}$	

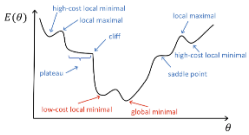
Gradient Computation: Backpropagation



- The gradient of w_{ij} in the *l*th layer (corresponding to unit j in layer l, connected to unit i in layer l-1) is a function of
  - All 'error' terms from layer l+1 $\delta_k^{(l+1)}$ -- An auxiliary term for computation, not to be confused with gradients
  - Output from unit i in layer l-1 (input to unit j in layer l) -- Can be stored at the feed forward phase of computation
- The 'error' terms $\delta_j^{(l)}$ is a function of
  - All $\delta_k^{(l+1)}$ in the layer l+1, if layer l is a hidden layer
  - The overall loss function value, if layer l is the output layer
- We can compute the error at the output, and distribute backwards throughout the network's layers (backpropagation)

Key Challenges

- **Optimization Problem in Deep Learning**
  - Minimize the (approximated) training error *E* $E(\boldsymbol{\theta}) = \frac{1}{m}\sum_{l=1}^{m} \text{Loss}(\hat{T}(X^l, \boldsymbol{\theta}), T^l)$
  - (Stochastic) gradient descent to find the model parameter $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \boldsymbol{g}_t$
- **Challenge 1: Optimization**
  - *E* is non-convex in general
  - How to find a high-quality local optima



- **Challenge 2: Generalization**
  - What we do: minimize (approximated) training error
  - What we really want: minimize the generalization error
  - How to mitigate over-fitting

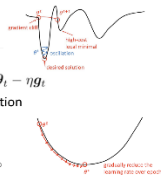
high-cost local minimal/cliff/local maximal/

Plateau: it is where the gradient is almost zero and the gradient descent may become quite slow.

Saddle point: it is where the gradient is zero and the gradient descent will be completely trapped.

- **Saturation of Sigmoid Activation Function**
  - The output $O = \sigma(I) = \frac{1}{1+e^{-I}} \in (0,1)$
  - The derivative $\frac{\partial O}{\partial I} = O(1 - O)$ decaying
  - The error of an output unit $\delta_j = O_j(1 - O_j)(O_j - T_j)$
- **Saturation of Sigmoid Activation Function**
  - If $O_j \approx 1$ or $O_j \approx 0$, both derivative and error will be close to 0
  - Further exacerbated due to backpropagation → gradient vanishing → B.P. is stuck or takes long time to terminate
- **A More Responsive Activation Function: Rectified Linear Unit (RELU)**
  - The output *O=f(I) = I* if *I>0, O=0* otherwise
  - The gradient: $\frac{\partial O}{\partial I} = 1$ if $I > 0$ and $\frac{\partial O}{\partial I} = 0$
  - The error: 0 if the unit is inactive (*I<0*), otherwise, aggregate all error terms from the units in the next higher layer the unit is connected to, w/o decaying → avoid gradient vanishing

Adaptive Learning Rate



- **Stochastic gradient descent to find a local optima**
  - Default choice for η: a fixed, small positive constant $\theta_{t+1} = \theta_t - \eta g_t$
  - Problems: slow progress, or jump over 'gradient cliff', or oscillation
- **Adaptive learning rate: let η change over epoch**
  - Strategy #1: $\eta_t = \frac{1}{t}\eta_0$
  - Strategy #2: $\eta_t = (1 - \frac{t}{T})\eta_0 + \frac{t}{T}\eta_\infty$ if $t \le T$ and $\eta_t = \eta_\infty$
    - Intuitions: smaller adjustment as algorithm progresses
    - $e.g.,\ \eta_0 = 0.9;\ \eta_\infty = 10^{-9}$
  - Strategy #3 (AdaGrad): $\eta_t = \frac{1}{\rho + r_i}\eta_0 \quad r_i = \sqrt{\sum_{k=1}^{t-1} g_{i,k}^2}$
    - Intuition: The magnitude of gradient g_t : indicator of the overall progress
    - $e.g., \rho = 10^{-8}$
  - Strategy #4 (RMSProp): exponential decaying weighted sum of squared historical gradients

No, it will produce the same output for each node regardless of the node ordering.

From the spatial aspect, the GCN output of a node x(i) can be written as:

$$x(i) = w_{ii}x(i) + \sum_{j \in N(i)} w_{ij}x(j).$$

The convolution operation aggregates the information from nerightboring node, which is insensitve to the node ordering.

Why does Dropout Work?

