LAB 11 | المخبر 11

Graph Neural Network for State Estimation

شبكة عصبية بيانية لتقدير حالة شبكة كهربائية

A transparent MATLAB implementation of graph convolution on a six-bus system

تنفيذ واضح في MATLAB للالتفاف البياني على شبكة من ستة قضبان

Prepared by | إعداد: Dr.-Ing. Aouss Gabash | د. م. أوس غباش

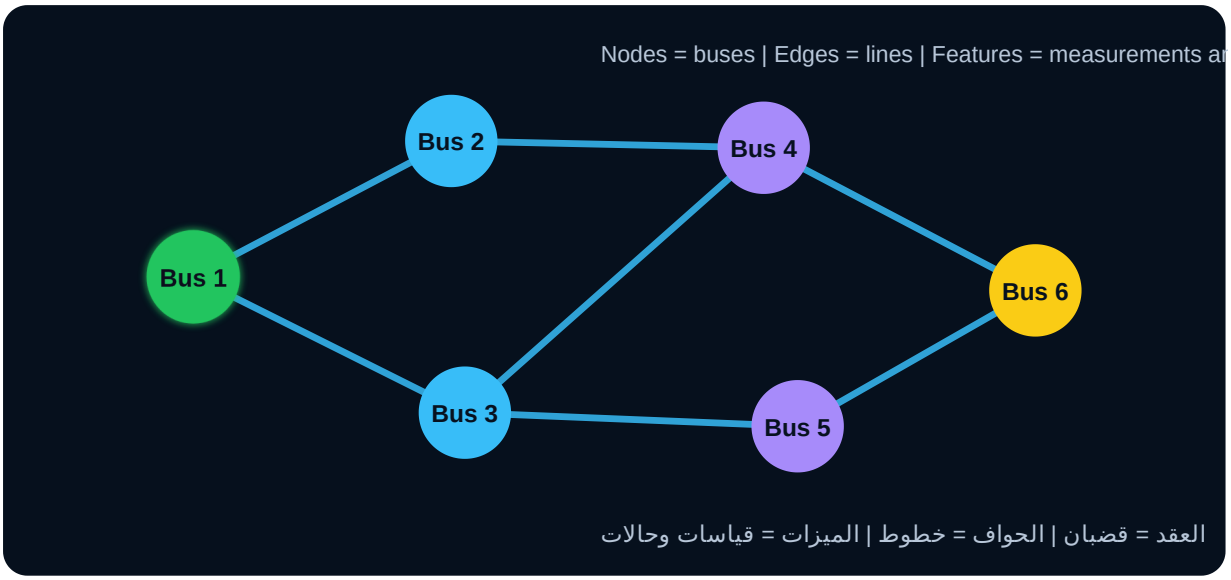
🎯 Laboratory Objectives

- Represent a six-bus power network as a graph.
- Construct adjacency, degree, and normalized adjacency matrices.
- Generate synthetic operating points using a DC power-flow relation.
- Implement a two-layer GCN manually in MATLAB.
- Train the GCN to estimate bus voltage angles.
- Compare it with a topology-blind multilayer perceptron.

🎯 أهداف المخبر

- تمثيل شبكة من ستة قضبان كرسم بياني.
- بناء مصفوفات المجاورة والدرجة والمجاورة المطَبّعة.
- إنشاء حالات تشغيل اصطناعية باستخدام تدفق القدرة المستمر.
- تنفيذ GCN من طبقتين يدويًا في MATLAB.
- تدريبها لتقدير زوايا الجهد.
- مقارنتها بشبكة أمامية لا تستخدم الطوبولوجيا.

1. Six-Bus Network



Bus 1 is the slack/reference bus. The remaining bus angles are estimated from noisy active-power injections.

1. شبكة من ستة قضبان

يُستخدم القضيب 1 كمرجع للزاوية، ويُطلب تقدير زوايا بقية القضبان انطلاقًا من حقن القدرة الفعالة المشوبة بالضجيج.

2. Mathematical Model

$$P = Bbus\theta$$

After removing the slack-bus row and column:

$$\theta red = Bred - 1Pred$$

The supervised learning input is the noisy injection vector, and the target is the true bus-angle vector.

2. النموذج الرياضي

يعتمد المخبر على نموذج DC، حيث ترتبط القدرة الفعالة بزوايا الجهد عبر مصفوفة Bbus. تُستخدم النتائج الصحيحة كتسميات للتدريب.

