GPUでの学習

これまでローカルPC上でパラメータサーチを行っていたが, これをCPU, GPUを用いた並列計算に変更.

GBDT・・・iterationの増加, 最大決定木数を800→6000へ,
決定木の深さを4まで探索
CNN・・・ 1000iteration→40000iterationまで増加, 入力の
正規化撤廃,
ロジスティック回帰, SVM・・・iterationの増加

データ

- ajaxカテゴリ
- 総bookmark数100以上のWebページのみ
- 入力は5日刻みで取得し, 30日間の平均bookmark数が2以下のものは除く

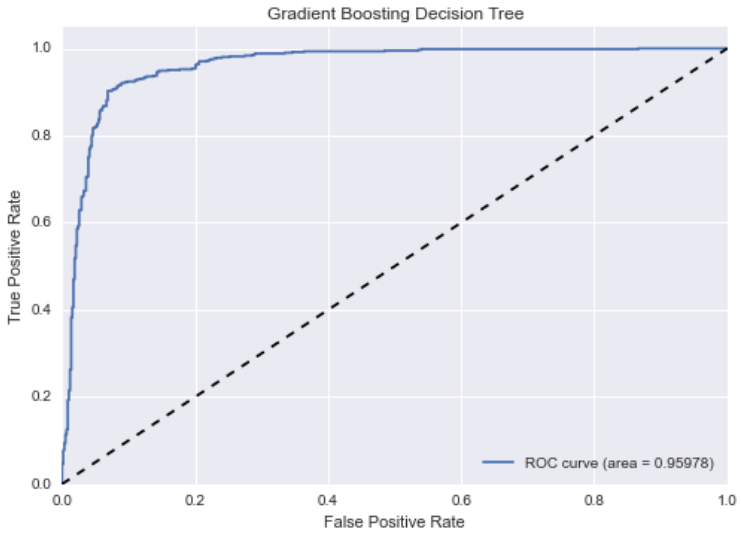
Convolutional Neural Network

モデル

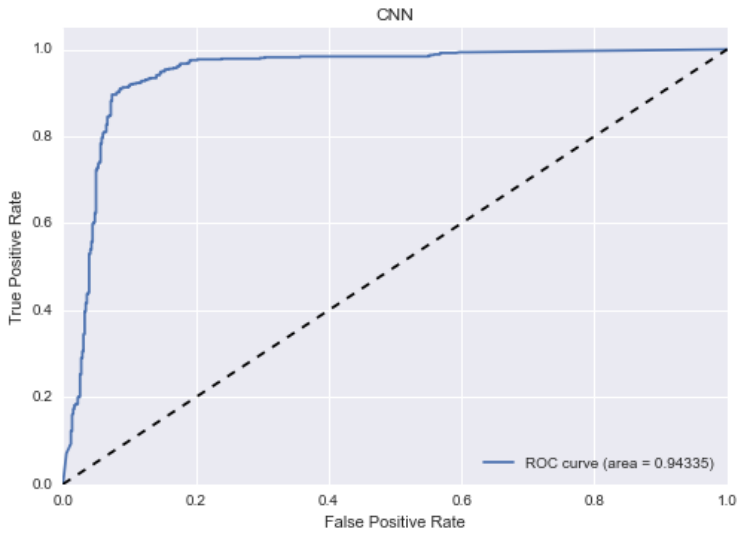
入力(30次元)
→1×5 convolutionで20枚にマップ→1×2 max pooling
→1×5 convolutionで50枚にマップ→1×2 max pooling
→full-connectで500次元にマップ(250→500)
→1次元の出力

- fully-connect layerはDropoutで正則化
- ADAMで最適化(SGDと比較して精度向上あり)
- バッチサイズ50で40000iteration学習
- クロスエントロピーロス
- 切断正規分布から重み初期値をサンプリング
- バイアスは初期値0を回避(微小値を加える)
- 0 padding

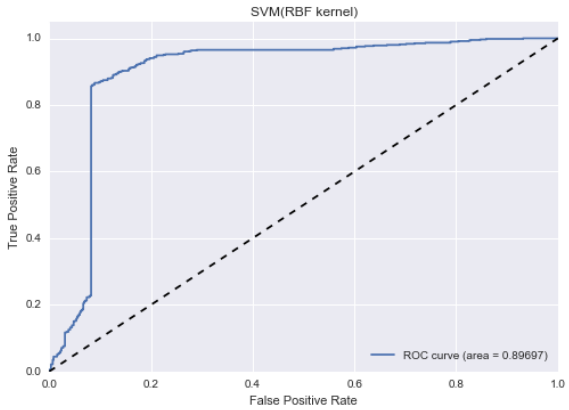
結果



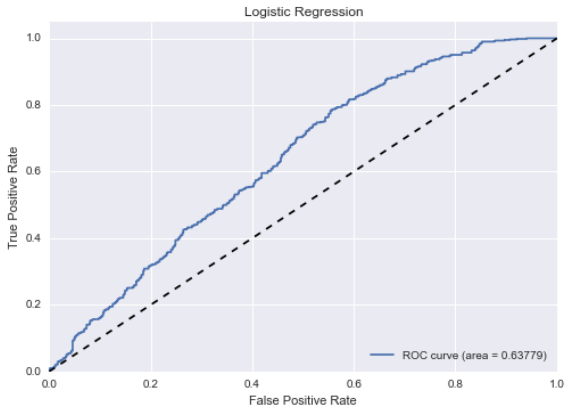
GBDT: AUC 0.95978



CNN: AUC 0.94335



SVM: AUC 0.89697



Logi: AUC 0.63779

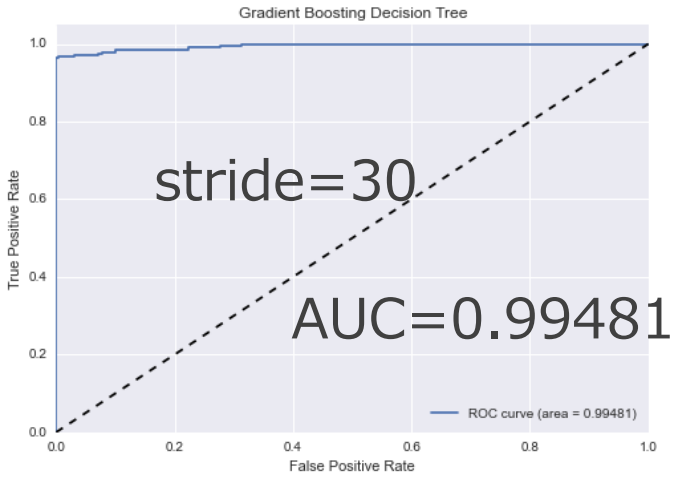
学習状況

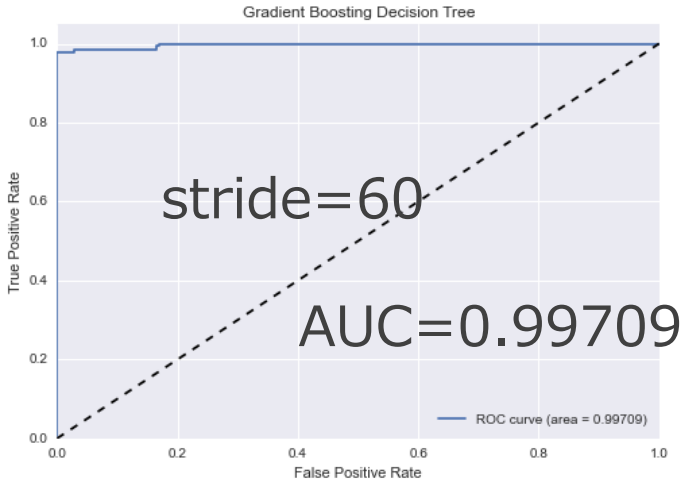
- training accuracyが1.0に達している状態で, test accuracyが0.9程度→trainとtestでaccuracyの差が大きい
- Predictの確率値が0.9999や0.00001などの極端な値を取る

Overfitting気味
→overfittingを解消するためtraining時からtestの状況に近づけるために,stride5によるdata augmentationをやめる
→stride30, 60で検証

