Fatoração de Matrizes para Sistemas de Classificação de *Machine Learning*

Comp. III - CCM - 2020

September 22, 2020

Algumas instruções:

O programa pode ser feito em duplas.

Você deve implementar o programa em Python3.7

Pode usar: Matplotlib, NumPy (apenas para trabalhar com aritmética de vetores, matrizes, leitura/escrita de dados), bibliotecas básicas auxiliares: sys, time, datetime, os, math.

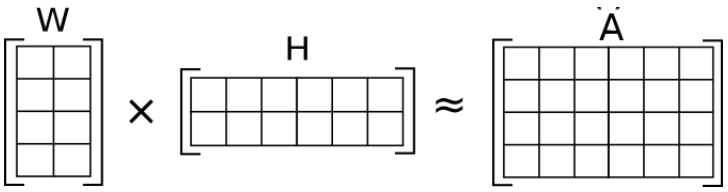
Não pode usar: SciPy, tensorflow, sklearn ou outras bibliotecas de machine learning e algebra linear computacional. Também não busque programas prontos na internet, desenvolva os códigos!

1 Introdução

Com o aumento da quantidade de dados circulando em todo o mundo, há uma crescente necessidade de se analisar dados automaticamente, sem a intervenção humana. Um dos principais desafios é o de encontrar padrões em quantidades enormes de dados, para obtermos formas de classificar e resumir os dados para que possam ser processadas por um humano.

Neste sentido, muitas ferramentas de aprendizado de máquina (*machine learning*) fazem uso de fatorações de matrizes de dados. A ideia básica deste nosso exercício é tentar fatorar um enorme conjunto de dados (uma matriz A), pela produto de duas matrizes (W e H) com certas característica que permitam uma analise simplificada dos dados e nos ajudem a agrupar e classificar os dados.

Neste exercício vamos estudar uma fatoração do tipo não-negativa, que fatora os dados, descritos em uma matriz A (tamanho $n \times m$, com m grande) por uma matriz W (tamanho $n \times p$, com p pequeno), vezes uma matriz H(tamanho $p \times m$), ambas sempre com valores não-negativos, como ilustra a figura abaixo.



O problema é que, como as dimensões das matrizes W e H são menores do que a de A, essa fatoração não necessariamente existe de forma exata! Portanto, resolvemos um problema aproximado, encontrando W e H que minimizam

$$\|A - WH\|. \tag{1}$$

O que isso tem a ver com *machine learning*?! Veremos mais adiante, primeiro vamos aprender a fazer esse tipo de fatoração, passo a passo. Usaremos um método de decomposição QR para resolver o problema de minimização, então vamos começar por essa parte.

2 Fatoração QR de matrizes e seu uso na solução de sistemas lineares

Desenvolveremos aqui o método de fatoração QR de matrizes $n \times m$, com $n \geq m$, através de rotações de Givens, descritas a seguir:

2.1 Rotações de Givens

Rotações de Givens são transformações lineares ortogonais de R^n em R^n. Recordemos inicialmente que uma transformação linear Q dada por uma matriz $n \times n$ é ortogonal se $Q^tQ = QQ^t = I$. Note que o produto de duas matrizes ortogonais também é ortogonal. Lembremos que uma transformação de R^2 em R^2 dada por

$$Q = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

representa uma rotação em torno da origem por um ângulo θ em sentido anti-horário. Uma tal transformação é claramente ortogonal. Rotações de Givens são transformações ortogonais $Q(i,j,\theta)$ de R^n em R^n tais que para $y = Q(i,j,\theta)x$ obtemos $y_k = x_k$, para $k \neq i$ e $k \neq j$, $y_i = \cos\theta x_i - \sin\theta x_j$ e $y_j = \sin\theta x_i + \cos\theta x_j$. Ou seja, tal transformação corresponde a uma rotação no plano das coordenadas i e j, deixando as outras invariantes. Ao aplicarmos uma rotação de Givens $Q(i,j,\theta)$ a uma matriz $W_{n\times m}$ apenas as linhas i e j de W são modificadas. Denotando $c = \cos\theta$ e $s = \sin\theta$ teremos após a aplicação da transformação que as linhas i e j da matriz resultante B serão dadas por

$$b_{i,k} = cw_{i,k} - sw_{j,k} \quad \text{e } b_{j,k} = sw_{i,k} + cw_{j,k} \quad , \quad k = 1, ..., m \tag{2}$$

enquanto que as linhas restantes não se alteram.