3. Complete MATLAB Script

```
%% Lab 11: Manual GCN for Six-Bus State Estimation
clear; clc; close all;
rng(11);

%% 1) Graph topology
N = 6;
edges = [
    1 2
    1 3
    2 4
    3 4
    3 5
    4 6
    5 6
];

A = zeros(N);
for k = 1:size(edges,1)
    i = edges(k,1);
    j = edges(k,2);
    A(i,j) = 1;
    A(j,i) = 1;
end

% Add self-loops and normalize
Atilde = A + eye(N);
Dtilde = diag(sum(Atilde,2));
S = Dtilde^(-0.5)*Atilde*Dtilde^(-0.5);

%% 2) DC network susceptance matrix
x = [0.18 0.22 0.20 0.25 0.30 0.17 0.21]';
b = 1./x;

Bbus = zeros(N);
for k = 1:size(edges,1)
    i = edges(k,1);
    j = edges(k,2);
    Bbus(i,i) = Bbus(i,i) + b(k);
    Bbus(j,j) = Bbus(j,j) + b(k);
```

```
    Bbus(i,j) = Bbus(i,j) - b(k);
    Bbus(j,i) = Bbus(j,i) - b(k);
end

slack = 1;
nonSlack = 2:N;
Bred = Bbus(nonSlack,nonSlack);

%% 3) Generate operating samples
numSamples = 5000;
X = zeros(N,1,numSamples);      % noisy injections
Y = zeros(N,1,numSamples);      % true angles

for s = 1:numSamples
    P = zeros(N,1);

    % Random net injections at non-slack buses
    P(nonSlack) = 0.35*randn(N-1,1);

    % Slack balances the total power
    P(slack) = -sum(P(nonSlack));

    theta = zeros(N,1);
    theta(nonSlack) = Bred\P(nonSlack);

    % Measurement noise
    Pmeas = P + 0.02*randn(N,1);

    X(:,:,s) = Pmeas;
    Y(:,:,s) = theta;
end

%% 4) Chronological-style split
nTrain = 3500;
nVal = 750;

iTrain = 1:nTrain;
iVal = nTrain+1:nTrain+nVal;
iTest = nTrain+nVal+1:numSamples;
```

```
XTrain = X(:,:,iTrain);
YTrain = Y(:,:,iTrain);
XVal = X(:,:,iVal);
YVal = Y(:,:,iVal);
XTest = X(:,:,iTest);
YTest = Y(:,:,iTest);

%% 5) Normalize using training statistics
muX = mean(XTrain,'all');
sigX = std(XTrain,0,'all');

XTrainN = (XTrain-muX)/sigX;
XValN = (XVal-muX)/sigX;
XTestN = (XTest-muX)/sigX;

muY = mean(YTrain,'all');
sigY = std(YTrain,0,'all');

YTrainN = (YTrain-muY)/sigY;
YValN = (YVal-muY)/sigY;

%% 6) Initialize two-layer GCN
Fin = 1;
Fhidden = 16;
Fout = 1;

W1 = dlarray(0.15*randn(Fin,Fhidden));
b1 = dlarray(zeros(1,Fhidden));
W2 = dlarray(0.15*randn(Fhidden,Fout));
b2 = dlarray(zeros(1,Fout));

avgW1=[]; avgSqW1=[];
avgb1=[]; avgSqb1=[];
avgW2=[]; avgSqW2=[];
avgb2=[]; avgSqb2=[];

numEpochs = 180;
learnRate = 2e-3;
miniBatchSize = 64;
iteration = 0;
```

```
trainLoss = zeros(numEpochs,1);
valLoss = zeros(numEpochs,1);

%% 7) Training loop
for epoch = 1:numEpochs
    order = randperm(nTrain);
    epochLoss = 0;
    nBatches = 0;

    for startIdx = 1:miniBatchSize:nTrain
        iteration = iteration + 1;
        batchIdx = order(startIdx:min(startIdx+miniBatchSize-1,n

        xb = dlarray(XTrainN(:,:,batchIdx),'SCB');
        yb = dlarray(YTrainN(:,:,batchIdx),'SCB');