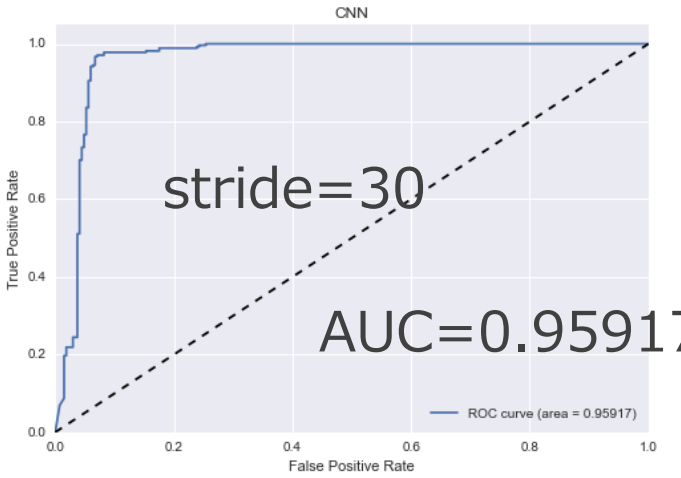
学習状況

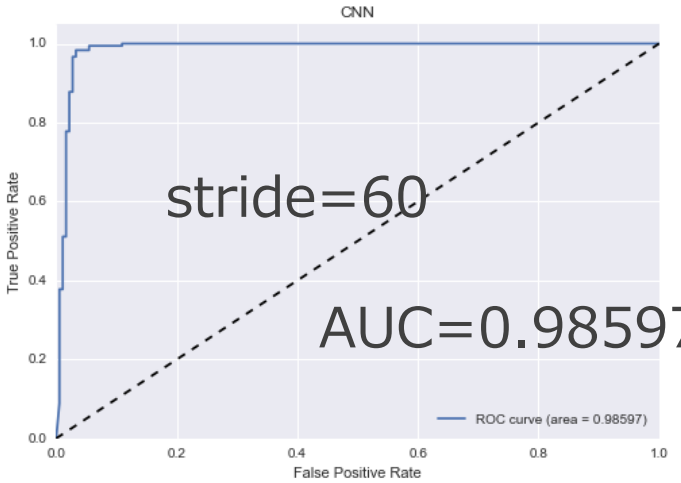
GBDT





CNN

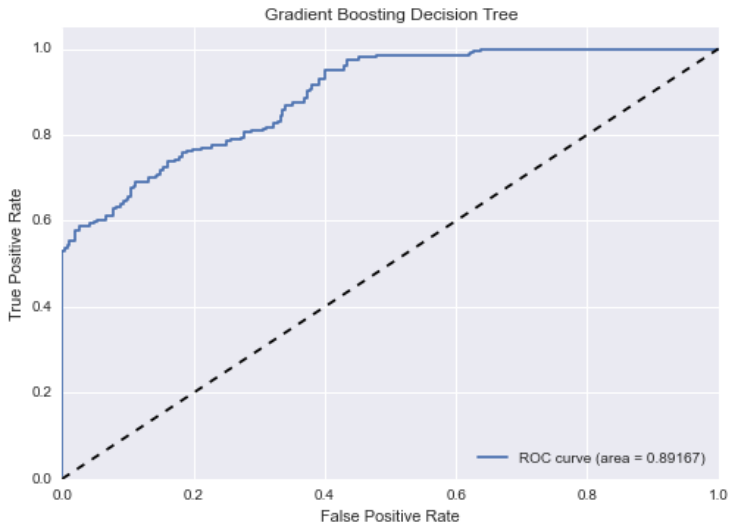




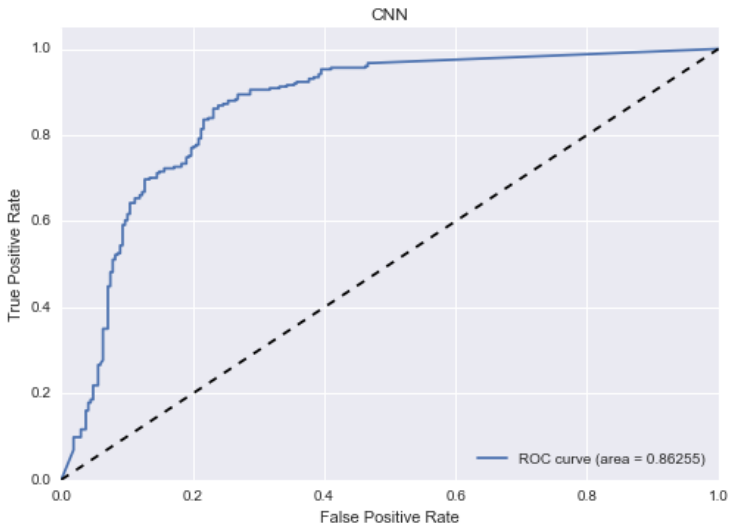
学習状況

学習時のtest accuracyの変化は, 省略するが問題なく下がっていた.

stride=60のGBDTで
stride=30のデータを分類



stride=60のCNNで
stride=30のデータを分類



strideの値ごとにoverfittingしている？

アイディア

GBDTとCNNで性能が異なる理由は？
→時系列の形状が多彩であり, それぞれ得意な領域がある
→テストデータ点からどの学習器が適しているかを求め,重み付き投票で予測を行う

まず $p_m(y|\boldsymbol{x}), m = cnn, gbdt$ を学習し,
あるテストデータ点x'について,
何らかの方法で予測に用いる学習器を重み付けする.

アイディア

1. 単純な手法→クラスタリング
    ・・・クラスタリングによってデータを適当に分割し，
    各クラスタに学習器を割り当てる

クラスタの分け方と学習器をCV
→p(x)の最適な学習器への割当てを求めようとしているに等しく、
かなり難易度が高い
→得られるクラスタと学習器の性質は基本的に無関係

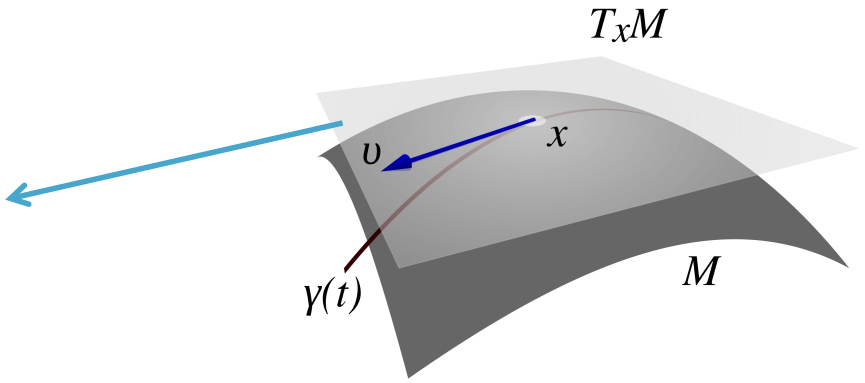
アイディア

2. テスト点近傍の情報を使う

テストデータ点に対してtangent spaceを考える
→tangent spaceでmaximize
→決定した重みでpredict

- tangent spaceは, あるテストデータ点からDTWで計算した距離が近い20点から形成されるとする
- tangent space内の近傍20点の正解ラベルに対する尤度が最大になるように学習器を重み付け

Tangent spaceをtest点
近傍のtraining dataで構成



DTW(Dynamic Time Warping)

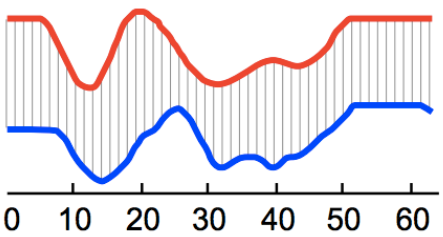
DTWの計算方法

2つの時系列X, Yについて, それぞれのデータ長がN, Mである時,

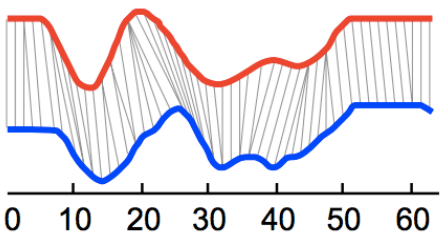
1. N×MのDTW距離を格納した類似度行列DTW[N][M]を作る.
2. 動的計画法によってDTW[0][0]～DTW[N-1][M-1]を全て求め,DTW[N-1][M-1]まで至る距離が最小となるパスを求める.
3. DTW[N-1][M-1]を時系列XとYの距離と定義する.

※時系列データの各点に対してその前後にwindowを定義し, その範囲内で類似度を計算している

時系列間の点の対応関係を, 多少の時間のズレや波形のブレにロバストになるように決定できる. これによって, **系列長の異なる時系列を比較**することができる.



ユークリッド距離での点の対応関係



DTWでの点の対応関係

定式化

tangent space内のtraining点をN点とし, そのインデックスをn, 学習器の個数をMとし, そのインデックスをmとすると,

$$\prod_{m}^{M} p_m(y^n|x^n)^{w_m}$$

によって予測を行う. この時,

学習器の重みは $\arg\max_{w_m} \prod_{n}^{N}\prod_{m}^{M} p_m(y^n|x^n)^{w_m}$ によって求められる.

対数を取ると, $\arg\max_{w_m} \sum_{n}^{N}\sum_{m}^{M} w_m \log p_m(y^n|x^n)$ の形になっているので, 線形な形になっており解析的に解ける. この際, 重みは0または1となる.

結果

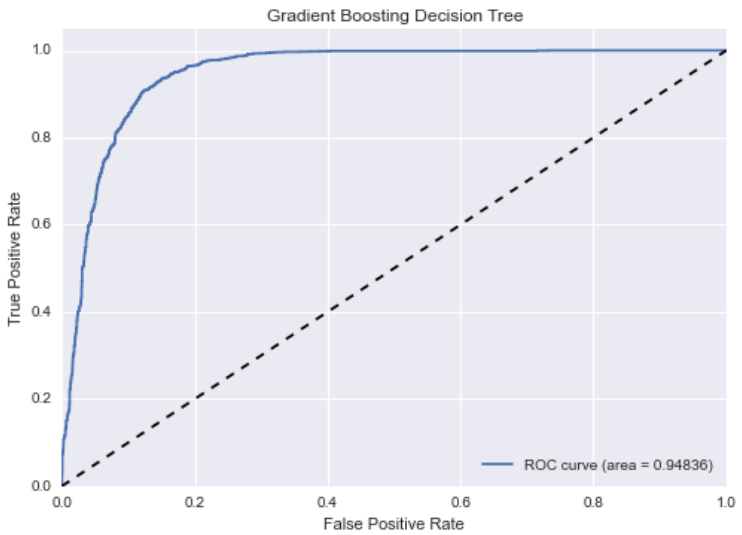
training accuracyが100%なので, そもそも学習器毎に苦手とする領域が存在せず, より確信度の高いpredictionが出やすいCNNが常に選択される.
→CNNは現状overfitting気味なので, training点に対する予測では0.999や0.0001といった極端な予測を行うが, GBDTでは, 決定木の多数決なので極端な予測が行われにくい
→問題を変更するかしないと, 今のままでは適用できなさそう

