05-features

June 14, 2020

1 Atividade 05: Utilizando Descritores de Imagem

Complete e entregue toda essa atividade (incluindo suas saídas e qualquer código adicional que você desenvolva) juntamente com a submissão de seu trabalho prático. Maiores detalhes podem ser vistos na página da disciplina.

Você viu que é possível se alcançar um resultado razoável na tarefa de classificação de imagens por meio do treinamento de um classificador linear utilizando os pixels da imagem como entrada.

Nesta atividade, você irá explorar a possibilidade de melhorar o desempenho de clssificação por meio do treinamento de um classificador não sobre os pixels brutos mas sobre descritores (ou características) calculados a partir dos pixels brutos.

Nesta atividade, você irá:

- extrair **descritores** (ou **características**) a partir das imagens do dataset
- treinar um **classificador SVM sobre dos descritores** extraídos e avaliar seu desempenho de classificação
- treinar uma **rede neural sobre dos descritores** extraídos e avaliar seu desempenho de classificação

```
In [1]: import random
        import numpy as np
        from dl.data_utils import load_CIFAR10
        import matplotlib.pyplot as plt

        from __future__ import print_function

        %matplotlib inline
        plt.rcParams['figure.figsize'] = (10.0, 8.0)
        plt.rcParams['image.interpolation'] = 'nearest'
        plt.rcParams['image.cmap'] = 'gray'

        %load_ext autoreload
        %autoreload 2

        print('Okay!')
```

```
Okay!
```

1.1 Leitura de dados

Similar aos exercícios anteriores, você vai carregar os dados do CIFAR-10 dataset a partir do disco.

```
In [3]: from dl.features import color_histogram_hsv, hog_feature

        def get_CIFAR10_data(num_training=49000, num_validation=1000, num_test=1000):
            # Carregga os dados CIFAR-10 brutos
            cifar10_dir = 'dl/datasets/cifar-10-batches-py'
            X_train, y_train, X_test, y_test = load_CIFAR10(cifar10_dir)

            # Subdivide os dados em conjuntos
            mask = list(range(num_training, num_training + num_validation))
            X_val = X_train[mask]
            y_val = y_train[mask]
            mask = list(range(num_training))
            X_train = X_train[mask]
            y_train = y_train[mask]
            mask = list(range(num_test))
            X_test = X_test[mask]
            y_test = y_test[mask]

            return X_train, y_train, X_val, y_val, X_test, y_test

        X_train, y_train, X_val, y_val, X_test, y_test = get_CIFAR10_data()

        print('Okay!')
```

```
Okay!
```

1.2 Extração de Descritores (ou Características)

Para imagem, você irá calcular um Histograma de Gradientes Orientados (HOG - Histogram of Oriented Gradients), bem como um histograma de cores utilizando o canal de matiz (*hue*) do espaço de cores HSV. O descritor final de cada imagem será um vetor obtido pela concatenação dos vetores correspondendes ao HOG e ao histograma de cor.

De forma simplificada, pode-se dizer que o HOG captura a informação de textura da imagem ignorando informações sobre cores, enquanto o histograma de cor representa a cor da imagem de entrada ignorando a textura. Dessa forma, espera-se que o uso dos dois em conjunto produza um resultado melhor que o obtido por qualquer um dos dois separadamente. Verificação dessa suposição poderia ser uma possível tentativa para ser realizada na seção de bonus.

Ambas as funões `hog_feature` e `color_histogram_hsv` operam sobre uma única imagem e retornam um vetor de características (descritor) para essa imagem. Assim, a função de extração de descritores `extract_features` recebe como entrada um conjunto de imagens e uma lista de funções de extração de descritores. Em seguida, ela executa cada uma das funções de extração de descritores sobre cada imagem do conjunto, armazenando os resultados em uma matriz em que cada linha está associada a uma imagens e cada coluna representa a concatenação de todos os vetores de características obtidos para essa imagem.