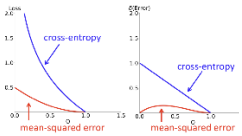
Dropout can be viewed as a regularization technique

Dropout: the model parameters of the current dropout network are updated based on that of the previous dropout network(s)

Cross Entropy

- **Measure the disagreement between the actual (T) and predicted (O) target values**
  - For regression: mean-squared error
  - For (binary) classification: cross-entropy
- **Mean-squared error vs. Cross-entropy**

Loss	Mean-squared error	Cross entropy
Error δ	$\frac{1}{2}(T-O)^2$	$-T\log O - (1-T)\log(1-O)$
	$O(1-O)(T-O)$	$T-O$



Assuming T = 1 (positive tuple)

- **Cross-entropy for Multiclass Problem**
  - Actual target $T = (T_1, T_2, ..., T_C)$
  - Predicted output $O = (O_1, O_2, ..., O_C)$

$$\text{Loss}(T, O) = -\sum_{j=1}^{C} T_j \log O_j$$

30

Turning point happens at 1/3

- Why not deep MLP (or feed-forward neural network)?
  - Deep multilayer perceptrons are **computationally expensive**, and **hard to train**.
    - Long training time, slow convergence, local minima
- Motivations of convolution
  - Sparse interactions
  - Parameter sharing
  - Translational equivalence
- The properties of CNNs are well aligned with properties of many forms of data (e.g. images, text), making them very successful

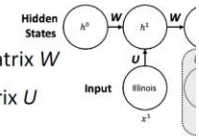
- **RNNs:**

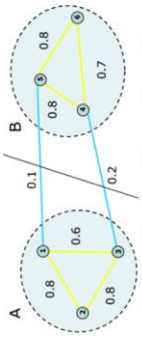
$$h^t = f(Ux^t + Wh^{t-1} + a)$$
$$\hat{y}^T = g(Vh^T + b).$$

$\hat{y}^5(w) = \hat{p}(w|Illi$

- **Feed-forward NNs:** $h_i = f(W_i h_{i-1} + b_i)$

- **Key Differences: Recurrence!**
  - same hidden-to-hidden weight matrix *W*
  - same input-to-hidden weight matrix *U*
  - same bias vector *a*





$$\text{Cut(A, B)} = \sum_{i \in A, j \in B} w_{ij} \Rightarrow \text{Cut(A, B)} = 0.3$$

$$\text{Cut(A, A)} = \sum_{i \in A, j \in A} w_{ij} \Rightarrow \text{Cut (A, A)} = 4.4$$

$$\text{Cut(B, B)} = \sum_{i \in B, j \in B} w_{ij} \Rightarrow \text{Cut (B, B)} = 4.6$$

$$\text{Vol(A)} = \sum_{i \in A} \sum_{j=1}^{n} w_{ij} \Rightarrow \text{Vol(A)} = 4.7$$

$$\text{Vol(B)} = \sum_{i \in B} \sum_{j=1}^{n} w_{ij} \Rightarrow \text{Vol(B)} = 4.9$$

LSTM:

Forget gate $f^t = \sigma(W_f[x^t, h^{t-1}] + a_f)$

Input gate $i^t = \sigma(W_i[x^t, h^{t-1}] + a_i)$

Output gate $o^t = \sigma(W_o[x^t, h^{t-1}] + a_o)$

$$\tilde{C}^t = \sigma(W_c[x^t, h^{t-1}] + a_c)$$

$$C^t = f^t \cdot C^{t-1} + i^t \cdot \tilde{C}^t$$

$$h^t = o^t \cdot \tilde{h}^t$$

$\tilde{h}^t$ is calculated by tanh(w C^t + a)

Att:

$$c^j = \sum_i a(i) h_S^i \quad a(i) = \frac{\exp(\text{score}(h_S^i, h_D^j))}{\sum_{i'} \exp(\text{score}(h_S^{i'}, h_D^j))}$$

$$Z = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}\boxed{X\Theta}$$

$$\tilde{Y} = \text{softmax}(\hat{A}\sigma(\hat{A}X\Theta_1)\Theta_2), \quad \hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$$

Types of Outliers: Global outliers(point anomalies), Contextual Outliers(conditional outliers), collective outliers(group anomaly)

Modeling normal objects and outliers

- Difficulty in modeling normality, ambiguity between normal and abnormal

Application-specific outlier detection

- General-purposed techniques

Noise vs. outliers

Challenge

- Noise: unavoidable, less interesting to the users, but make outlier detection more challenge (e.g., hide the outlier, blur the boundary, mislead detection)

Interpretability

- Why does the algorithm 'think' an object looks suspicious?
- Basic Idea
  - Assume normal data are generated by a stochastic process
  - Data objects in low density regions are flagged as outlier
- Parametric Methods
  - the normal data objects are generated by a parametric distribution with a finite number of parameters
- Non-Parametric Methods **Statistical Approaches**
  - Do not assume a priori statistical model with a finite number of parameters
- Basic Idea
  - Intuition: objects that are far from others can be regarded as outliers
  - Assumption: the proximity of an outlier object to its nearest neighbors significantly deviates from the proximity of most other objects to their nearest neighbors
- Distance-Based Outlier Detection **Proximity-based**
  - Consult the neighborhood of a sample
  - Outlier: if there are not enough objects in its neighborhood

$$\frac{\|\{o' | dist(o, o') \le r\}\|}{\|D\|} \le \pi dist(o, o_k) > r$$

Nystrom, A' is a sample of A.

$$U = \begin{pmatrix} A' \\ B^T \end{pmatrix} U'(\Lambda')^{-1}$$