2.2 Fatoração QR

Agora iremos mostrar como transformar uma matriz $W_{n\times m}$, com $n \geq m$, em uma matriz $R_{n\times m}$, com $R_{i,j} = 0$ se $i > j$ (ou seja, R será triangular superior), através de sucessivas transformações de Givens.

Suponha que nas linhas i e j de W tenhamos as primeiras $k-1$ posições nulas e que desejemos aplicar uma transformação de Givens $Q(i,j,\theta)$ a W de forma a fazer com que $w_{j,k} = 0$ após sua aplicação. Para tanto, basta definir adequadamente o valor de θ, ou equivalentemente, de $c = \cos\theta$ e de $s = \sin\theta$. Escolhendo

$$c = \frac{w_{i,k}}{\sqrt{w_{i,k}^2 + w_{j,k}^2}} \quad \text{e} \quad s = -\frac{w_{j,k}}{\sqrt{w_{i,k}^2 + w_{j,k}^2}}$$

teremos o efeito desejado. Note que as primeiras $k-1$ posições nas linhas i e j continuarão nulas e a k-ésima entrada da linha j se torna zero após a aplicação da transformação (conforme a expressão (2)).

Observação: Outra forma, numericamente mais estável, de determinar valores de c e s é a seguinte: Se $|w_{i,k}| > |w_{j,k}|$ defina

$$\tau = -w_{j,k}/w_{i,k}, c = 1/\sqrt{1+\tau^2} \text{ e } s = c\tau \ . \tag{3}$$

Caso contrário, defina

$$\tau = -w_{i,k}/w_{j,k}, s = 1/\sqrt{1+\tau^2} \text{ e } c = s\tau. \tag{4}$$

Por exemplo, considere a matriz:

$$W = \begin{bmatrix} 2 & 1 & 1 & -1 & 1 \\ 0 & 3 & 0 & 1 & 2 \\ 0 & 0 & 2 & 2 & -1 \\ 0 & 0 & -1 & 1 & 2 \\ 0 & 0 & 0 & 3 & 1 \end{bmatrix} \ .$$

Se quisermos produzir um zero na terceira coluna da quarta linha da matriz (onde temos o valor -1) podemos utilizar uma rotação de Givens $Q(3,4,\theta)$, com os valores de $w_{3,3}$ e $w_{4,3}$ usados para se obter c e s. Obtemos

$\tau = 1/2$, $c = 2/\sqrt{5}$ e $s = 1/\sqrt{5}$. Após a aplicação da rotação de Givens correspondente, a nova matriz W é dada por:

$$W = \begin{bmatrix} 2 & 1 & 1 & -1 & 1 \\ 0 & 3 & 0 & 1 & 2 \\ 0 & 0 & 5/\sqrt{5} & 3/\sqrt{5} & -4/\sqrt{5} \\ 0 & 0 & 0 & 4/\sqrt{5} & 3/\sqrt{5} \\ 0 & 0 & 0 & 3 & 1 \end{bmatrix} .$$

Iremos definir sucessivas transformações de Givens $Q_1, Q_2, ..., Q_l$, tais que $Q_l \ ...Q_2 \ Q_1 W = R$ e portanto, como as transformações de Givens são ortogonais, teremos que $W = QR$, com Q ortogonal dada por $Q = Q_1^t \ Q_2^t ... \ Q_l^t$. Para tal, só precisamos definir uma ordem conveniente para ir zerando as entradas da matriz W abaixo da diagonal através de rotações de Givens. Há várias formas de se fazer isto. Sugerimos o seguinte algoritmo:

Para $k = 1$ até m faça

Para $j = n$ até $k + 1$ com passo -1 faça

$i = j - 1$

Se $w_{j,k} \neq 0$ aplique $Q(i, j, \theta)$ à matriz W (com c e s definidos por $w_{i,k}$ e $w_{j,k}$)

Fim do para

Fim do para

2.3 Resolvendo sistemas lineares através da Fatoração QR

Sistemas sobredeterminados

Seja W uma matriz $n \times m$ com colunas linearmente independentes e $b \in R^n$. Para solução do sistema $Wx = b$ no sentido de mínimos quadrados, ou seja, para a determinação de $x \in R^m$ que minimiza o valor de $E = ||Wx - b||$, com a norma Euclidiana do R^n, também podemos fazer uso da fatoração $W = QR$. Primeiramente observemos que como Q é ortogonal (ou seja, $Q^tQ = I$) temos que $||Q^tx||^2 = (Q^tx)^t(Q^tx) = x^tQQ^tx = x^tx = ||x||^2$. Assim, minimizar o valor de $E = ||Wx - b||$ é equivalente a minimizar $||Q^tWx - Q^tb|| = ||Rx - \tilde{b}||$, onde $\tilde{b} = Q^tb$. Como as primeiras m linhas de R formam uma matriz $m \times m$ triangular superior e não singular, está definido unicamente $x \in R^m$ que satisfaz as m primeiras equações de $Rx = \tilde{b}$. Em virtude do fato de que as últimas $n - m$ linhas de R são nulas, estas equações não serão satisfeitas se o correspondente lado direito não for nulo, independentemente do valor de x. Assim, é evidente que a solução $x \in R^m$ das m primeiras equações de $Rx = \tilde{b}$ minimiza o valor de $||Rx - \tilde{b}||$ (e consequentemente de $E = ||Wx - b||$). Mais ainda, obtemos que

$$E = ||Wx - b|| = \sqrt{\tilde{b}_{m+1}^2 + \tilde{b}_{m+2}^2 + ... + \tilde{b}_n^2} \ .$$

Note que o mesmo algoritmo pode ser utilizado no caso em que $n = m$ (sistema determinado), no qual o erro quadrático E será nulo. Observe também que para a solução do sistema por MMQ, o lado direito b passa pelas mesmas transformações que são aplicadas à matriz W. Assim, o sistema pode ser resolvido através do algoritmo:

Para k=1 a m faça

Para j=n a k+1 com passo -1 faça

i=j-1

Se $w_{j,k} \neq 0$ aplique $Q(i, j, \theta)$ à matriz W (com c e s definidos por $w_{i,k}$ e $w_{j,k}$) e ao vetor b.

Fim do para

Fim do para

Para k=m a 1 com passo -1

$x_k = (b_k - \sum_{j=k+1}^{m} w_{k,j}x_j)/w_{k,k}$

Fim do para

Este último loop consiste na solução do sistema triangular resultante. Nesta descrição a Matriz W é sempre sobrescrita e portanto ao final ela contém os valores de R.

2.4 Primeira Tarefa

Dados uma matriz W $n \times m$ e um vetor $b \in R^n$ você deve desenvolver um algoritmo que através de sucessivas rotações de Givens transforma W na matriz triangular superior R e o vetor b no vetor modificado $\tilde{b}$ e a seguir resolve o sistema $Rx = \tilde{b}$. Para a implementação destes algoritmos sugerimos que inicialmente você desenvolva um procedimento que dados dois números (correspondentes a $w_{i,k}$ e $w_{j,k}$ conforme (3) e (4)) calcule os valores de c e s a serem usados na rotação de Givens correpondente. Escreva também um procedimento para implementar a rotação de Givens, conforme o pseudo-código seguinte:

Rot-givens(W,n,m,i,j,c,s)

```
inteiros m,n,i,j,r
reais W(n,m), c, s, aux
para r=1,m
    aux = c * w(i,r) - s * w(j,r)
    w(j,r) = s * w(i,r) + c * w(j,r)
    w(i,r) = aux
fim do para
```