In [4]:
```
from dl.features import *

num_color_bins = 10 # Número de slots (bins) do histograma de cor
feature_fns = [hog_feature, lambda img: color_histogram_hsv(img, nbin=num_color_bins)]
X_train_feats = extract_features(X_train, feature_fns, verbose=True)
X_val_feats = extract_features(X_val, feature_fns)
X_test_feats = extract_features(X_test, feature_fns)

# Preprocessamento: Subtrair o descritor médio
mean_feat = np.mean(X_train_feats, axis=0, keepdims=True)
X_train_feats -= mean_feat
X_val_feats -= mean_feat
X_test_feats -= mean_feat

# Preprocessamento: Dividir pelo desvio padrão. Isto assegura que cada descritor
# possui aproximadamente a mesma escala.
std_feat = np.std(X_train_feats, axis=0, keepdims=True)
X_train_feats /= std_feat
X_val_feats /= std_feat
X_test_feats /= std_feat

# Preprocessamento: acrescenta uma dimensão de viés (bias) necessária ao SVM
X_train_feats = np.hstack([X_train_feats, np.ones((X_train_feats.shape[0], 1))])
X_val_feats = np.hstack([X_val_feats, np.ones((X_val_feats.shape[0], 1))])
X_test_feats = np.hstack([X_test_feats, np.ones((X_test_feats.shape[0], 1))])

print('Okay!')
```

```
Done extracting features for 1000 / 49000 images
Done extracting features for 2000 / 49000 images
Done extracting features for 3000 / 49000 images
Done extracting features for 4000 / 49000 images
Done extracting features for 5000 / 49000 images
Done extracting features for 6000 / 49000 images
Done extracting features for 7000 / 49000 images
Done extracting features for 8000 / 49000 images
Done extracting features for 9000 / 49000 images
Done extracting features for 10000 / 49000 images
Done extracting features for 11000 / 49000 images
Done extracting features for 12000 / 49000 images
Done extracting features for 13000 / 49000 images
Done extracting features for 14000 / 49000 images
Done extracting features for 15000 / 49000 images
Done extracting features for 16000 / 49000 images
Done extracting features for 17000 / 49000 images
Done extracting features for 18000 / 49000 images
Done extracting features for 19000 / 49000 images
Done extracting features for 20000 / 49000 images
```

```
Done extracting features for 21000 / 49000 images
Done extracting features for 22000 / 49000 images
Done extracting features for 23000 / 49000 images
Done extracting features for 24000 / 49000 images
Done extracting features for 25000 / 49000 images
Done extracting features for 26000 / 49000 images
Done extracting features for 27000 / 49000 images
Done extracting features for 28000 / 49000 images
Done extracting features for 29000 / 49000 images
Done extracting features for 30000 / 49000 images
Done extracting features for 31000 / 49000 images
Done extracting features for 32000 / 49000 images
Done extracting features for 33000 / 49000 images
Done extracting features for 34000 / 49000 images
Done extracting features for 35000 / 49000 images
Done extracting features for 36000 / 49000 images
Done extracting features for 37000 / 49000 images
Done extracting features for 38000 / 49000 images
Done extracting features for 39000 / 49000 images
Done extracting features for 40000 / 49000 images
Done extracting features for 41000 / 49000 images
Done extracting features for 42000 / 49000 images
Done extracting features for 43000 / 49000 images
Done extracting features for 44000 / 49000 images
Done extracting features for 45000 / 49000 images
Done extracting features for 46000 / 49000 images
Done extracting features for 47000 / 49000 images
Done extracting features for 48000 / 49000 images
Okay!
```

1.3 Treinamento do Classificador SVM sobre os Descritores

Usando o código desenvolvido anteriormente para um classificador SVM multiclasse, você deve treinar vários SVMs sobre os descritores extraídos acima. Isto deve alcançar resultados melhores que os obtidos pelo treinamento de SVMs diretamente sobre os pixels brutos.

