CS231n Convolutional Neural Networks for Visual Recognition

線形分類

前のセクションでは，画像の分類問題を紹介しました。さらに、学習セットの（注釈付きの）画像と比較することで画像にラベルを付けるk-近傍法(kNN)分類器について説明しました。これまで見てきたように、kNNには数々の欠点があります。

> ・分類器は学習データをすべて記憶し、将来のテストデータとの比較のために保存しなければなりません。データセットのサイズが容易にギガバイト級になるため、容量効率が悪くなる。
> ・テスト画像の分類は、すべてのトレーニング画像との比較を必要とするため、コストがかかる。

この章の概要を説明に入ります。私たちは今、画像分類に対するより強力なアプローチを開発しようとしていますが、これは最終的にはニューラルネットワークや畳み込みニューラルネットワーク(CNN)全体にまで拡張していくことになります。このアプローチには2つの主要なコンポーネントがあります：生(raw)データをクラスのスコアにマッピングするスコア関数と、予測されたスコアと正解ラベル(教師データ)の間の一致を定量化する損失関数です。次に、これを最適化問題として、スコア関数のパラメータに関して損失関数を最小化します。

画像からラベルスコアへのマッピングをパラメタライズする

このアプローチの最初の構成要素は、画像のピクセル値を各クラスの信頼度スコアにマッピングするスコア関数を定義することです。具体的な例を挙げてアプローチを展開していきます。

前述のように、画像 $x_i \in R^D$ の学習データセットを仮定して、それぞれがラベル y^i に関連付けられているとします. (但し、$i = 1...N, y_i \in 1...K$とする)
つまり、N個のサンプル(それぞれ次元数D)とK個の異なるカテゴリがあります。例えば、CIFAR-10では、10の異なるクラス（犬、猫、車…など）があるため、それぞれ次元数
$D = 32 \times 32 \times 3 = 3072$ピクセル、カテゴリ数$K = 10$、訓練画像数 $N = 50{,}000$のセットとなります。
ここでスコア関数を定義します。

$$f : R^D \mapsto R^K$$

この関数は生の画像のピクセルをクラスのスコアにマップするものです。

線形分類器。このモジュールでは、可能な限り最も単純な関数である線形写像から始めます。

$$f(x_i, W, b) = Wx_i + b$$

上記の式において、画像 x_i は、その全画素が形状[$D \times 1$]の1列ベクトルに再配列されていると仮定している。行列 W(= [$K \times D$]サイズ) とベクトル b (=[$K \times 1$]サイズ) は，この関数における変数です.

Wのパラメータはしばしば「重み」と呼ばれます。bはバイアスベクトルと呼ばれます。出力スコア(出力段)に影響を与えるものの、データ x_i自体には影響を与えないためです。俗に「パラメータ」と言った場合は「重み」のことを指します。

注意点:

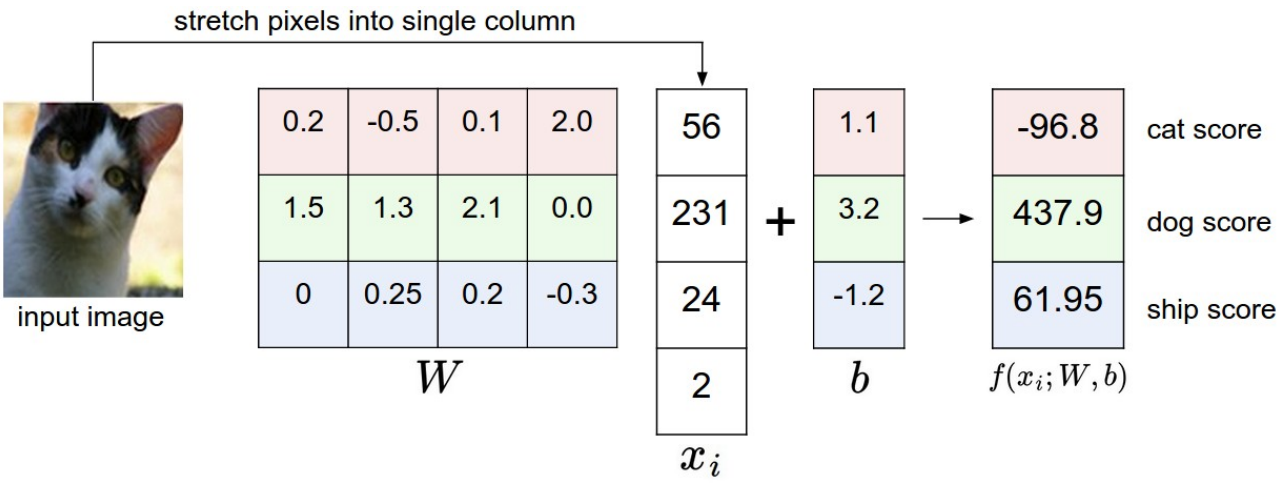
・単一行列の乗算 Wx_i は、各分類器が行列Wの行方向の10個それぞれの分類器（各クラスに1つ）を並列に評価していることに注意してください.
・また、入力データ(x_i, y_i)は固定のものと考えていますが、パラメータW, bの設定は制御できます。目標は、計算されたスコアが学習セット全体の正解ラベル(教師信号)と一致するように、これらのパラメータを設定することです。これがどのように行われるかについてはもっと詳しく説明しますが、直感的には、正しいクラスのスコアは不正確なクラスのスコアよりも高くなる事を期待できます。
・テスト画像の分類には、1つの行列の乗算と加算が含まれており、テスト画像をすべてのトレーニング画像と比較するよりもかなり高速です

先取り解説:

畳み込みニューラルネットワークは、上記のように画像ピクセルとスコアを正確にマッピングしますが、マッピング関数(f)はより複雑で、より多くのパラメータを含みます。

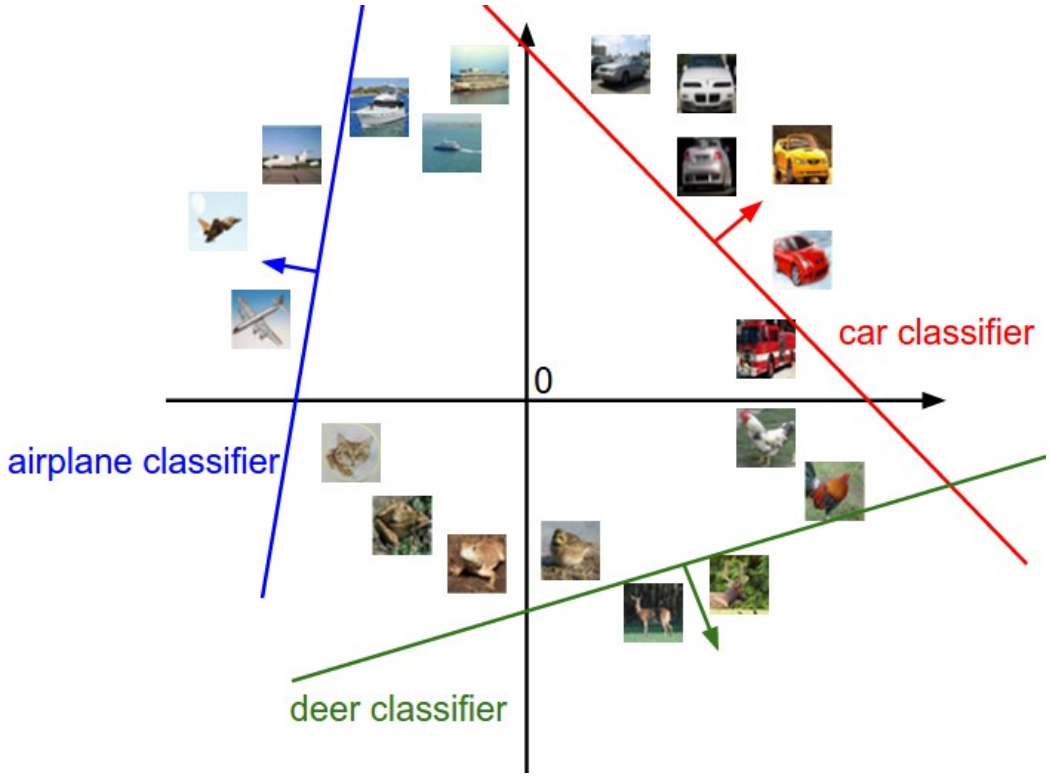
線形分類器の解析

線形分類器は、あるクラスのスコアを、RGBカラーチャンネルのすべてのピクセル値の重み付けされた合計として計算します。これらの重みにどのような値を設定するかによって、この関数は画像の特定の位置にある特定の色を（それぞれの重みの符号に応じて）好き嫌いする能力を持っています。例えば、画像の側面に青が多い場合（これは水に対応している可能性が高い）、「船」クラスの可能性が高いと想像することができます。この場合、「船」分類器は青チャンネルの重みには正の重みが多く（青の存在は船のスコアを増加させます）、赤/緑チャンネルの重みには負の重みが多く（赤/緑の存在は船のスコアを減少させます）、と予想するかもしれません。



上は画像と分類スコアの対応付けの例である。画像x_iのピクセル数は モノクロ4 ピクセル (この例では簡潔にするためにカラーチャンネルは考慮していません) としている。3 つの分類 (赤 (=猫)、緑 (=犬)、青 (=船) のクラス) があると仮定します。 (単に3つの分類を色分けして示しているだけであり、RGBチャンネルとは関係ありません。念のため)。画像のピクセルを1列に伸ばし、各クラスのスコアを得るために行列の乗算を行う。しかしここで取り上げている重みWのセットは全く良好な判定をしていない。重みWは、猫の画像であるにもかかわらず猫の分類へ低い判定を下しており、画像は犬であると判定している。

高次元の点としての画像の類推。画像は高次元の列ベクトルに引き伸ばされているので，各画像はこの空間における1つの点として解釈できます (例えば，CIFAR-10の各画像は，32x32x3ピクセル=3072次元空間における点とみなせる)．同様に、データセット全体はラベル付きの点の集合である。
各クラスのスコアをすべての画像ピクセルの重み付き和として定義したので、各クラスのスコアはこの空間上の線形関数となります。3072次元の空間を可視化することはできませんが，これらの次元をすべて2次元上に押しつぶすことを想像すると，分類器が何をしているのかを可視化することができます.



上の図は各画像を1点とし、3つ(航空機・車・鹿)の分類器を可視化した画像空間の視覚的表現。車の分類器を例にすると（赤線）、赤線は、車のクラスのスコアが0になる空間内のすべての点を示している。赤い矢印は増加方向を示しているので、赤線の右側にある点はすべて正の（直線的に増加する）スコアを持ち、赤線の左側にある点はすべて負の（直線的に減少する）スコアになる。

上記から分かる通り、Wの各行はクラスの1つの分類器です。これらを幾何学的に考えると、Wの行の1つを変えることによってピクセル空間の対応する線が異なる方向に回転するということです。一方、バイアスbを変更すると、分類器は線を平行移動させます。特に、バイアス項がない場合、$x_i = 0$を入力すると、重みに関係なく常に0となり、すべての分類器の線が原点を遠らざるを得ないことに注意してください。

線形分類器のテンプレートマッチングとしての解釈。重みWのもう一つの解釈は，Wの各行がクラスの1つのテンプレート（またはプロトタイプと呼ばれることもある）に対応するということです．画像に対する各クラスのスコアは，内積（またはドット積）を用いて各テンプレートを画像と比較し，最も「フィットする」ものを見つけることによって得られます．この用語を用いて表現するのであれば、線形分類器はテンプレートを学習する「テンプレートマッチング」を行っていることになります。別の考え方としては、効果的に最近傍探索を行っていますが、(kNNのように)何千もの学習画像を持つ代わりに、クラスごとに1つの画像を使用しているだけです。また、L1やL2の距離の代わりに、（負の）内積を距離として使用します。