Note que esta rotação de Givens altera apenas as linhas i e j da Matriz W, deixando as restantes inalteradas, sendo i e j as linhas de W que foram usadas nas definições de c e s. Caso as primeiras $k-1$ colunas das linhas i e j de W sejam nulas, o loop no algoritmo poderia variar de k a m, diminuindo o número de operações. Neste caso o valor de k deveria ser também um argumento do procedimento acima. Para aplicar a rotação de Givens ao lado direito b usa-se a mesma rotina (correspondendo ao caso em que $m = 1$). Quando são resolvidos vários sistemas simultâneos, quando aplicamos a rotação ao lado direito b, o valor de m corresponde ao número de sistemas.

Estes algoritmos devem ser escritos para poderem ser utilizados na solução dos problemas de seu projeto. Como neste você terá que resolver problemas simultâneos, é conveniente que tanto b como x (as soluções dos sistemas) sejam definidos como matrizes (com a segunda dimensão representando quantos sistemas são resolvidos simultaneamente), com cada coluna de b sendo o lado direito de um dos sistemas e a coluna correspondente de x a solução deste.

Como teste inicial dos algoritmos, considere os casos (descreva os resultados no relatório):

a) $n = m = 64$, $W_{i,i} = 2, i = 1, n$, $W_{i,j} = 1,$ se $|i-j| = 1$ e $W_{i,j} = 0,$ se $|i-j| > 1$. Use $b(i) = 1, i = 1, n$.
b) $n = 20, m = 17$, $W_{i,j} = 1/(i+j-1),$ se $|i-j| \leq 4$ e $W_{i,j} = 0,$ se $|i-j| > 4$. Use $b(i) = i, i = 1, n$.

Vários sistemas simultâneos

Na aplicação a ser resolvida nesta tarefa, queremos minimizar $||A - WH||$ onde A é uma matriz $n \times m$, W é $n \times p$ e H é $p \times m$, com a norma acima definida como a norma de Frobenius: $||A|| = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{m} a_{i,j}^2}$. Para tanto, com W e A dadas, queremos determinar a matriz H. Note que o produto de W por uma coluna h_j de H deve aproximar a coluna a_j de A. Em outras palavras, queremos minimizar $||A-WH||^2 = \sum_{j=1}^{m} ||Wh_j - a_j||^2$, o que corresponde à resolução de m sistemas sobredeterminados (todos com a mesma matriz W) simultaneamente, um para cada coluna da matriz A (sendo a solução a coluna correspondente da matriz H). Para tanto podemos adaptar o algoritmo para os sistemas sobredeterminados:

Para k=1 a p faça

Para j=n a k+1 com passo -1 faça

i=j-1

Se $w_{j,k} \neq 0$ aplique $Q(i,j,\theta)$ à matriz W (com c e s definidos por $w_{i,k}$ e $w_{j,k}$) e à matriz A.

Fim do para

Fim do para

Para k=p a 1 com passo -1

Para j=1 a m faça

$h_{k,j} = (a_{k,j} - \sum_{i=k+1}^{p} w_{k,i}h_{i,j})/w_{k,k}$

Fim do para

Fim do para

Como teste inicial dos algoritmos, considere os casos (descreva os resultados no relatório):

c) $n = p = 64$, $W_{i,i} = 2, i = 1, n$, $W_{i,j} = 1,$ se $|i-j| = 1$ e $W_{i,j} = 0,$ se $|i-j| > 1$. Defina $m = 3$, resolvendo 3 sistemas simultâneos, com $A(i,1) = 1,\ A(i,2) = i,\ A(i,3) = 2i-1, i = 1, n$.

d) $n = 20, p = 17$, $W_{i,j} = 1/(i+j-1),$ se $|i-j| \leq 4$ e $W_{i,j} = 0,$ se $|i-j| > 4$. Defina $m = 3$, resolvendo 3 sistemas simultâneos, com $A(i,1) = 1,\ A(i,2) = i,\ A(i,3) = 2i-1, i = 1, n$.