In [13]:
```
# Usar o conjunto de validação para ajustar a taxa de apredizado e regularização

from dl.classifiers.linear_classifier import LinearSVM

learning_rates = [1e-9, 1e-8, 1e-7]
regularization_strengths = [5e4, 5e5, 5e6]

results = {}
best_val = -1
best_svm = None
```

```
        ################################################################################
        # TODO:                                                                        #
        # Escrever o código que escolhe os melhores hiperparâmetros usando o conjunto  #
        # de validação. Deve ser muito similar ao código feito para o SVM.             #
        # Além disso, deve-se armazenar a melhor acurácia de validação em best_val     #
        # e o objeto LinearSVM que obteve esse resultado em best_svm.                  #
        # Você pode experimentar com diferentes número de shots (bins) no histograma   #
        # de cor. Se você proceder corretamente deve ser capaz de obter uma acurácia   #
        # próxima de 0.44 no conjunto de validação.                                    #
        ################################################################################
        iters = 2000 #100
        for lr in learning_rates:
            for rs in regularization_strengths:
                svm = LinearSVM()

                svm.train(X_train_feats, y_train, learning_rate=lr, reg=rs, num_iters=iters)

                y_train_pred = svm.predict(X_train_feats)
                acc_train = np.mean(y_train == y_train_pred)
                y_val_pred = svm.predict(X_val_feats)
                acc_val = np.mean(y_val == y_val_pred)

                results[(lr, rs)] = (acc_train, acc_val)

                if best_val < acc_val:
                    best_val = acc_val
                    best_svm = svm
        ################################################################################
        #                              END OF YOUR CODE                                #
        ################################################################################

        # Exibe os resultados.
        for lr, reg in sorted(results):
            train_accuracy, val_accuracy = results[(lr, reg)]
            print('lr %e reg %e train accuracy: %f val accuracy: %f' % (
                        lr, reg, train_accuracy, val_accuracy))

        print('best validation accuracy achieved during cross-validation: %f' % best_val)

        print('Okay!')
```

```
lr 1.000000e-09 reg 5.000000e+04 train accuracy: 0.088429 val accuracy: 0.083000
lr 1.000000e-09 reg 5.000000e+05 train accuracy: 0.102041 val accuracy: 0.108000
lr 1.000000e-09 reg 5.000000e+06 train accuracy: 0.362306 val accuracy: 0.365000
lr 1.000000e-08 reg 5.000000e+04 train accuracy: 0.086041 val accuracy: 0.079000
lr 1.000000e-08 reg 5.000000e+05 train accuracy: 0.413122 val accuracy: 0.412000
lr 1.000000e-08 reg 5.000000e+06 train accuracy: 0.415612 val accuracy: 0.412000
lr 1.000000e-07 reg 5.000000e+04 train accuracy: 0.415388 val accuracy: 0.416000
```

```
lr 1.000000e-07 reg 5.000000e+05 train accuracy: 0.413286 val accuracy: 0.406000
lr 1.000000e-07 reg 5.000000e+06 train accuracy: 0.349102 val accuracy: 0.367000
best validation accuracy achieved during cross-validation: 0.416000
Okay!
```

In [14]:
```python
# Avaliar o melhor modelo SVM com o conjunto de teste
y_test_pred = best_svm.predict(X_test_feats)
test_accuracy = np.mean(y_test == y_test_pred)
print(test_accuracy)

print('Okay!')
```

```
0.427
Okay!
```

In [15]:
```python
# Um importante mecanismo para se ganhar intuição sobre como um modelo funciona
# é visualizar e analizar os erros que ele comete.
# Nesta visualização, serão exibidas exemplos de imagens que foram incorretamente
# classificadas pelo seu modelo.
# A primeira coluna exibe imagens que o seu modelo classificou como avião ('plane')
# mas cuja a classificação correta é outra. As outras colunas se comportam de forma
# análoga.

examples_per_class = 8
classes = ['plane', 'car', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truc
for cls, cls_name in enumerate(classes):
    idxs = np.where((y_test != cls) & (y_test_pred == cls))[0]
    idxs = np.random.choice(idxs, examples_per_class, replace=False)
    for i, idx in enumerate(idxs):
        plt.subplot(examples_per_class, len(classes), i * len(classes) + cls + 1)
        plt.imshow(X_test[idx].astype('uint8'))
        plt.axis('off')
        if i == 0:
            plt.title(cls_name)
plt.show()

print('Okay!')
```


```
Okay!
```

1.3.1 Pergunta 01:

Descreva o que você observou nos resultados incorrretamente classificados. Eles fazem algum 'sentido'?