他のカテゴリでの精度

Ajaxのみを用いるのではなく, 全カテゴリーのデータで学習

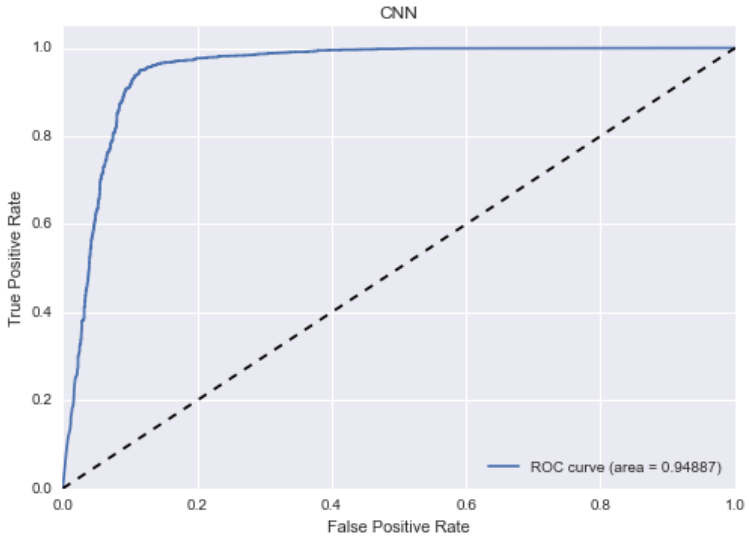
- Webページの数は合計1,577,000(最大がphotographyの437,000, 最小がajaxの56000)
- これらから, 入力において重なりがないようにstride=30で約20000点のトレーニングデータを取得(テストデータは5000点)

GBDT



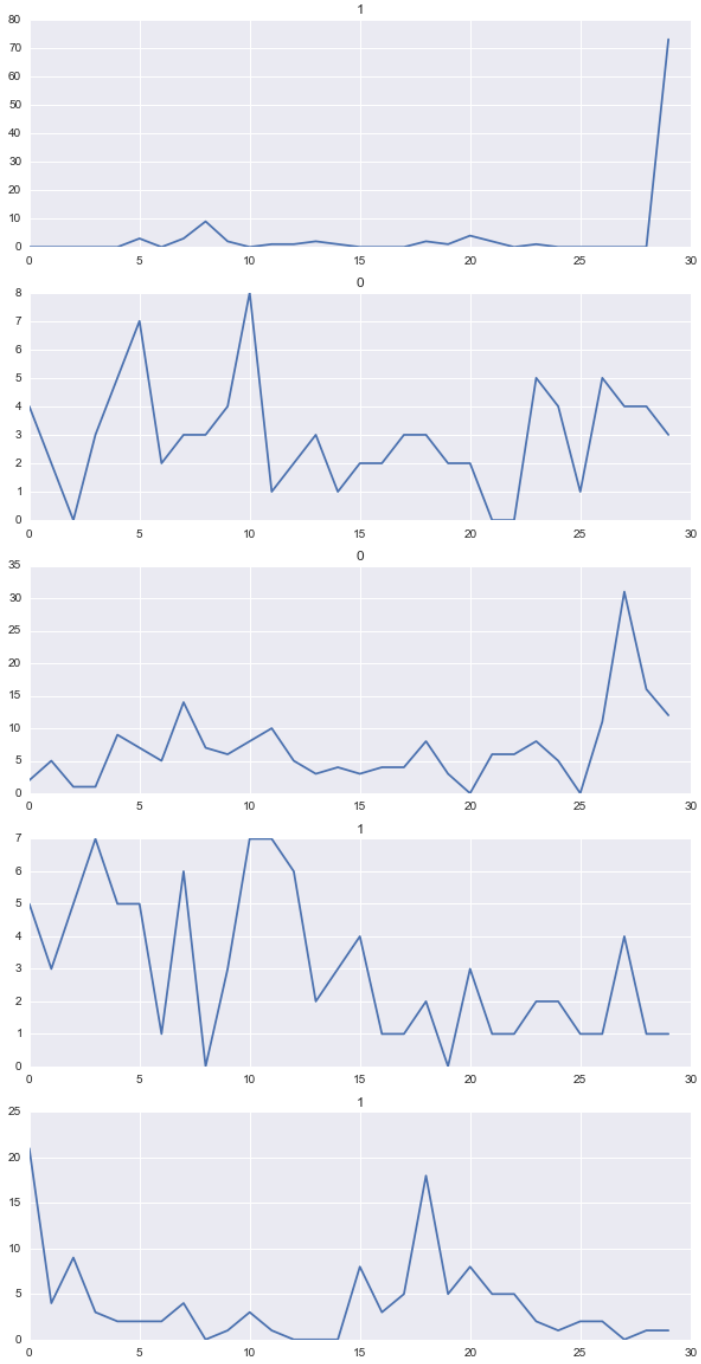
AUC=0.94836

CNN



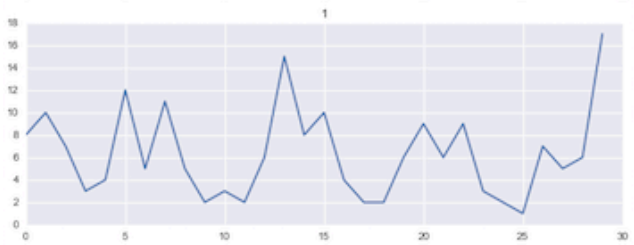
AUC=0.94887

ミスパターン

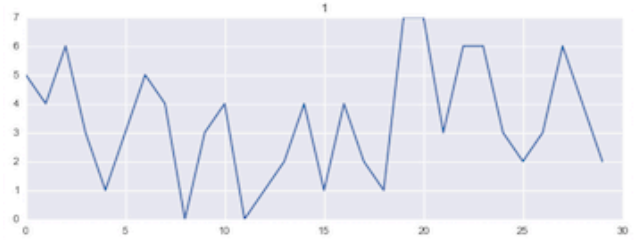


正解パターン

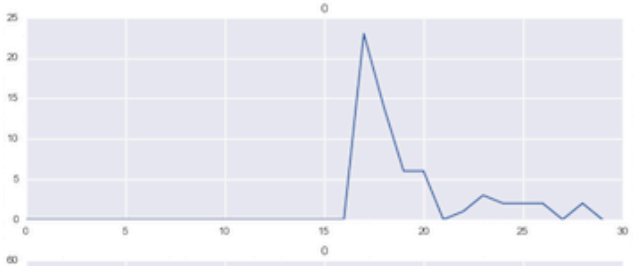
1



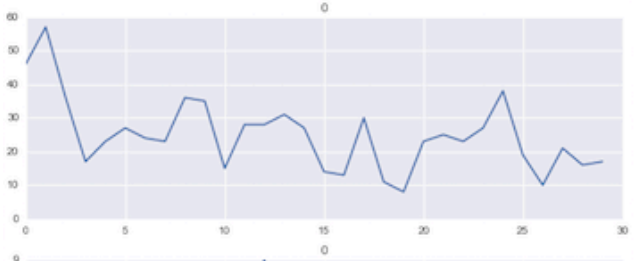
1



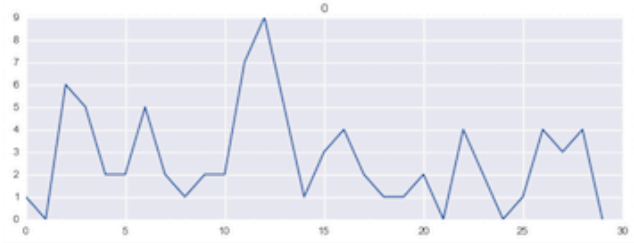
0



0

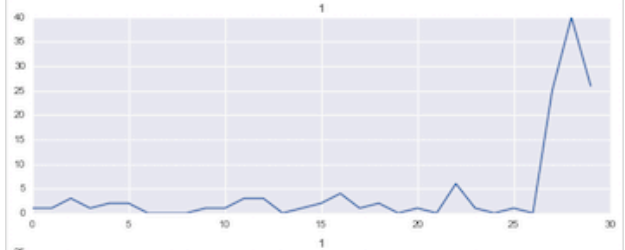


0

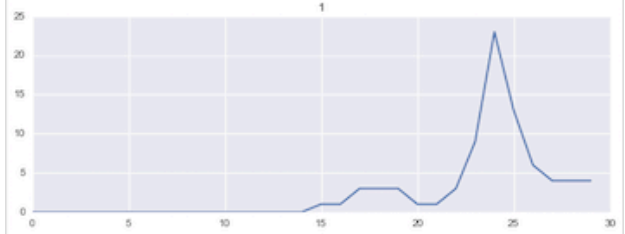


1

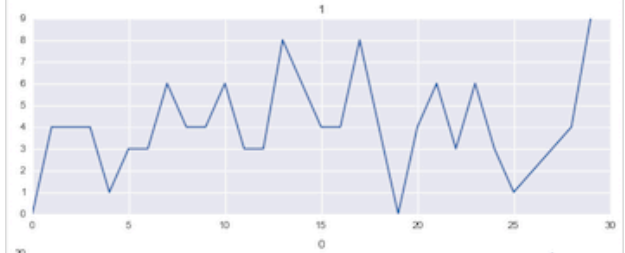
1



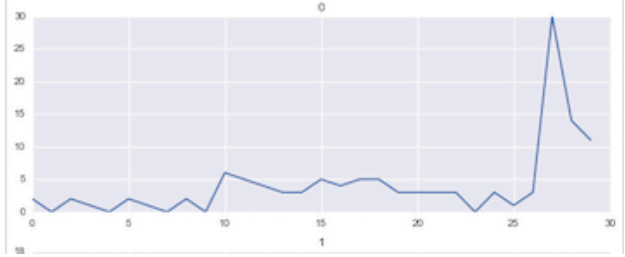
1



1



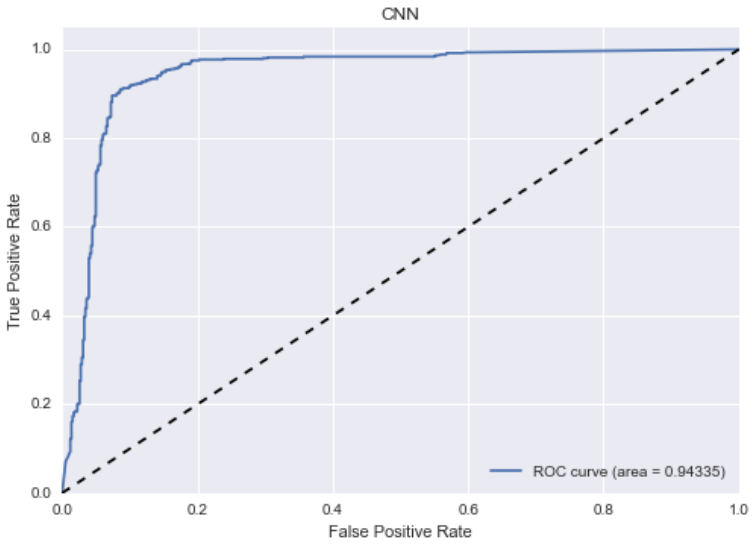
0



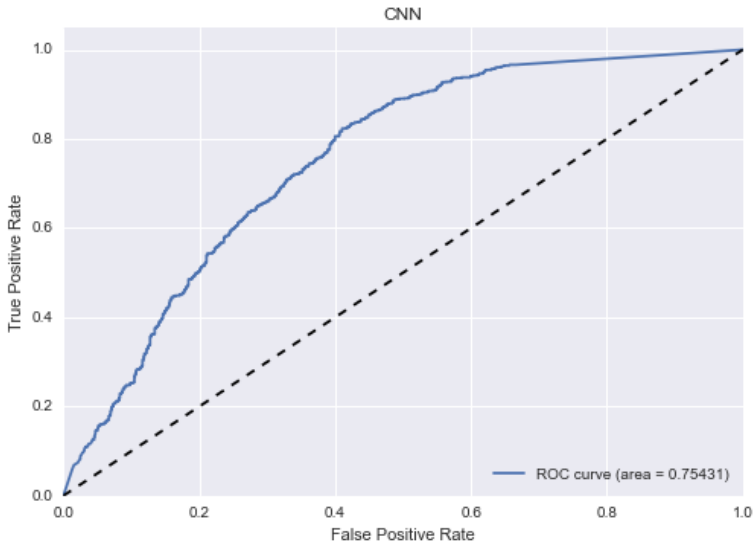
全カテゴリで学習したCNNの各カテゴリへの適用

全カテゴリを用いて学習したCNNを各カテゴリへ適用する

stride=30のajax



stride=5のajax

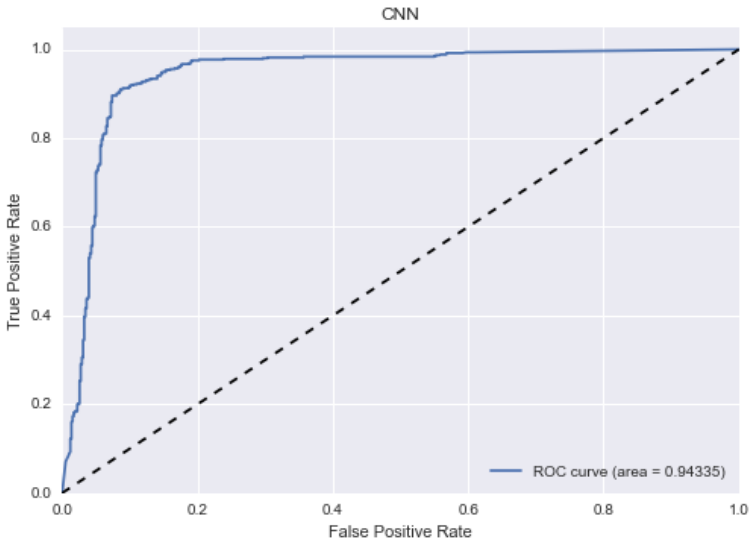


Ajax以外のほぼ全てのカテゴリーについてもほぼ同様な結果が得られた
→newsでのみ結果がやや異なった

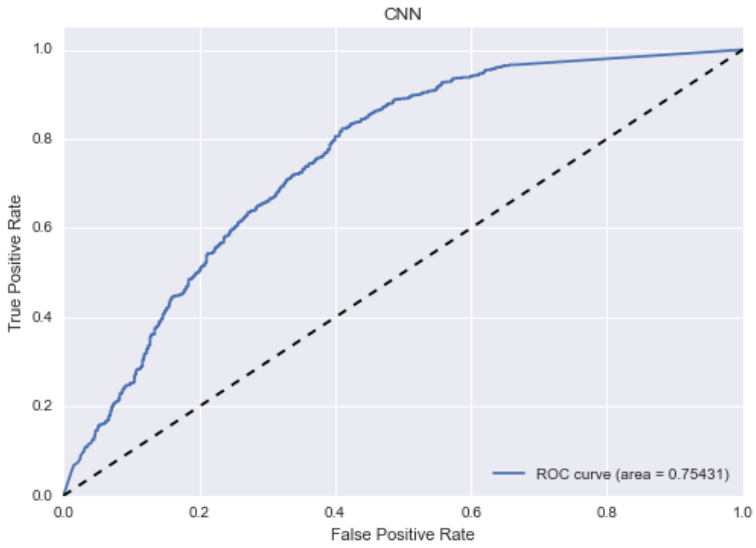
全カテゴリで学習したCNNの各カテゴリへの適用

全カテゴリを用いて学習したCNNを各カテゴリへ適用する

stride=30のajax



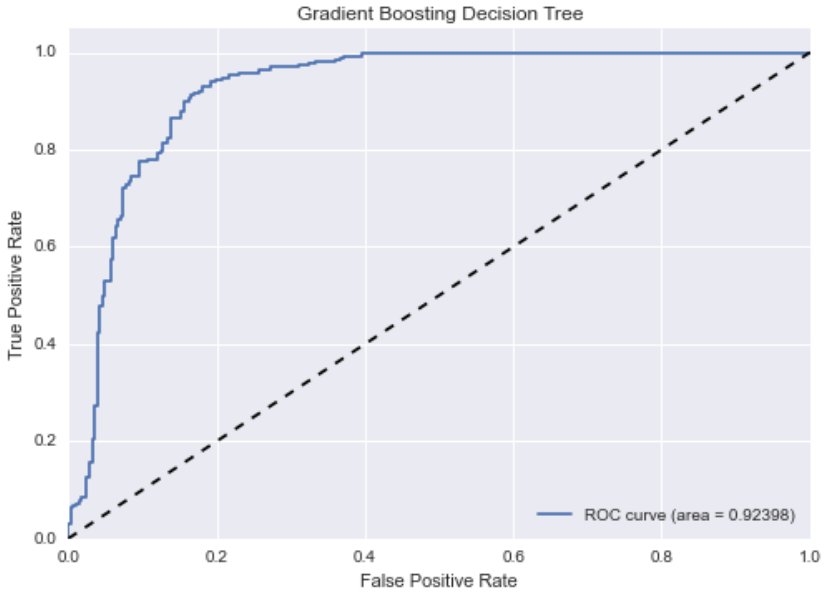
stride=5のajax



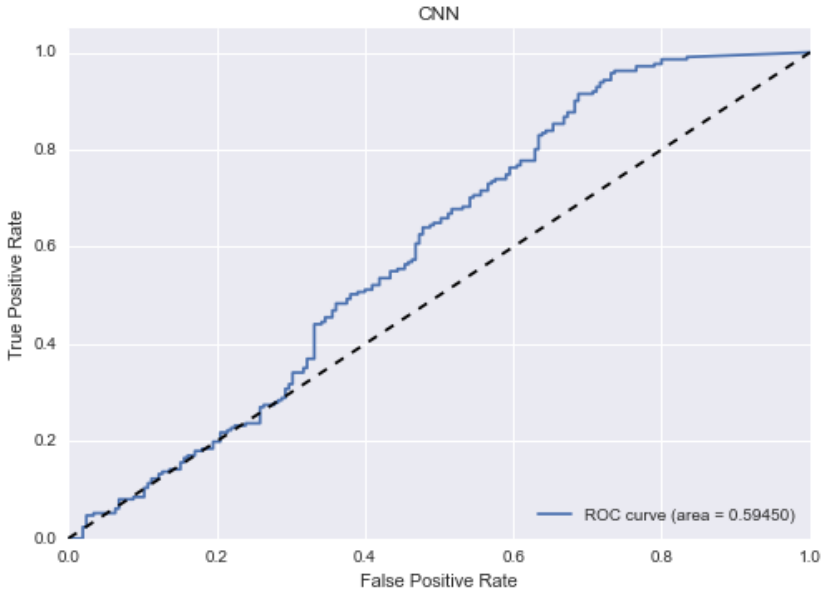
Ajax以外のほぼ全てのカテゴリーについてもほぼ同様な結果が得られた
→newsでのみ結果がやや異なった

全カテゴリで学習したCNNの各カテゴリへの適用

stride=30のnews

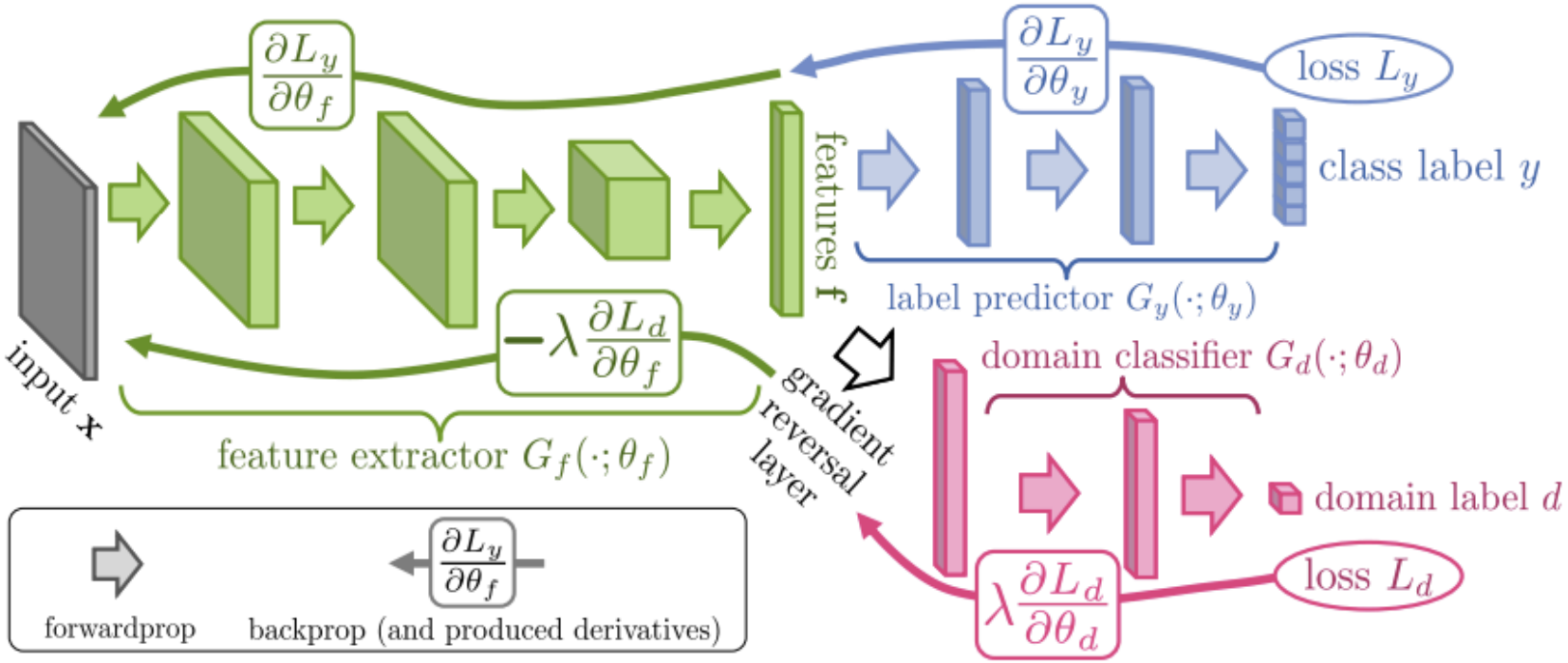


stride=5のnews



特にnewsのような一部のカテゴリでは,
bookmarkのされ方が大きく異なる？

Domain Adversarial Training



異なるが似ているドメインのデータセットに対して, 特徴量抽出を
共通のものにしたい時に使われる手法
→ただし, 学習が非常に難しいらしい
→単純に, 最終層をドメインごとに分ける方法の方が無難か

前回のスライド

シミュレーション設定

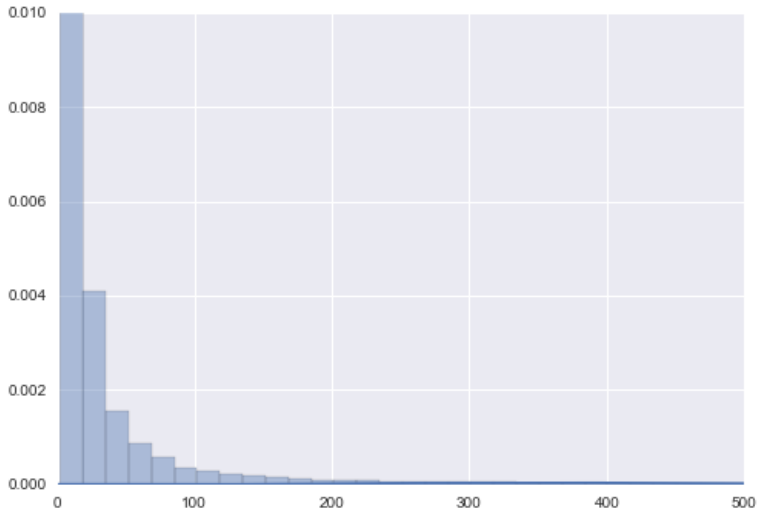
学習データ

- “ajax”カテゴリに属するWebサイト(約56000件)のうちbookmark数が100以上のもの(2076件).
- bookmark開始時点から60日分の時系列を取得し, 最初の30日のbookmark時系列を入力とする.
- 後半30日分の時系列からbookmark数の平均を計算し, 最初の30日(入力)の平均より大きければ1, 小さければ0とラベル付けする.
- ~~速度と加速度を計算して3×30の多チャンネル入力とした.~~

変更点:

- 速度と加速度をなくした
  →A/Bテストでほぼ結果が変わらなかったため

Ajaxにおけるbookmark数の分布



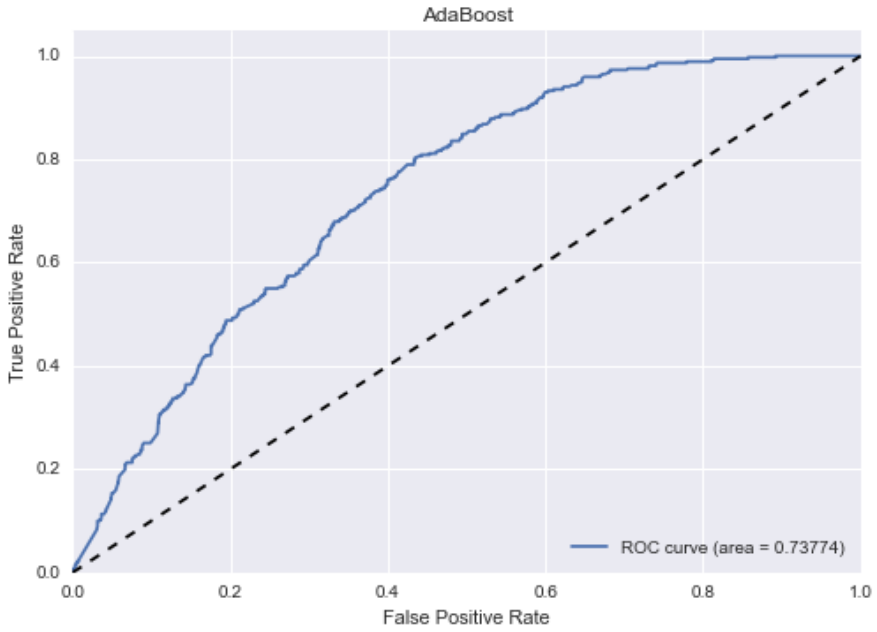
シミュレーション設定

その他

- 学習データでは, 正例と負例が同数となるようにサンプリング
- チャンスレベルは約80%
- 入力とする系列のbookmark数が少なすぎると, 将来的に現在よりbookmark数が伸びると判断されたとしてもその規模は小さい可能性があるので, 入力時系列の平均bookmark数に閾値を設定
  - とりあえず平均bookmark数 >= 2
- データシャッフル
- 精度評価はROC curveとprecision-recallによって行った
- Deep Neural Net, CNN以外の全ての手法について, パラメータのgrid searchを行った
  - DNN, CNNでは手動でパラメータを変更した

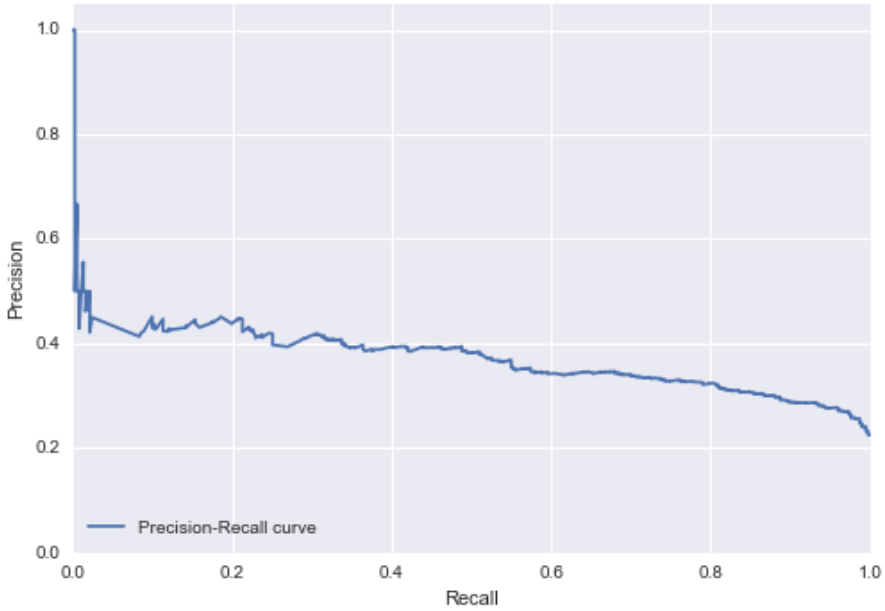
分類結果: Gradient Boosting Tree

ROC



AUC = 0.738

Precision-recall



average precision = 0.365

Convolutional Neural Network

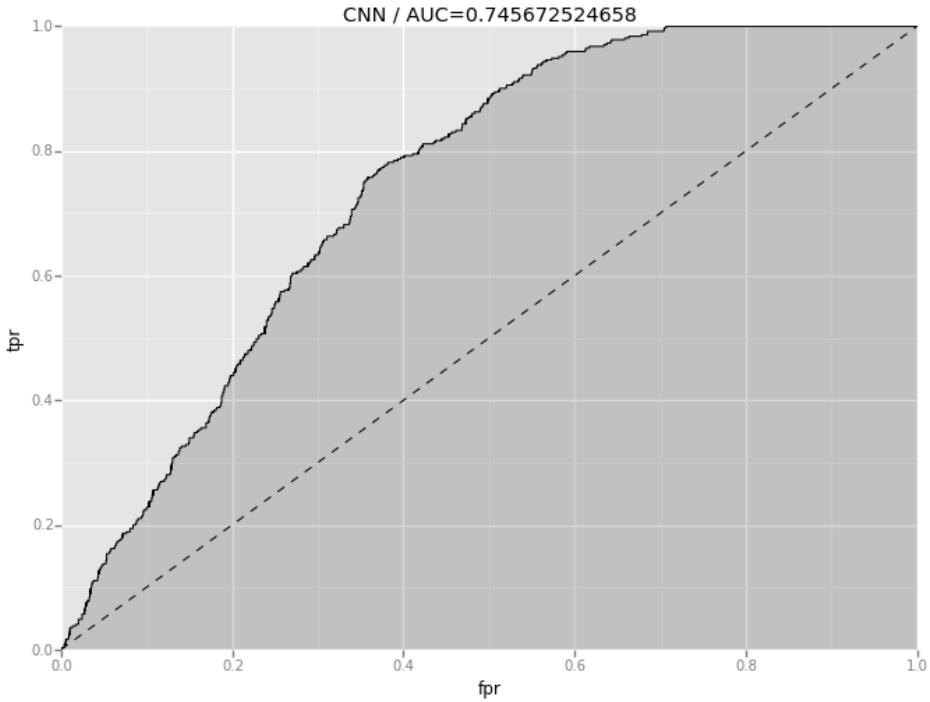
モデル

入力(30次元)
→1×5 convolutionで20枚にマップ→1×2 max pooling
→1×5 convolutionで50枚にマップ→1×2 max pooling
→full-connectで500次元にマップ(250→500)
→1次元の出力

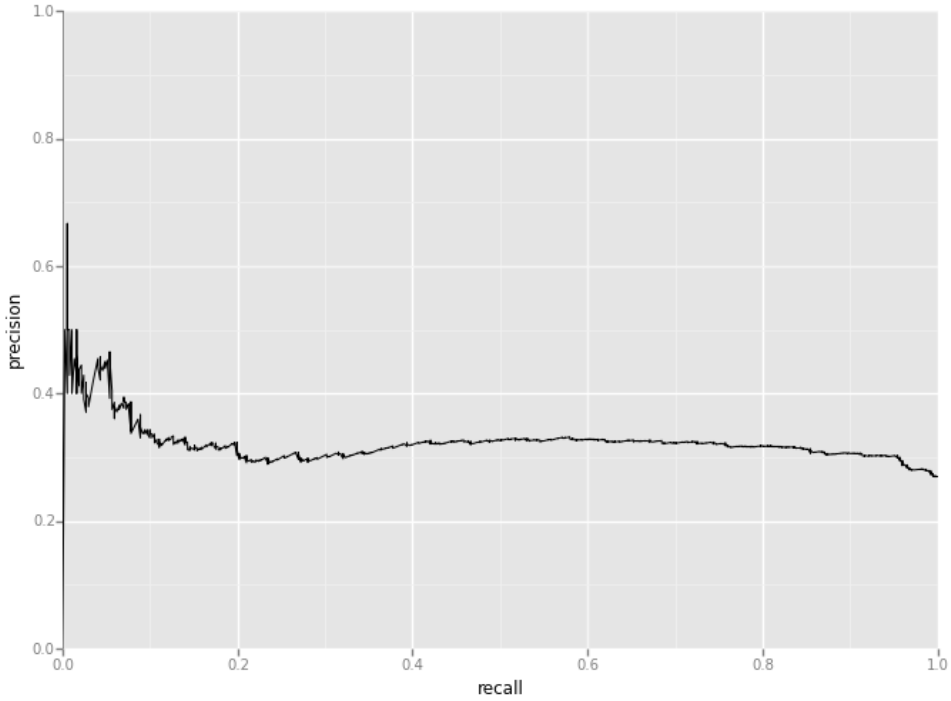
※CNNとNNで共通する設定

- fully-connect layerはDropoutで正則化
- ADAMで最適化(SGDと比較して精度向上あり)
- バッチサイズ50で50epoch学習
- クロスエントロピーロス
- 切断正規分布から重み初期値をサンプリング
- バイアスは初期値0を回避(微小値を加える)
- 0 padding

分類結果: Convolutional Neural Network



AUC=0.746



Average precision = 0.344

分類結果まとめ

各手法でのAUC, average precision

	Logistic Regression	SVM	Gradient Boosting Tree	Neural Net	CNN
AUC	0.605	0.634	0.738	0.699	0.746
average precision	0.235	0.319	0.365	0.334	0.344

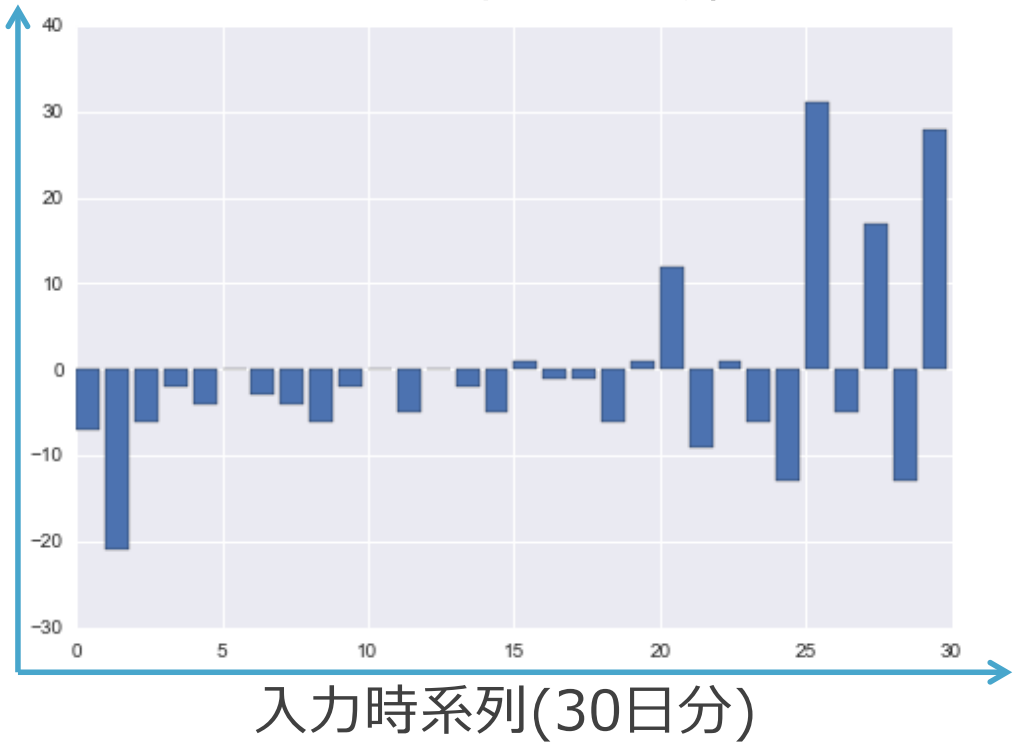
- データ数が少ないこともあり, 再現性はそこまで高くない. ±0.02程度ばらつく.
- AUCについて, GBT>CNNとなることもある
- 総じて見れば, GBTが最も精度が高いように見える

ロジスティック回帰とCNNのWeightを可視化

ロジスティック回帰

Weight
の値(×1000)

※値は1000
倍して小数点
以下切り捨て



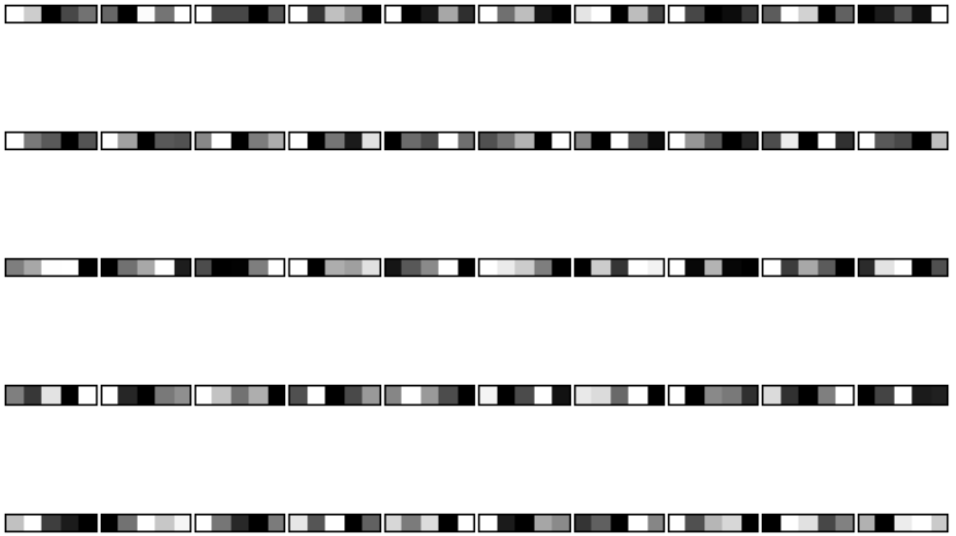
・ロジスティック回帰では入力時系列の中でも特に直近の部分に
weightが集中している
→直感に沿う結果
・前半にも多少weightが学習されている
→サイトは公開されてから数日でbookmark数が爆発することがよく
あり, それを反映していると考えられる