        [loss,grads] = dlfeval(@modelGradients, ...
            xb,yb,S,W1,b1,W2,b2);

        [W1,avgW1,avgSqW1] = adamupdate(W1,grads.W1, ...
            avgW1,avgSqW1,iteration,learnRate);
        [b1,avgb1,avgSqb1] = adamupdate(b1,grads.b1, ...
            avgb1,avgSqb1,iteration,learnRate);
        [W2,avgW2,avgSqW2] = adamupdate(W2,grads.W2, ...
            avgW2,avgSqW2,iteration,learnRate);
        [b2,avgb2,avgSqb2] = adamupdate(b2,grads.b2, ...
            avgb2,avgSqb2,iteration,learnRate);

        epochLoss = epochLoss + double(gather(extractdata(loss))
        nBatches = nBatches + 1;
    end

    trainLoss(epoch) = epochLoss/nBatches;

    xv = dlarray(XValN,'SCB');
    yv = dlarray(YValN,'SCB');
    yvPred = gcnForward(xv,S,W1,b1,W2,b2);
    valLoss(epoch) = mean((extractdata(yvPred)-extractdata(yv)).
```

```
    if mod(epoch,20)==0
        fprintf('Epoch %3d | Train %.5f | Val %.5f\n', ...
            epoch,trainLoss(epoch),valLoss(epoch));
    end
end

%% 8) Test prediction
xt = dlarray(XTestN,'SCB');
ytPredN = gcnForward(xt,S,W1,b1,W2,b2);
YPred = extractdata(ytPredN)*sigY + muY;

% Enforce reference angle exactly
YPred(1,:,:) = 0;

%% 9) Performance
err = YPred-YTest;
MAE = mean(abs(err),'all');
RMSE = sqrt(mean(err.^2,'all'));

fprintf('\nGCN Test MAE  = %.6f rad\n',MAE);
fprintf('GCN Test RMSE = %.6f rad\n',RMSE);

%% 10) Plot learning curves
figure;
semilogy(trainLoss,'LineWidth',1.3); hold on;
semilogy(valLoss,'LineWidth',1.3);
grid on;
xlabel('Epoch');
ylabel('MSE');
legend('Training','Validation');
title('GCN Training History');

%% 11) Plot one test sample
sample = 25;
figure;
stem(1:N,YTest(:,:,sample),'filled','LineWidth',1.2); hold on;
stem(1:N,YPred(:,:,sample),'LineWidth',1.2);
grid on;
xlabel('Bus');
ylabel('Voltage Angle (rad)');
```

```
legend('True','GCN Estimate');
title('Six-Bus State Estimation');

%% 12) Error by bus
busRMSE = squeeze(sqrt(mean(err.^2,3)));
figure;
bar(1:N,busRMSE);
grid on;
xlabel('Bus');
ylabel('RMSE (rad)');
title('Estimation Error by Bus');

%% Local functions
function Y = gcnForward(X,S,W1,b1,W2,b2)
    % X: N x 1 x Batch
    B = size(X,3);
    Sdl = dlarray(repmat(S,1,1,B),'SSB');

    % Graph aggregation: S*X
    AX = pagemtimes(Sdl,X);

    % Feature transform
    H = pagemtimes(AX,W1) + reshape(b1,1,[],1);
    H = relu(H);

    % Second graph aggregation
    AH = pagemtimes(Sdl,H);

    % Output layer
    Y = pagemtimes(AH,W2) + reshape(b2,1,[],1);
end

function [loss,grads] = modelGradients(X,T,S,W1,b1,W2,b2)
    Y = gcnForward(X,S,W1,b1,W2,b2);
    loss = mean((Y-T).^2,'all');

    [gW1,gb1,gW2,gb2] = dlgradient(loss,W1,b1,W2,b2);

    grads.W1 = gW1;
    grads.b1 = gb1;
```

```
    grads.W2 = gW2;
    grads.b2 = gb2;
end
```

