Optymalną architekturę modelu wybraliśmy porównując wartości BCELoss (Binary Cross-Entropy Loss) dla zbioru walidacyjnego. Najniższa wartość (uzyskana w dowolnym epochu) wskazywała na najlepszy model. Zmian dokonywaliśmy w liczbie neuronów w warstwie ukrytej (hidden\_dim w klasie GCNMOdel (na stronie 2)), liczbie neuronów w warstwie ukrytej perceptronu (dla modeli nieliniowych, które osiągnęły gorsze wyniki niż liniowe) oraz liczbie osadzeń (wymiarowość embeddingu). Dla warstwy ukrytej sprawdziliśmy 16, 32 oraz 64 neurony i ostatnia, największa wartość dawała najlepsze wyniki, toteż pozostawiliśmy 64 neurony. Dla warstwy ukrytej perceptronu również sprawdziliśmy 16, 32 i 64 neurony i tym razem środkowa wartość zapewniła najlepsze rezultaty. Z kolei sprawdzane przez nas liczby osadzeń (oprócz 1 i 2) wynosiły 64, 128 i 256. Tu również najlepsza była wartość środkowa – 128.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznieKlasyfikator trenowany był wg algorytmu:

Zapobiegliśmy przeuczeniu poprzez zachowanie jedynie modelu o najniższym BCELoss dla zbioru walidacyjnego i zatrzymując trenowanie, jeśli 10 razy z rzędu po uczeniu model nie będzie lepszy od najlepszego modelu.

Obraz zawierający tekst, Wykres, linia, diagram

Opis wygenerowany automatycznieObraz zawierający tekst, zrzut ekranu, Wykres, linia

Opis wygenerowany automatycznieRozłożenie danych i granica decyzyjna przy kodowaniu w jednym wymiarze dla najlepszej uzyskanej dokładności na zbiorze walidacyjnym (lewo) oraz wykres wartości MSELoss (prawo):

Jak można łatwo zauważyć wartości znacznie nachodzą na siebie; nie jest to dobra metoda kodowania, ponieważ trudno jest oddzielić od siebie klasy.Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznieObraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Rozłożenie danych i granica decyzyjna przy kodowaniu w 2 wymiarach dla najlepszej uzyskanej dokładności na zbiorze walidacyjnym (lewo) oraz wykres wartości MSELoss (prawo):

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznieObraz zawierający tekst, zrzut ekranu, Wykres, diagram

Opis wygenerowany automatycznieWyraźnie widać, że podział jest znacznie lepszy od 1-wymiarowego, lecz punkty nadal nie są wyraźnie rozdzielone.

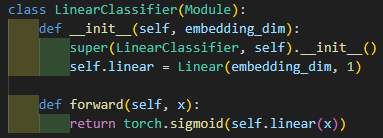
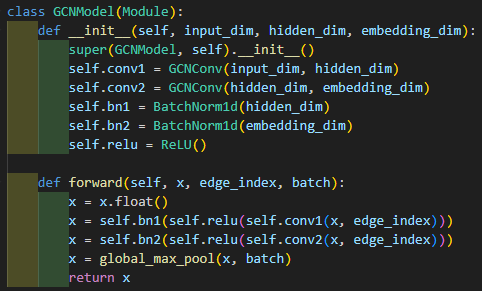
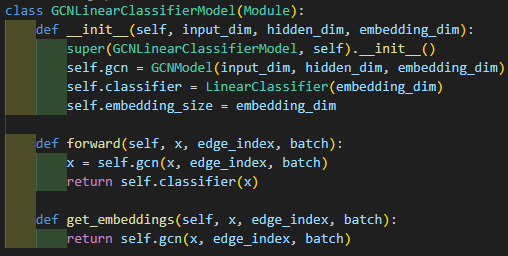
Obraz zawierający tekst, Wykres, diagram, linia

Opis wygenerowany automatycznie

Obraz zawierający Wykres, tekst, linia, diagram

Opis wygenerowany automatycznieObraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznieNajlepszy wynik dokładności uzyskaliśmy jednak dla embeddingu 128-wymiarowego dla modelu GCNLinearClassifierModel:



Obraz zawierający tekst, zrzut ekranu, diagram, linia

Opis wygenerowany automatycznie Dla modelu linowego rozkład w 2 wymiarach wyglądał następująco:

Optymalną architekturę modelu wybraliśmy porównując wartości MSELoss (Mean Squared Error Loss) dla zbioru walidacyjnego. Najniższa wartość (uzyskana w dowolnym epochu) wskazywała na najlepszy model. Wybór hiperparametrów był analogiczny do zadania klasyfikacji. Wyniki również były zbliżone, zatem zdecydowaliśmy się na 128 wymiarów osadzeń i 64 neurony w warstwie ukrytej i 32 neurony w warstwie ukrytej perceptronu.

Model regresji trenowany był zgodnie z poniższym algorytmem:  
Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Zapobiegliśmy przeuczeniu poprzez zachowanie jedynie modelu o najniższym MSELoss dla zbioru walidacyjnego i zatrzymując trenowanie, jeśli 10 razy z rzędu po uczeniu model nie będzie lepszy od najlepszego modelu.

Obraz zawierający tekst, Wykres, zrzut ekranu, linia

Opis wygenerowany automatycznieNajniższą wartość MSELoss w przypadku kodowania w 1 wymiarze uzyskaliśmy dla modelu liniowego (TransformerLinearRegressionModel): Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

Obraz zawierający tekst, Wykres, linia, zrzut ekranu

Opis wygenerowany automatycznieJak wyraźnie widać uzyskaliśmy słaby wynik.  
Jeszcze gorsze wyniki uzyskaliśmy dla modelu nieliniowego (Transformer NonlinearRegressionModel) przedstawionego po prawej:

Obraz zawierający tekst, zrzut ekranu, diagram, Wykres

Opis wygenerowany automatycznieDla kodowania w 2 wymiarach osiągnięto najlepsze wyniki przy użyciu TransformerNonlinearRegressionModel:

Obraz zawierający tekst, Wykres, linia, diagram

Opis wygenerowany automatycznieObraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

Jednak najlepsze wyniki osiągnięto dla kodowania w 128 wymiarach z użyciem TransformerLinearRegressionModel: Obraz zawierający tekst, zrzut ekranu, Czcionka

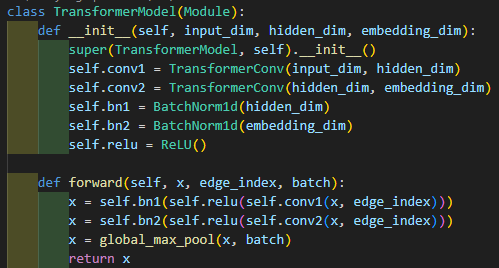
Opis wygenerowany automatycznie

Obraz zawierający tekst, Wykres, linia, diagram

Opis wygenerowany automatycznieObraz zawierający tekst, zrzut ekranu, diagram, linia

Opis wygenerowany automatycznieDla modelu linowego rozkład w 2 wymiarach wyglądał następująco:

Obraz zawierający tekst, zrzut ekranu, Czcionka, oprogramowanie

Opis wygenerowany automatycznie 

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie