GCN

November 6, 2020

[1]:
```
using GeometricFlux;
using Flux;
using Flux: onecold, crossentropy, throttle, glorot_uniform, @epochs;
using JLD2;  # use v0.1.2
using SparseArrays;
using Statistics: mean;
using LightGraphs: SimpleGraphs, adjacency_matrix;
```

Načtení dat - dataset Cora

[2]:
```
@load "data/cora_features.jld2" features;
@load "data/cora_labels.jld2" labels;
@load "data/cora_graph.jld2" g;

train_X = Float32.(features);  # dim: num_features * num_nodes
train_y = Float32.(labels);  # dim: target_catg * num_nodes

adj_mat = Matrix{Float32}(adjacency_matrix(g));
```

Nastavení parametrů modelu - Šířka skryté vrstvy - Počet výstupních tříd - Počet trénovacích epoch

[3]:
```
hidden_layer_width = 16;
num_classes = 7;
epochs = 20;
```

Definice modelu pomocí metod balíčku `GeometricFlux.jl` - Jedna vrstva GCN šířky `hidden_layer_width` s aktivační funkcí ReLU - Dropout - Druhá vrstva GCN šířky `num_classes` s lineární aktivací - Softmax funkce

[ ]:
```
model = Chain(
    GCNConv(adj_mat, size(train_X, 1) => hidden_layer_width, relu),
    Dropout(0.5),
    GCNConv(adj_mat, hidden_layer_width => num_classes),
    softmax
);

parameters = Flux.params(model);
```

Alternativně si mohu GCN vrstvy definovat sám

[4]:
```
W1 = Float32.(glorot_uniform(hidden_layer_width, size(train_X, 1)));
b1 = Float32.(glorot_uniform(hidden_layer_width))
function GCN1(X::AbstractMatrix)
    L = normalized_laplacian(adj_mat, eltype(X); selfloop = true);
    return relu.(W1 * X * L .+ b1)
end

W2 = Float32.(glorot_uniform(num_classes, hidden_layer_width));
b2 = Float32.(glorot_uniform(num_classes))
function GCN2(X::AbstractMatrix)
    L = normalized_laplacian(adj_mat, eltype(X); selfloop = true);
    return W2 * X * L .+ b2
end

model = Chain(
    GCN1,
    Dropout(0.5),
    GCN2,
    softmax
);

parameters = Flux.params(W1, b1, W2, b2);
```

Definice ztrátové funkce - cross-entropy. Jako průběžnou míru budeme ukazovat přesnost na trénovacích datech.

[5]:
```
loss(x, y) = crossentropy(model(x), y);
accuracy(x, y) = mean(onecold(model(x)) .== onecold(y));
```

Trénujeme pomocí metody ADAM s `= 0.05`.

[6]:
```
train_data = [(train_X, train_y)];
opt = ADAM(0.05);
evalcb() = @show(accuracy(train_X, train_y));

@epochs epochs Flux.train!(loss, parameters, train_data, opt,␣
 ↪cb=throttle(evalcb, 10));
```

```
 Info: Epoch 1
 @ Main /home/marekdedic/.julia/packages/Flux/05b38/src/optimise/train.jl:114
```

```
accuracy(train_X, train_y) = 0.17540620384047267
```

```
 Info: Epoch 2
 @ Main /home/marekdedic/.julia/packages/Flux/05b38/src/optimise/train.jl:114
```

```
accuracy(train_X, train_y) = 0.2891432791728213
```

```
  Info: Epoch 3
  @ Main /home/marekdedic/.julia/packages/Flux/05b38/src/optimise/train.jl:114
accuracy(train_X, train_y) = 0.3308714918759232
  Info: Epoch 4
  @ Main /home/marekdedic/.julia/packages/Flux/05b38/src/optimise/train.jl:114
accuracy(train_X, train_y) = 0.35782865583456425
  Info: Epoch 5
  @ Main /home/marekdedic/.julia/packages/Flux/05b38/src/optimise/train.jl:114
accuracy(train_X, train_y) = 0.3836779911373708
  Info: Epoch 6
  @ Main /home/marekdedic/.julia/packages/Flux/05b38/src/optimise/train.jl:114
accuracy(train_X, train_y) = 0.4146971935007385
  Info: Epoch 7
  @ Main /home/marekdedic/.julia/packages/Flux/05b38/src/optimise/train.jl:114
accuracy(train_X, train_y) = 0.44054652880354506
  Info: Epoch 8
  @ Main /home/marekdedic/.julia/packages/Flux/05b38/src/optimise/train.jl:114
accuracy(train_X, train_y) = 0.45679468242245197
  Info: Epoch 9
  @ Main /home/marekdedic/.julia/packages/Flux/05b38/src/optimise/train.jl:114
accuracy(train_X, train_y) = 0.4634416543574594
  Info: Epoch 10
  @ Main /home/marekdedic/.julia/packages/Flux/05b38/src/optimise/train.jl:114
accuracy(train_X, train_y) = 0.465288035450517
  Info: Epoch 11
  @ Main /home/marekdedic/.julia/packages/Flux/05b38/src/optimise/train.jl:114
accuracy(train_X, train_y) = 0.46122599704579026
  Info: Epoch 12
  @ Main /home/marekdedic/.julia/packages/Flux/05b38/src/optimise/train.jl:114
accuracy(train_X, train_y) = 0.46491875923190545
  Info: Epoch 13
  @ Main /home/marekdedic/.julia/packages/Flux/05b38/src/optimise/train.jl:114
accuracy(train_X, train_y) = 0.473781388478582
  Info: Epoch 14
  @ Main /home/marekdedic/.julia/packages/Flux/05b38/src/optimise/train.jl:114
accuracy(train_X, train_y) = 0.48522895125553916
```

```
 Info: Epoch 15
 @ Main /home/marekdedic/.julia/packages/Flux/05b38/src/optimise/train.jl:114
accuracy(train_X, train_y) = 0.4951994091580502
 Info: Epoch 16
 @ Main /home/marekdedic/.julia/packages/Flux/05b38/src/optimise/train.jl:114
accuracy(train_X, train_y) = 0.5062776957163959
 Info: Epoch 17
 @ Main /home/marekdedic/.julia/packages/Flux/05b38/src/optimise/train.jl:114
accuracy(train_X, train_y) = 0.5188330871491876
 Info: Epoch 18
 @ Main /home/marekdedic/.julia/packages/Flux/05b38/src/optimise/train.jl:114
accuracy(train_X, train_y) = 0.5214180206794683
 Info: Epoch 19
 @ Main /home/marekdedic/.julia/packages/Flux/05b38/src/optimise/train.jl:114
accuracy(train_X, train_y) = 0.5273264401772526
 Info: Epoch 20
 @ Main /home/marekdedic/.julia/packages/Flux/05b38/src/optimise/train.jl:114
accuracy(train_X, train_y) = 0.53397341211226
```

Kód je modifikací příkladů balíčku GeometricFlux.jl.