ちょっと先取り：上記はCIFAR-10の学習終了時に学習した重みの例。例えば、船のテンプレートには、予想通り青いピクセルが多く含まれています。そのため、このテンプレートは、海の上の船の画像との内部積を照合すると、高いスコアが得られます。

上図の馬のテンプレートには2頭身の馬が含まれているように見えますが、これはデータセットの左向きと右向きの馬の両方が含まれているためです。線形分類器は、データ中の馬のこれら2つのモードを1つのテンプレートに統合します。同様に、車の分類器は、いくつかの車種や向きといったモードを結合して1つのテンプレートにしたようです。また、CIFAR-10データセットには、他のどの色の車よりも赤い車が多いことも示唆しています。線形分類器は色の異なる車を適切に識別には弱すぎますが、後に見るように、ニューラルネットワークはこのタスクを実行できるようにしてくれます。少し先の話ですが、ニューラルネットワークは、隠れた層で特定の車のタイプ（例えば、左向きの緑の車、前向きの青の車など）を検出する中間層ニューロンを開発することができ、次の層のニューロンは、個々の車検出器の重み付けされた合計を通して、これらをより正確な車のスコアに結合することができるようになるでしょう。

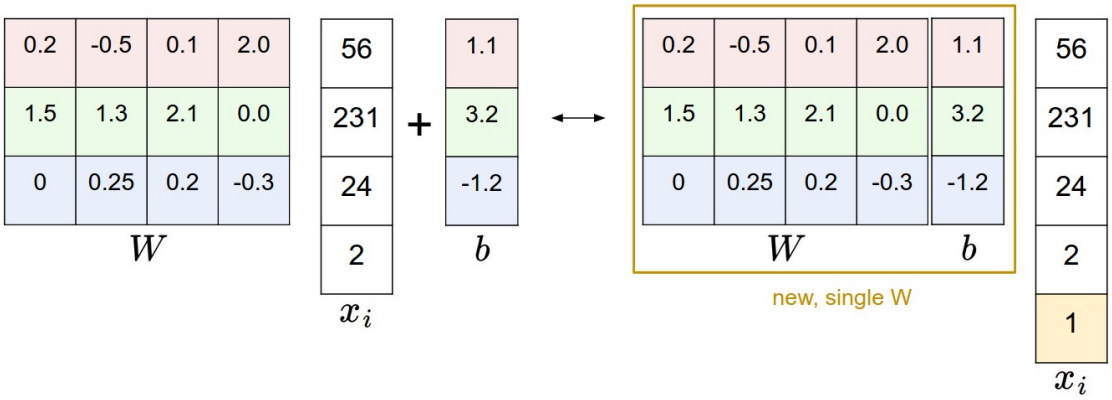
バイアスのトリック。先に進む前に、2つのパラメータW, bを1つのものとして表現するための一般的な単純化のトリックについて触れておきたいと思います。スコア関数を次のように定義したことを思い出してください。

$$f(x_i, W, b) = Wx_i + b$$

2 つのパラメータセット（バイアス b と重み W）を別々に追跡するのは少し面倒です。一般的に使用されるトリックは、ベクトルx_iを、常に定数 1 –デフォルトのバイアス次元を保持する1 つの次元– を追加して、2 つのパラメータセットを 1 つの行列に結合することです。次元が追加されると、新しいスコア関数は単一の行列の乗算に単純化されます。

$$f(x_i, W) = Wx_i$$

CIFAR-10の例では、x_iは$[3072 \times 1]$の代わりに$[3073 \times 1]$になり（追加した次元が定数1を保持しています）、W は$[10 \times 3072]$のではなく$[10 \times 3073]$となりました。W が現在のバイアスbに対応する追加された列は、下図のように図示することで明確になるかもしれません。



バイアストリックの図解。行列の乗算を行ってからバイアスベクトルを加算すること（左）は，すべての入力ベクトルx_iに定数1のバイアス次元を追加し，重み行列Wにバイアス列bを1列拡張することと等価です（右）．このように，すべてのベクトルに1を加えることでデータを前処理する場合，重みとバイアスを保持する2つの行列の代わりに，重みの1つの行列を学習するだけでよい．