Preferred Networks インターン選考2019 コーディング課題 機械学習・数理分野

変更履歴

- 2019年4月19日 : 初版

回答にあたっての注意

- ソースコードには以下のいずれかの言語を利用してください。
  - C, C++, Python, Ruby, Go, Java, Rust
- コーディング能力を見るという目的のため、各言語の標準ライブラリのみを使用して下さい。必要であれば、以下に指定するライブラリを使用してもよいです。以下に指定していないライブラリの使用は禁止します。
  - 行列計算ライブラリ (Python: numpy、C++: Eigen など) (自動微分ライブラリや、深層学習フレームワークなどの使用は禁止します)
  - 結果を可視化するためのライブラリ (Python: matplotlib など)
- 課題には自分だけで取り組んでください。この課題を他の応募者を含めた他人と共有・相談することを禁止します。課題期間中に GitHub の公開リポジトリ等にソースコードや問題をアップロードする行為も禁止します。漏洩の証拠が見つかった場合、その応募者は失格となります。ある応募者が別の応募者に回答をコピーさせた場合、双方の応募者が失格となります。
- 想定所要時間は最大2日です。全課題が解けていなくても提出は可能ですので、学業に支障の無い範囲で取り組んで下さい。

提出物

以下のものを提出して下さい。ディレクトリ名やファイル名は以下に従って下さい。

- 課題1-4のソースコード
  - `src` というディレクトリを作ってそこにソースコードを置いて下さい。
  - ソースコードは課題ごとに別々になっていても、課題1-4を通じて1つのファイルにまとまっていても構いません。
- ソースコードのビルド方法・実行手順について記したテキスト
  - `README.txt`、`README.md` などのファイル名にしてください。
  - 補足資料として、ソースコードの説明を記述して頂いても構いません。
- 課題4でテストデータに対して予測したラベル
  - `prediction.txt` というファイル名にして、提出物のディレクトリ直下に置いてください。
- 課題3,4のレポート
  - レポートは課題3,4を通して A4 用紙で1枚もしくは2枚以内程度でまとめてください。図や表も含めます。
  - `report.pdf`、`report.txt`、`report.md`、`report.doc` などのファイル名にしてください。

評価基準

提出物の評価にあたって以下のような要素を考慮します。

- ソースコードが他人が読んで理解しやすいものになっていること。
- ソースコードが正しいものであることを検証するためにある程度の単体テストや検証のためのコードが書かれていること。
- 提出物を見て追試がしやすい形になっていること。
- レポートが要点についてわかりやすくまとまっていること。

一次選考後の面接で提出されたコードについて質問する可能性があります。

提出方法

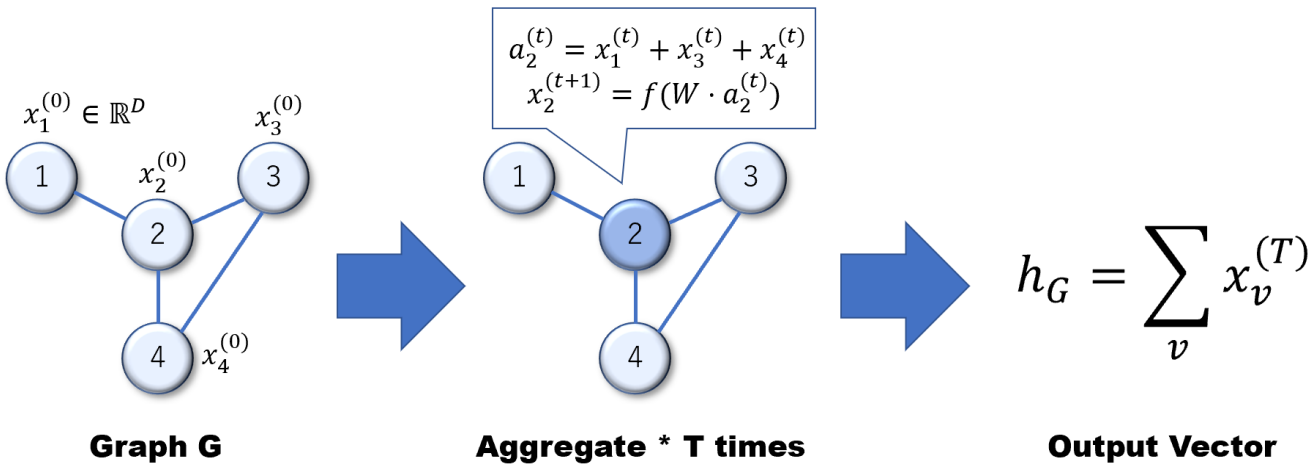
上記の提出物を単一のパスワード無しzipファイルにまとめ、こちらの専用フォームより応募してください。締切は日本時間2019年5月7日（火）12:00です。