3. الكود الكامل في MATLAB

ينشئ الكود شبكة من ستة قضبان، ويولد آلاف حالات التشغيل، ثم ينفذ طبقتين من GCN يدويًا باستخدام المصفوفة المطَبّعة S وحلقة تدريب مخصصة.

4. Normalized Adjacency

$$S = D\tilde{} - 1/2(A + I)D\tilde{} - 1/2$$

The multiplication S·X lets each bus receive a weighted average of its own feature and neighboring features.

4. المجاورة المطَبّعة

تسمح المصفوفة S لكل قضيب بدمج قياسه مع قياسات القضبان المجاورة مع منع العقد ذات الدرجة العالية من السيطرة.

5. Two-Layer GCN

$$H = ReLU(SXW_1 + b_1)$$

$$\hat{Y} = SHW_2 + b_2$$

The first layer learns hidden node representations. The second layer maps them to estimated voltage angles.

5. شبكة GCN من طبقتين

تتعلم الطبقة الأولى تمثيلًا داخليًا لكل قضيب، ثم تحول الطبقة الثانية هذا التمثيل إلى زاوية جهد مقدرة.

6. Why the Slack Bus Is Special

Voltage angles are defined relative to a reference. Therefore, the slack-bus angle is fixed to zero.

A learning model may produce a small nonzero slack angle unless the constraint is enforced explicitly or included in the loss.

6. لماذا قضيب المرجع مميز؟

زوايا الجهد نسبية، لذلك تُثبت زاوية قضيب المرجع عند الصفر. ويُعاد فرض هذا الشرط بعد التنبؤ في الكود.

7. Interpretation of the Input and Output

Input at each node

Noisy active-power injection P_i.

Output at each node

Estimated voltage angle θ_i.

The same graph matrix is shared across all operating samples.

7. تفسير الدخل والخرج

يحمل كل قضيب قياس حقن قدرة فعال كدخل، وتنتج الشبكة زاوية الجهد المقدرة لكل قضيب.

8. Expected Results

- Training and validation losses decrease steadily.
- Estimated angles follow the true DC-state solution.
- The slack-bus error is zero after enforcing the reference.
- Buses with weaker measurement information may show larger errors.

8. النتائج المتوقعة

يجب أن ينخفض الخطأ، وأن تتبع الزوايا المقدرة الحل الحقيقي. وقد تختلف الدقة بين القضبان بحسب موقعها واتصالها.

9. Comparison with a Topology-Blind MLP

Flatten each six-bus injection vector and train an ordinary fully connected network:

```
layersMLP = [
    featureInputLayer(6)
    fullyConnectedLayer(32)
    reluLayer
    fullyConnectedLayer(32)
    reluLayer
    fullyConnectedLayer(6)
    regressionLayer
];
```

Use exactly the same training, validation, and test samples. Compare RMSE, parameter count, and robustness to a line outage.

9. المقارنة مع MLP دون طوبولوجيا

درّب شبكة أمامية عادية على المتجه نفسه ثم قارن الدقة وعدد المعاملات والأداء عند خروج خط.

10. Topology-Change Experiment

1. Remove edge (3,5) from A.
2. Recompute S.
3. Regenerate Bbus for the changed network.
4. Test the original GCN without retraining.
5. Fine-tune the model with a small number of new samples.

10. تجربة تغير الطوبولوجيا

1. احذف الخط بين القضيبين 3 و5.
2. أعد حساب S.
3. أعد بناء Bbus.
4. اختبر النموذج القديم دون تدريب.
5. أعد ضبطه بعدد قليل من العينات.

11. Student Tasks

1. Increase the measurement noise from 0.02 to 0.05.
2. Change hidden features from 16 to 8, 32, and 64.
3. Add reactive-power or voltage-magnitude features.
4. Use a weighted adjacency matrix based on 1/x.
5. Add a physics penalty $||P-Bbus\cdot\hat{\theta}||^2$ to the loss.
6. Compare one, two, and three GCN layers.

11. مهام الطالب

1. زد ضجيج القياس.
2. غيّر عدد الميزات المخفية.
3. أضف ميزات كهربائية أخرى.
4. استخدم مجاورة موزونة بـ 1/x.
5. أضف عقوبة فيزيائية لمعادلة القدرة.
6. قارن أعماقًا مختلفة.

12. Mini Project

Physics-informed graph state estimator: Extend the loss with a DC power-balance residual and test whether the model becomes more robust under noisy or missing measurements.

12. مشروع مصغر

أضف إلى دالة الخسارة خطأ اتزان القدرة وفق نموذج DC، ثم اختبر متانة النموذج عند الضجيج أو فقد القياسات.

13. Laboratory Report

- Draw the graph and list all edges.
- Show A, D̃, and S.
- Explain the two GCN equations.
- Report training, validation, and test errors.
- Plot true and estimated angles.

- Compare GCN and MLP.
- Discuss topology changes and physical constraints.

13. تقرير المخبر

يتضمن التقرير الرسم البياني والمصفوفات والمعادلات والنتائج والرسوم والمقارنة مع MLP وتحليل تغير الطوبولوجيا والقيود الفيزيائية.