Sua Resposta: *Fazem sentido pois as features que estão sendo levadas em consideração são os histogramas de cores. Isso pode resultar em um erro principalmente em imagens que tem o fundo com a mesma cor predominante.*

1.4 Rede Neural aplicada a Descritores de Imagem

Em uma atividade anterior, você observou que o treinamento de uma rede neural de duas camadas sobre os pixels brutos das imagens foi capaz de obter um resultado de classificação melhor que o de classificadores lineares para a mesma entrada.

Logo acima, você acaba de observar que classificadores lineares aplicados a descritores de imagens apresentam resultados superiores aos classificadores lineares aplicados aos pixels brutos.

Por completudo, agora você deve tentar treinar uma rede neural sobre os descritores de imagem.

Essa abordagem deve apresentar um resultado superior a todos as abordagens anteriores: você deve conseguir facilmente uma acurácia de classificação (taxa de acerto) maior que 55% no conjunto de teste. (OBS: minha solução obteve uma acurácia em torno de 60%!)

In [21]:
```
from dl.classifiers.neural_net import TwoLayerNet

input_dim = X_train_feats.shape[1] # Recupera o tamanho total do vetor de descritores
hidden_dim = 500
num_classes = 10

net = TwoLayerNet(input_dim, hidden_dim, num_classes)
best_net = None

################################################################################
# TODO:                                                                        #
# Treinar uma rede neural de duas camadas sobre os descritores de imagem.      #
# Você deve realizar validação cruzada de forma a obter valores para os vários #
# hiperparâmetros como feito anteriormente.                                    #
# Além disso, deve-se armazenar o melhor modelo obtido na variável best_net.   #
################################################################################
iters = 2000 #100
for lr in learning_rates:
    for rs in regularization_strengths:
        svm = LinearSVM()

        svm.train(X_train_feats, y_train, learning_rate=lr, reg=rs, num_iters=iters)

        y_train_pred = svm.predict(X_train_feats)
        acc_train = np.mean(y_train == y_train_pred)
        y_val_pred = svm.predict(X_val_feats)
        acc_val = np.mean(y_val == y_val_pred)

        results[(lr, rs)] = (acc_train, acc_val)

        if best_val < acc_val:
            best_val = acc_val
            best_net = svm
################################################################################
#                              END OF YOUR CODE                                #
################################################################################

print('Okay!')
```

```
Okay!
```

In [23]:
```
# Avaliar sua melhor rede neural sobre o conjunto de teste.
# Você deve ser capaz de obter um resultado acima de 55% de acurácia.

test_acc = (best_net.predict(X_test_feats) == y_test).mean()
print('Test accuracy: ', test_acc)

print('Okay!')
```

```
Test accuracy:  0.418
Okay!
```

2 Bonus: Projete seus próprios descritores!

Você pode observar como simples descritores de imagens podem melhorar o resultado da tarefa de classificação de imagens. Até agora, você experimentou com HOG e histograma de cor, porém outros tipos de descritores poderiam ser capazes de alcançar resultados ainda melhores.

Para obter pontos extras, você deve projetar e implementar um novo tipo de descritor e usá-lo para classification de imagens do CIFAR-10 dataset.

Explique como seu descritor funciona e por quais razões você acredita que ele será útil na tarefa de classificação de imagens.

Você deve acrescentes células neste notebook, exemplificando o cálculo de seu descritor, a validação cruzada realizada sobre os hiperparâmetros e a análise comparativa de seu desempenho com o *baseline* (representado pelo 'HOG + histograma de cor').

3 Bonus: Faça alguma coisa a mais!

Utilize o material e código apresentado, nessas 05 atividades para fazer algo interessante.

Você sugere alguma questão a mais que poderia ter sido feita?

As atividades propostas despertaram em você alguma ideia interessante enquanto trabalhava nelas?

Esta é a sua chance de se mostrar (e colaborar, é claro)!