問い合わせ

問題文に関して質問がある場合はintern2019-admin@preferred.jpまでご連絡ください。問題文に訂正が行われた場合は応募者全員にアナウンスいたします。なお、アプローチや解法に関する問い合わせにはお答えできません。

問題文

Graph neural network (GNN) はグラフデータを扱うニューラルネットのモデルです。この課題では Graph neural network をフルスクラッチで実装し、用意したデータセットに対して分類器を学習することを目的にします。



グラフ G が与えられるものとします。グラフ G の頂点集合と枝集合をそれぞれ V, E とします。この課題では全体を通して10〜15頂点程度の無向グラフを扱うものとします。GNN ではまずそれぞれの頂点 $v \in V$ に特徴ベクトル $x_v^{(0)} \in \mathbb{R}^D$ を割り当てます。ここで、D は特徴ベクトルの次元であり、固定された整数値です。GNN は以下の集約と呼ばれる操作を繰り返すことで、t ステップ目での特徴ベクトルの集合 $\{x_v^{(t)}\}_{v \in V}$ を $\{x_v^{(t+1)}\}_{v \in V}$ に更新します。

- (集約-1): $a_v^{(t)} := \sum_{w:(w,v) \in E} x_w^{(t)}$.
- (集約-2): $x_v^{(t+1)} := f(W \cdot a_v^{(t)})$.

ここで、W は GNN がパラメータとして持つ $D \times D$ 行列、f は非線形関数です。W は学習によって最適化する対象となります。この課題では f は要素ごとに ReLU を適用する関数とします。なお、パラメータ W はステップ t に依存せず、集約の各ステップで共通のパラメータを用いることとします。

集約を何ステップか行った後、特徴ベクトルを足し合わせることで、グラフ G に対する特徴ベクトル h_G を得ます。

- (READOUT) $h_G := \sum_{v \in V} x_v^{(T)}$.

この特徴ベクトルを用いることで、グラフデータに対する分類問題を解くことができます。

なお、実際の研究や応用で使われる GNN にはさまざまなバリエーションがあり、ここで説明したものよりも複雑なモデルや問題設定が考えられることが多々あります。

課題1

パラメータ W が固定されていると仮定した上で、上記に述べた GNN を実装してください。具体的には、

- グラフ G が与えられたときに (集約-1), (集約-2), (READOUT) を計算し h_G を求める関数・もしくはクラスを記述してください。グラフデータをコード上でどのように保持するかは任意とします。保持の仕方としては、例えば隣接行列形式などがあります。
- 初期の特徴ベクトル $x_v^{(0)}$ をどうするかは任意とします。例えば、特徴ベクトルの最初の要素だけ1でそれ以外が0のベクトルにするなどがありえます。

計算結果が正しいことを確かめるためのテストを記してください。テストに用いるハイパーパラメータ D や T の値は任意とします。

課題2

GNN を使ってグラフの2値分類問題を解くことを考えます。上記の h_G を求めた後、h_G の重み和を取ります。それを Sigmoid 関数に掛けることで $(0, 1)$ の確率値 p を得ます。分類器の予測ラベル $\hat{y}$ は、$p > 1/2$ のとき 1、そうでないとき 0 を取るものとします。

- $s := A \cdot h_G + b$.
- $p := \mathrm{Sigmoid}(s) := 1/(1 + \exp(-s))$.
- $\hat{y} = \mathbf{1}[p > 1/2]$.

