GithubのIssuesとPull Requests

バグを見つけたら？新機能を提案するには？コードを修正して貢献するには？まずこのページを読んでください.

バグ報告

作成したコードが動作せず，Kerasに問題があると思ったら？バグ報告のために以下の手順を踏んでください.

1. 既にバグは修正されているかもしれません．Kerasの現在のmaster branchと最新のTheano/TensorFlow/CNTKのmaster branchに更新してください． Theanoの簡単な更新方法：`pip install git+git://github.com/Theano/Theano.git --upgrade`
2. 類似したissueを検索してください．解決済みのissueも検索するために，`is:open`を消して検索してください．他の方が既にバグを見つけているかもしれません．また，KerasのFAQを確認してください．まだ，問題が解決されなかったら？Githubのissueで報告してください.
3. あなたの設定に関する有用な情報を教えてください：使っているOSはなんですか？Kerasのバックエンドに何を使っていますか？GPU上で実行していますか？もしそうならCudaとCuDNNのバージョンはなんですか？GPUは何ですか？
4. Issueの内容を再現するスクリプトを提供してください．スクリプトは，すぐに実行可能であり，外部データのダウンロードは不要にすべきです（何かしらのテストデータに対してモデルの実行が必要な場合，ランダムに生成したデータを使ってください）．コードを投稿するためにGithub Gistsの利用を推奨します．再現できないissueはclosedになりやすいです.
5. もし可能なら，あなた自身でバグを修正してください--できれば！

より多くの情報を提供することで，バグの検証がより簡単になり，より早く行動を起こすことができます．もしissueをすぐに解決したい場合，上記の手順を踏むことは重要です.

機能リクエスト

使いたいKerasの機能やKeras APIの変更に関するリクエストのためにGithub issuesを利用できます.

1. 欲しい機能とその追加が重要である理由をわかりやすく詳細に説明してください．Kerasの少数のユーザのためではなく，大多数のために有益な機能を望んでいることを気に留めてください．もし少数のユーザを対象とするなら，Kerasのアドオンライブラリとして作成することを考慮してください．KerasにとってAPIやコードベースの膨大化を回避することは重要です.

2. あなたが考えているAPIのデモと追加したい機能のユースケースを示すコードスニペットを提供してください．もちろん，実際のコードを書く必要はありません！
3. 機能について議論してから，Pull Requestを試みることができます．もし可能なら，コードを書いてください．常に私たちはその機能を追加するよりも多くの作業を持っています．もしあなたがコードを書けるなら，プロセスを速めるでしょう．

貢献するためのリクエスト

ボードに未解決のissueや追加すべき機能に関するリストがあります．もし，Kerasへの貢献を始めたいなら，ここから始められます．

Pull Requests

pull requestをどこに送るべきですか？

1. **Kerasの改善とバグ修正**はKeras `master` branchに送ってください．
2. **試験的な新機能**としてのレイヤーやデータセットはkeras-contribに送ってください．Kerasのコアに属すべきRequests for Contributionsにある新機能以外です．もしコア機能だと思う場合は，追加する機能の説明するための設計書を送ることで，主張できます（以下の説明を見てください）．

バグ修正やドキュメントの向上，新しい機能の追加とは対照的に，**コーディングタイル**を主としたPRsはほぼ拒否されることに注意してください．

ここでは，あなたの改善したコードを送るためのクイックガイドを示します:

1. もしPRによって機能的な変更が生じる場合，変更すべきかどうか，どのようにそれを対処するか議論するために設計書を書いてKerasのメーリングリストに投稿してください．これによって，あなたのPRが閉じられることを防ぐでしょう！もちろん，PRが単純なバグ修正なら，必要ありません．設計書の作成と投稿手順は以下の通りです：
    - Google Doc templateを開き，これを新しいGoogle docにコピーします．
    - 内容を記入します．サンプルコードを含める必要があることに気をつけてください．コードを挿入するために，CodePrettyのようなGoogle docの拡張機能を使ってください（いくつかこのような拡張機能が利用可能です）
    - 共有設定を"everyone with the link is allowed to comment"にしてください．
    - 私たちが気づくように（全部大文字の）`[API DESIGN REVIEW]`からはじまるタイトルをつけたこの文書を`keras-users@googlegroups.com`に投稿してください．
    - コメントを待ち，コメントがきたら答えてください．必要に応じて文書を編集してください．
    - 提案書は最終的に承認か拒否をされます．承認されたら，あなたがPull Requestを送るかPull Requestを書くように依頼してください．
2. コードを書きましょう．ここが辛いパートです！
3. 追加した新しい関数やクラスに適切なdocstringを書いてください．あなたの関与したコードが最新のdocstringとドキュメントであることを確認してください．**Docstringのスタイルは重視すべきです．** 特にフォーマットはMarkDownで，（可能な

ら）`Arguments`，`Returns`，`Raises`のセクションがあるべきです．参考例としてコードベースにおける他のdocstringを見てください．
4. テストを書いてください．あなたのコードは完全なユニットテストのカバレッジが必要です．PRをすぐにマージして欲しい場合は重要です．
5. ローカルでテストスイートを実行してください．これは簡単です：Kerasのフォルダから`py.test tests/`を実行します．
   - テストに関するライブラリをインストールする必要があります：`pip install -e .[tests]`．
6. 全テストが通ることを確かめてください．
   - Python 2.7とPython 3.6のTheanoバックエンド．Theanoの開発バージョンであることを確かめてください．
   - Python 2.7とPython 3.6のTensorFlowバックエンド．TensorFlowの開発バージョンであることを確かめてください．
   - Python 2.7とPython 3.6のCNTKバックエンド．CNTKの開発バージョンであることを確かめてください．
7. PEP8の構文規則に従っていますが，1行の長さに関しては教義的でありません． ただし，合理的だと考える長さを維持してください．楽をするために，PEP8 linterの実行を推奨します：
   - PEP8パッケージのインストール：`pip install pep8 pytest-pep8 autopep8`
   - スタンドアローンなPEP8のチェック：`py.test --pep8 -m pep8`
   - いくつかのPEP8に関するエラーは，自動修正が可能です：`autopep8 -i --select <errors> <FILENAME>`
     例：`autopep8 -i --select E128 tests/keras/backend/test_backends.py`
8. コミット時は，適切で記述的なcommitメッセージを使ってください．
9. ドキュメントを更新してください．新機能追加の場合，新機能の使用方法に関するコードスニペットを含めてください．
10. PRを送ってください．もしあなたの変更が以前の議論で承認されており，完全で（通過する）ユニットテストと適切なdocstring/ドキュメントが含まれていれば，PRはすぐにマージされるでしょう．そうでない場合は．．．

新しいサンプルコードの追加

たとえKerasのソースコードへ貢献しなくても，Kerasを使った簡潔で強力な応用例を持っていましたら，examplesのコレクションへの追加を考えてみてください．既にあるexamplesは慣用的なKerasコードです：あなたのスクリプトも同じように作成してください．