ロジスティック回帰とCNNのWeightを可視化

first convolution layer

1×5のfilter×20枚

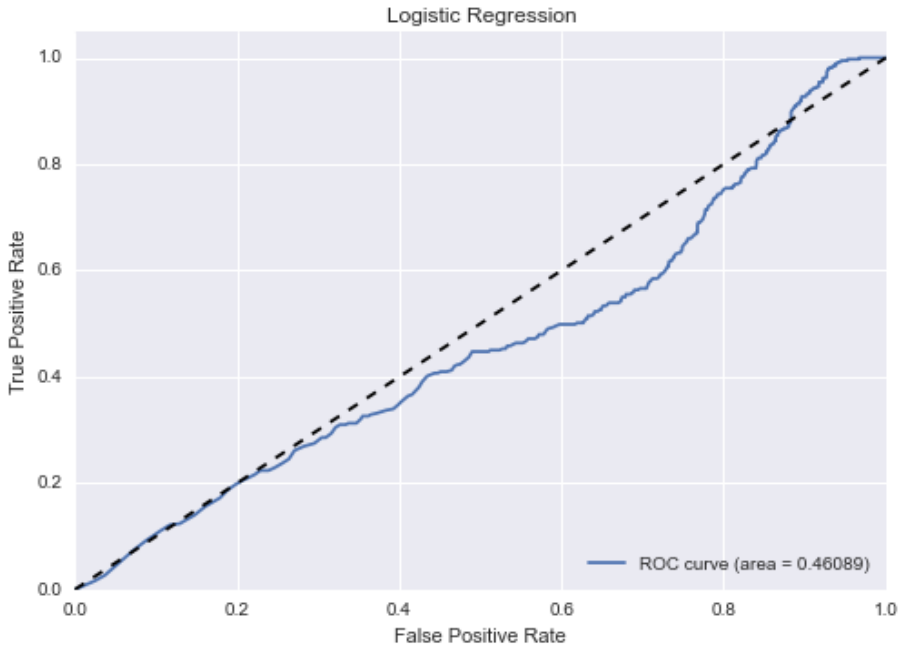
second convolution layer



1×5のfilter×50枚

・CNNのフィルターは解釈できていない
→ランダム？のように見える
→ただし, ロジスティック回帰<CNNとなっているので, 特徴抽出過程で何か良い特徴量が取れている？

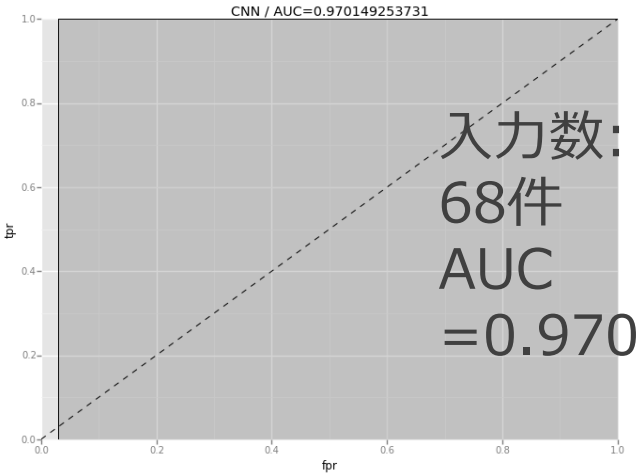
・ロジスティック回帰とCNNの性能に大きな差がある
→CNNの特徴抽出で差がでている
・入力の位置不変なbookmarkの盛り上がりをCNNではうまく捉えられているのでは？
→入力のbookmark数のみを特徴量にしたロジスティック回帰の性能を評価しよう



AUC = 0.46 になり,
全く予測できず

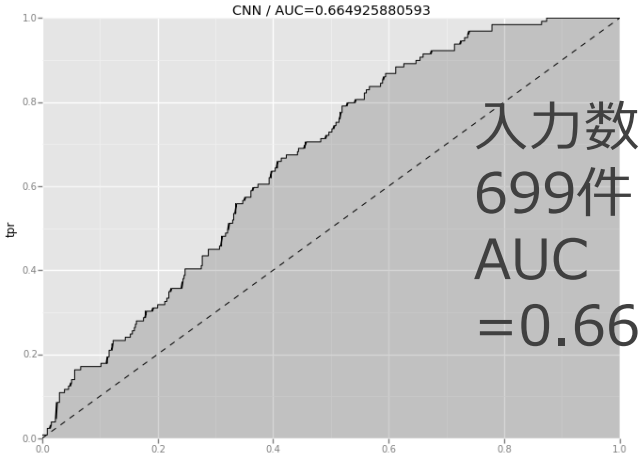
test入力の総bookmark数(日平均2以上なので2×30=60が最低ライン)によって性能は変化するか？
→bookmark数が500以上, 300以上, 100以下の三つのパターンで

Bookmark数>500



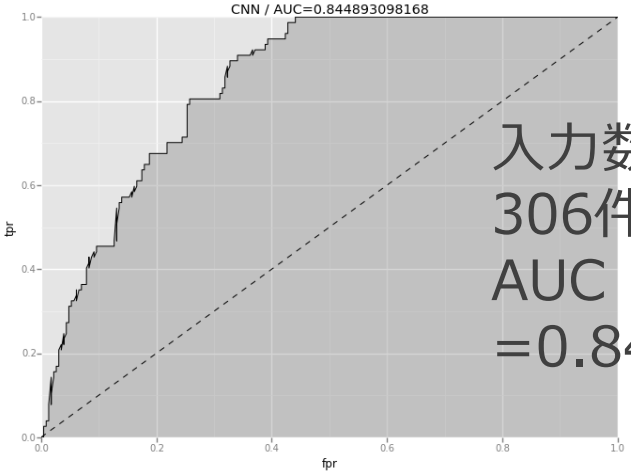
入力数:
68件
AUC
=0.970

Bookmark数<100



入力数:
699件
AUC
=0.665

Bookmark数>300



入力数:
306件
AUC
=0.845

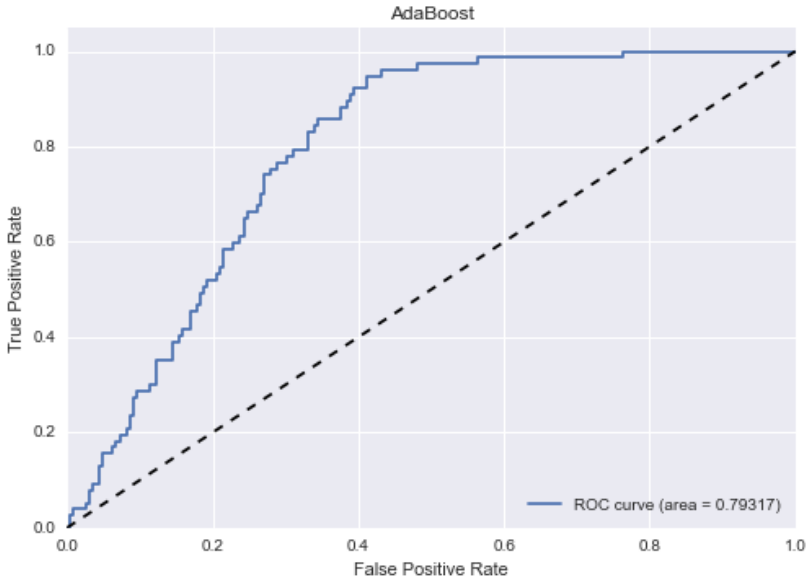
・bookmark数が確保されている場合ほど予測が上手くいっている
- 単に入力が大きければ今後はやらないだろう, という予測？
- 一日2bookmarkずつとかだと予測は難しい→ボリュームがあれば割とうまくいく？

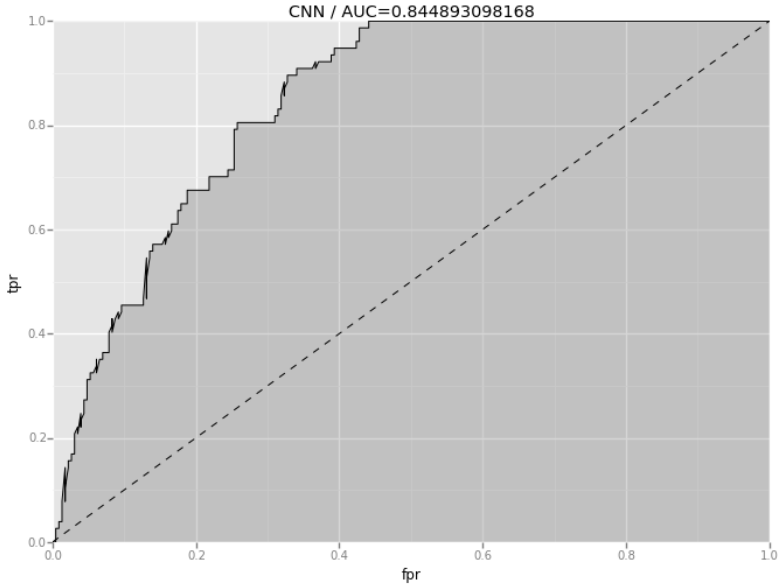
(ここまで前回のスライド)

Bookmark数300以上のWebサイトの分類では, CNNのAUCが上昇していた

→GBT(AdaBoost)でも同じ傾向が見られるか？

この条件ならCNNが勝つのか？





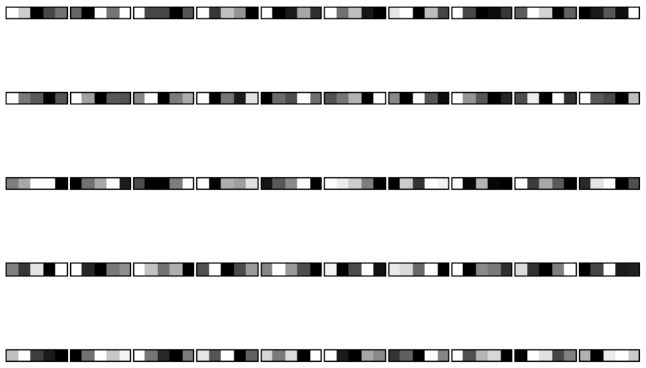
GBT: 0.793　　CNN:0.845

AUCで, GBTよりCNNの方がスコアが高くなる

学習されたモデルの可視化・・・大きく分けて二通り

1. 重みをニューロン毎にプロット
   →低レイヤーでは, 入力との対応がつきやすいため, 人間可読な結果が出やすい
2. あるニューロンの出力を最大化するような入力を求める
   →いわゆる猫ニューロンなどがこれにあたる. より深いレイヤーでは, 重みを可視化するだけでは何が何やらわからなくなるので, あるニューロン(あるいは出力)が強く反応するような入力を検出したり, 入力を変数としてニューロン出力を最大化する問題を勾配法で解く.