ここで、$A \in \mathbb{R}^D$ と $b \in \mathbb{R}$ は分類器の持つパラメータとします。以降、$\theta = \{W, A, b\}$ を分類器のパラメータ集合とします。

グラフ G に対する正解ラベルを $y \in \{0, 1\}$ とするとき、この予測に対する binary cross-entropy 損失関数(loss)は以下で与えられます。

- $L := L(G, y; \theta) := -y \log p - (1 - y) \log(1 - p)$.

(注: binary cross-entropy 損失関数を定義通りに計算すると、計算途中でオーバーフローする可能性があります。関数を適切に実装することでオーバーフローを回避するようにしてください。例えば、以下のように式変形し、例えば s が十分大きいときには $\log(1 + \exp(s)) \simeq s$ と近似することでオーバーフローを回避できます。)

- $L = y \log(1 + \exp(-s)) + (1 - y) \log(1 + \exp(s))$.

GNN の学習では、損失 L が小さくなるようにパラメータ集合 θ を最適化します。最適化のために勾配法を用います。すなわち、損失 L に対するパラメータ集合の勾配 $\partial L/\partial \theta = \{\partial L/\partial W, \partial L/\partial A, \partial L/\partial b\}$ を計算し、この方向に沿ってパラメータを更新する、という処理を反復的に行います。この課題では実装を簡単にするため、数値微分によって勾配を求めることにします。

まず、課題1で実装した GNN を拡張し、グラフ G とラベル y に対してその損失 L を計算する処理を記述してください。

続いて、損失 L の θ に関する勾配を数値微分で計算するコードを書いてください。これは以下で計算するものとします。

- パラメータ θ の i 番目の要素を θ_i と記すことにします。このとき、勾配 $\partial L/\partial \theta_i$ を $(L(G, y; \theta + \epsilon \delta_i) - L(G, y; \theta))/\epsilon$ によって近似します。 ここで、δ_i は i 番目の要素だけ 1 でそれ以外が 0 のベクトル、$\epsilon > 0$ は微小な実数値とします。

これによって勾配 $\partial L/\partial \theta$ を計算した後、以下の計算をしてパラメータ θ を更新してください。

- $\theta := \theta - \alpha \partial L/\partial \theta$.

10頂点程度の適当な固定されたグラフ G と適当なラベル $y \in \{0, 1\}$ に対してこの更新を反復すると、損失が下がっていくことを確認してください。

ここで、$\alpha > 0$ は学習率(learning rate)と呼ばれるハイパーパラメータです。

実装に用いるハイパーパラメータは任意とします。ハイパーパラメータの一例については後述する「ハイパーパラメータのヒント」の項を参考にしてもよいです。

うまく実装した場合、十分な回数の反復を行えば多くの場合で損失 $L(G, y; \theta)$を 0.01 以下にできることを確認しています。ただし、モデルパラメータの初期値や入力 (G, y) によっては勾配が消滅して損失が減らないことも稀にあります。

課題3

課題2で作った分類モデルを、実データセットを使って学習させましょう。データセットはグラフとラベルのペア (G, y) の集合から構成されます。

学習用のデータセットは `datasets/train` ディレクトリにあります。ディレクトリには `{id}_graph.txt` というファイルと `{id}_label.txt` という2種類のファイルがあり、それぞれ1つのグラフとそれに対応するラベルが記されています。グラフは以下のような隣接行列形式でエンコードされています。なお、ここで与えられる隣接行列は対称行列になっています。

```
n  # グラフの頂点数
a_11 a_12 ... a_1n  # もし頂点 i と j の間に枝が存在するならば a_{ij}=1,
a_21 a_22 ... a_2n  # そうでないとき a_{ij}=0 である
...
a_n1 a_n2 ... a_nn
```

