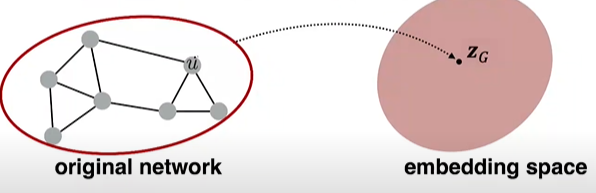
Embedding Entire Graph

# Embedding Entire Graph

**Goal** --- we want to embed a subgraph or an entire graph G, Graph embedding: zG.

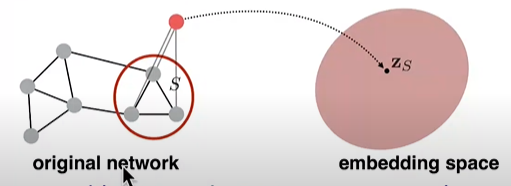
**Tasks**:

* Classifying toxic vs non-toxic molecules
* Identifying anomalous graphs

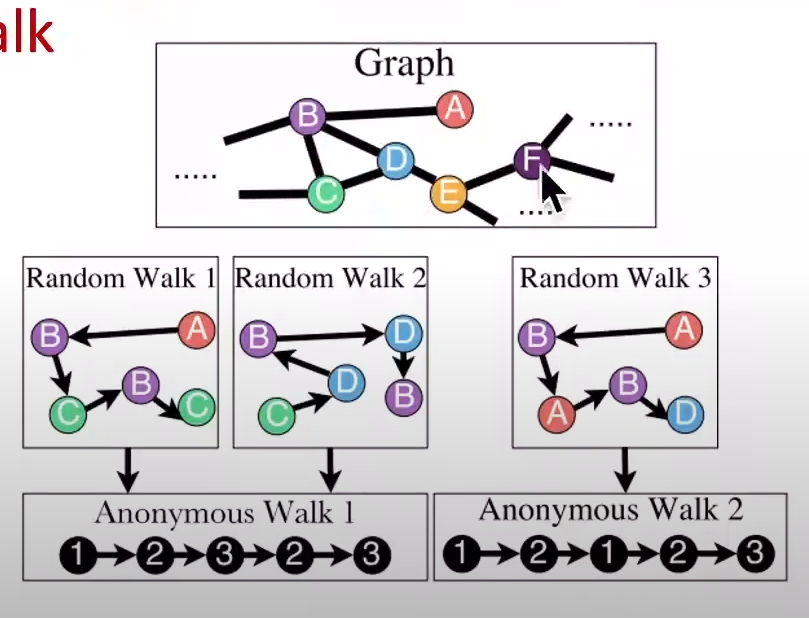
**Approach 1:**

* Run a standard graph embedding technique on the (sub)graph G
* Then just sum (or average) the node embeddings in the (sub)graph G

**Approach 2:**

* Introduce a “virtual node” to represent the (sub)graph and run a standard graph embedding technique

**Approach 3 --- Anonymous Walk Embeddings:**

States in anonymous walks correspond to the index of the first time we visited the node in a random walk.

Agnostic to the identity of the nodes visited (hence anonymous)

Example RW1:

**Step 1**: node A 🡪 node 1

**Step 2**: node B 🡪 node 2 (different from node 1)

**Step 3**: node C 🡪 node 3 (different from node 1, 2)

**Step 4**: node B 🡪 node 2 (same node in step 2)

**Step 5**: node C 🡪 node 3 (same node in step 3)

**Note**: RW2 gives the same anonymous walk

**Number of walks grows:**

Figure Length of Anonymous Walks

For example: There are 5 anonymous walks wi of length 3:

w1 = 111, w2 = 112, w3 = 121, w4 = 122, w5 = 123

# Simple Use of Anonymous Walks

* Simulate anonymous walks wi of l steps and record their counts
* Represent the graph as a probability distribution over these walks

For example:

Set L = 3

Then we can represent the graph as a 5-dim vector:

* Since there are 5 anonymous walks wi of length 3: 111, 112, 121, 122, 123
* ZG[i] = probability of anonymous walk wi in G

# Sampling Anonymous Walks

Sampling anonymous walks --- Generate independently a set of m random walks

Represent the graph as a probability distribution over these walks

How many random walks m do we need?

* We want the distribution to have error of more than with prob less than :

Where is the total number of anonymous walks of length L.

**For example:**

There are = 877 anonymous walks of length L = 7. If we set = 0.1 and = 0.01 then we need to generate m = 122,500 random walks.

# New idea --- Learn Walk Embeddings

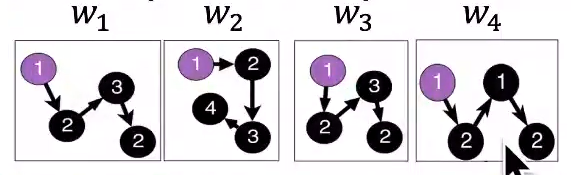
Rather than simply represent each walk by the fraction of times it occurs, we learn embedding zi of anonymous walk wi

* Learn a graph embedding ZG together with all the anonymous walk embeddings zi

Z = {zi: i = 1 …… }， Where is the number of sampled anonymous walks

**How to embed walks?**

**Idea**: Embed walks s.t. the next walk can be predicted.

* A vector parameter zG for input graph
* The embedding of entire graph to be learned
* Starting from node 1: Sample anonymous random walks, e.g.
* Learn to predict walks that co-occur in -size window (e.g. predict w2 given w1, w3 if = 1)

**Objective:**

* Sum the objective over all nodes in the graph
* Run T different random walks from u each of length l:
* Learn to predict walks that co-occur in -size window
* Estimate embedding zi of anonymous walk wi

Let be number of all possible walk embeddings

Objective:

* ) =
* --- All possible walks (require negative sampling)
* means an average of anonymous walk embeddings in window, concatenated with the graph embedding zG
* --- are learnable parameters. This represents a linear layer.
* We obtain the graph embedding zG (learnable parameter) after optimization
* Use zG to make predictions (e.g., graph classification)
* Option 1: Inner product kernel
* Option 2: Use a neural network that take zG as input to classify

# Summary

We discussed 3 ideas to graph embeddings:

* Approach 1: Embed nodes and sum/avg them
* Approach 2: Create super-node that spans the (sub)graph and then embed that node
* Approach 3: Anonymous Walk Embeddings:
* Idea 1: Sample the anonymous walks and represent the graph as fraction of times each anonymous walk occurs
* Idea 2: Embed anonymous walks, concatenate their embeddings to get a graph embedding.

# How to use embeddings

How to use embeddings zi of nodes:

* Clustering/Community Detection --- cluster points zi
* Node Classification --- Predict label of node i based on zi
* Link Prediction --- Predict edge (i, j) based on (zi, zj)

Where we can: cancatenate, avg, product, or take a difference between the embeddings:

* Concatenate:
* Hadamard: (per coordinate product
* Sum/Avg:
* Distance:
* Graph Classification --- graph embedding zG via aggregating node embeddings or anonymous random walks. Predict label based on graph embedding zG