1.がこれまでやっていたような方法.(↓)



・・・今回は2.を使って評価したい

猫ニューロン論文:
http://static.googleusercontent.com/media/research.google.com/ja//archive/unsupervised_icml2012.pdf

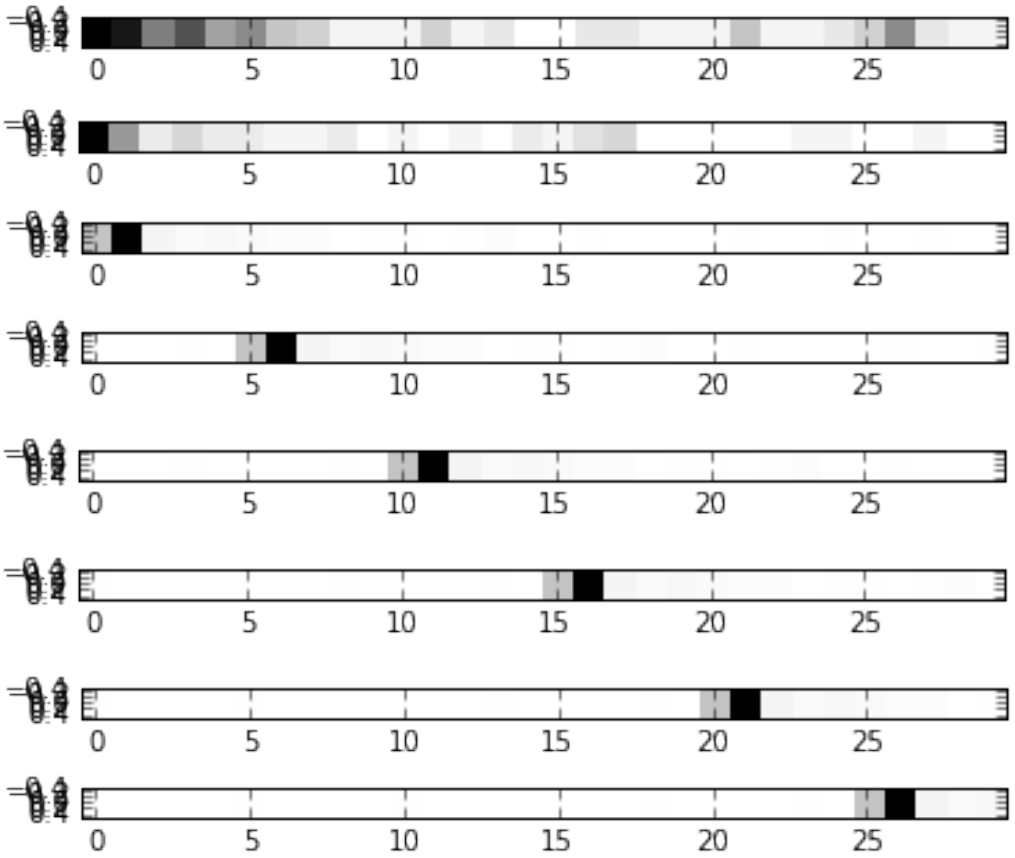
おじいさんニューロンが強く反応した入力例

CNNがどのような特徴抽出を行っているかを見たい

→test入力のうち, 0, 1と判別されるものそれぞれについて, スコアが高いものを列挙

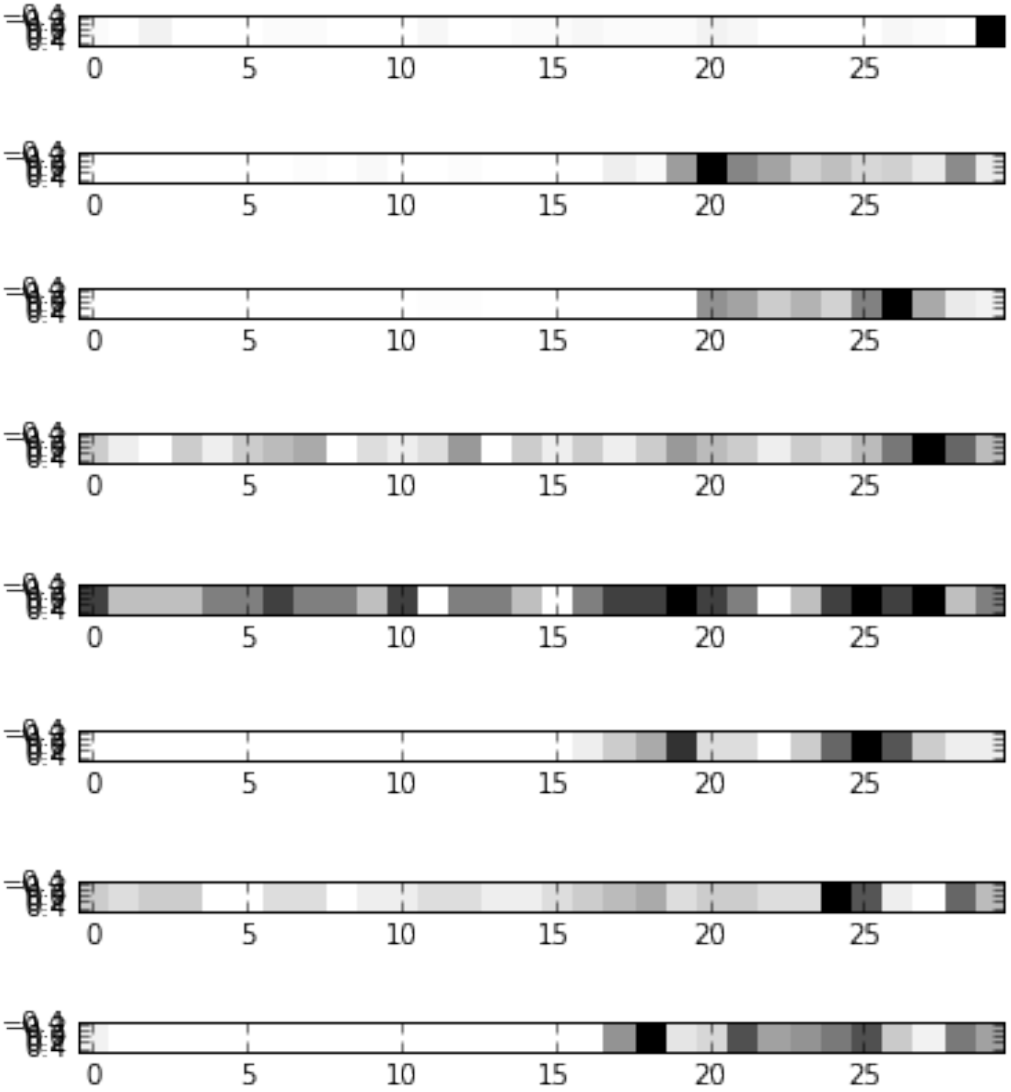
CNNがどのような特徴抽出を行っているかを見たい
→test入力のうち, 0, 1と判別されるものそれぞれについて,
　スコアが高いものを列挙

ラベル0(減少傾向)と推定された入力



・直感的に正しそうな結果
・一瞬流行ったものについては, それが一瞬であれば時期関係なく今後も見込みがない？

ラベル1(増加傾向)と推定された入力



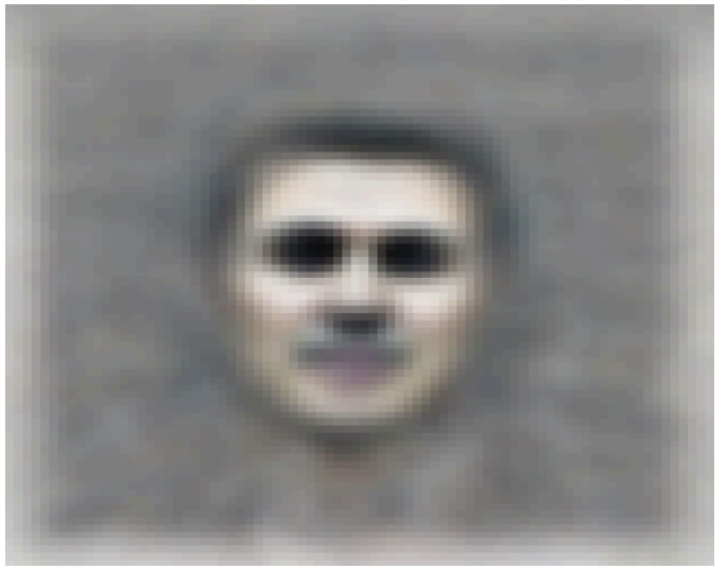
- ラベル0と推定されたものより後半に重みが集中している
- やや解釈しづらい

入力を変数として, あるニューロンが最も反応する入力を人工的に作る

・学習済みのモデルの重みを固定して, 勾配法によって入力変数を最適化する

$$x^* = \arg\min_x f(x; W, H), \ \text{subject to} \ ||x||_2 = 1.$$

例: おじいさんニューロン
おじいさん系の顔で出力が高くなるニューロンについて, 最も高い反応を示すであろう画像を仮想的に作成している.



・とりあえず出力の0, 1それぞれについて, 最も高い反応を示す入力を生成する
・シングルレイヤーでは動くことを確認
・CNNで上手く最適化されず困っています・・・