学習のために Stochastic Gradient Descent (SGD) を組んでください。これは次のような処理を繰り返すアルゴリズムです。

- (サンプリング) データセットから B 個のデータをランダムにサンプリングします。$B \geq 1$ はハイパーパラメータです。 サンプリングした B 個のデータをミニバッチと呼びます。
- (勾配計算) ミニバッチ内のデータに対して勾配を計算し、その平均を取ってください。すなわち、ミニバッチのデータを $(G^1, y^1), \ldots, (G^B, y^B)$ として、$\Delta\theta := \frac{1}{B} \sum_{j=1}^{B} \partial L(G^j, y^j)/\partial \theta$ を計算します。
- (更新-SGD) $\theta := \theta - \alpha \Delta\theta$ として更新します。ここで、$\alpha > 0$ は学習率です。

ミニバッチのランダムサンプリングは、1度サンプリングしたものは2度引かないように行います。全部のデータをサンプリングするまでの周期をエポックと呼びます。1つのエポックが終わったらサンプリングを最初からまた行います。

またさらに、SGD の拡張である Momentum SGD を実装してください。これは元の SGD の更新部分を、以下のように変えたものです.

- (更新-Momentum SGD 1) 前のステップでの更新量を w とします。最初の実行時には w はゼロベクトルです。
- (更新-Momentum SGD 2) $\theta := \theta - \alpha\Delta\theta + \eta w$ と更新します。ここで $\eta > 0$ はモーメントと呼ばれるハイパーパラメータです。この更新の後、w は $w := -\alpha\Delta\theta + \eta w$ と更新します。

SGD と Momentum SGD をそれぞれ実装して下さい。次に、`data/train` にあるデータを学習用のデータと検定用のデータの2種類に分割してください。学習用のデータを用いて分類器のモデルを数エポック学習させ、そのとき学習用のデータと検定用のデータのそれぞれに対してデータ内での平均損失や平均精度がどのようになるかを観測し、レポートに記してください。なお、検定用のデータは分類器の学習には用いないことに注意してください。

ハイパーパラメータのヒント

以下にハイパーパラメータの例を示します。課題3でデータで学習させるにあたってこれらのハイパーパラメータを使用すると検定用のデータに対する平均予測精度を 60% 程度にできることを確認しています。なお、これらのハイパーパラメータは一例であり、実装ではこれ以外のハイパーパラメータを使っても良いです。

- 集約を行うステップ数 $T = 2$.
- 特徴ベクトルの次元数 $D = 8$.
- 学習率 $\alpha = 0.0001$, モーメント $\eta = 0.9$.
- パラメータ W と A の初期値は平均 0 標準偏差 0.4 の正規分布からサンプリングして初期化する。パラメータ b は 0 に初期化する。
- 数値微分の摂動 $\epsilon = 0.001$.
- 学習の際のエポック数: 10〜100程度 (注: 初期値によって収束するかどうかが結構変わります)

課題4

課題3の実装に対して以下の中からいずれかの工夫を加え、実装してください。

- パラメータの更新アルゴリズムとして Adam [1] を実装してください。
- これまでに実装した GNN のモデルに何らかの変更を加えてください。例えば、活性化関数の ReLU を何か他の関数に置き換えたり、集約ステップで用いる変換を線形関数から2層の Multi-Layer Perceptron に置き換える(参考: [2])とどうなるか、などです。
- データに対する前処理として、元のグラフに対して何らかの加工をしたときにモデルの性能がどのように変化するか観測してください。この際、新しい Edge Type を導入して、元のモデルでの集約のステップを改変してもよいです。一例としては、元のグラフに新しい頂点を1つ足し(super-node と呼ぶことにします)、super-node とそれ以外の頂点を新しい Edge Type の枝で結ぶ、というのがあります。

`datasets/test` の中にテスト用のデータがあります。これは `datasets/train` と同じフォーマットで、ラベルデータのみが存在しないものとなっています。学習したモデルを用いて `datasets/test` の中のグラフデータに対して分類を行い、1つのテキストファイルにして提出してください。テキストファイルは以下のように1行に1つの予測ラベルが記されている形式にしてください。

```
<0_graph.txt に対する予測ラベル>
<1_graph.txt に対する予測ラベル>
<2_graph.txt に対する予測ラベル>
...
```

参考文献

[1] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." ICLR 2014.

[2] Xu, Keyulu, et al. "How Powerful are Graph Neural Networks?." ICLR 2019.