3 Fatoração por matrizes não negativas

Seja A uma matriz $n \times m$, cujas entradas são todas maiores ou iguais a zero (ou seja, A é uma matriz não negativa). Dado um número inteiro positivo p menor que m e n, gostaríamos de fatorar a matriz A como produto de duas matrizes: uma matriz W $n \times p$ e uma matriz H $p \times m$, ambas também não negativas. Normalmente, uma tal fatoração não existe. Procura-se então uma fatoração desse tipo (conhecida na literatura como "non-negative matrix factorization" - NMF) que minimize o valor do erro quadrático

$$E = ||A - WH||^2 = \sum_{i=1}^{n}\sum_{j=1}^{m}(A_{i,j} - (WH)_{i,j})^2 \quad .$$

Este problema não é muito simples, uma vez que o erro pode possuir vários mínimos locais.

Há várias abordagens (algumas heurísticas) para determinação de pontos de mínimo local do erro. Nesta tarefa iremos considerar o método de mínimos quadrados alternados (alternating Least Squares).

Este é executado através do seguinte algoritmo (com a matriz A não negativa $n \times m$ e o valor de p dados):

Inicialize randomicamente a matriz W com valores positivos

Armazene uma cópia da matriz A

Repita os seguintes passos até que a norma do erro se estabilize (diferença entre as normas do erro em dois passos consecutivos $< \epsilon$ (use $\epsilon = 10^{-5}$ no seu programa) ou que um número máximo de iterações ($itmax$, escolha $itmax = 100$) seja atingido

Normalize W tal que a norma de cada uma de suas colunas seja 1 ($w_{i,j} = w_{i,j}/s_j$, com $s_j = \sqrt{\sum_{i=1}^{n} w_{i,j}^2}$)

Resolva o problema de mínimos quadrados $WH = A$, determinando a matriz H (são m sistemas simultâneos! Cuidado, pois A é modificada no processo de solução. Na iteração seguinte deve-se usar a matriz A original novamente. Por isso armazena-se uma cópia de A!)

Redefina H, com $h_{i,j} = \max\{0, h_{i,j}\}$

Compute a matriz A^t (transposta da matriz A original)

Resolva o problema de mínimos quadrados $H^tW^t = A^t$, determinando a nova matriz W^t. (são n sistemas simultâneos!)

Compute a matriz W (transposta de W^t)

Redefina W, com $w_{i,j} = \max\{0, w_{i,j}\}$

Fim do Repita

Segunda Tarefa

Implemente o algoritmo de mínimos quadrados alternados para obtenção de fatorações não negativas.

Considere a seguinte matrix A definida abaixo. Realize a fatoração WH desta matriz considerando $p = 2$, isto é, W com tamanho 3×2 e H com tamanho 2×3. Verifique se o resultado esta de acordo com a multiplicação $A = WH$ em relação aos dados fornecidos abaixo (note que neste caso a decomposição é exata).

$$A = \begin{pmatrix} 3/10 & 3/5 & 0 \\ 1/2 & 0 & 1 \\ 4/10 & 4/5 & 0 \end{pmatrix}$$

$$W = \begin{pmatrix} 3/5 & 0 \\ 0 & 1 \\ 4/5 & 0 \end{pmatrix}$$

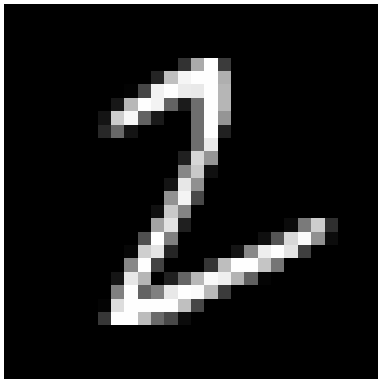
$$H = \begin{pmatrix} 1/2 & 1 & 0 \\ 1/2 & 0 & 1 \end{pmatrix}$$

Comente os resultados no relatório. Você pode criar outros exemplos de matrizes a partir de fatorações de sua escolha (ou seja, matrizes A que sejam produtos de W e H escolhidas) e testar seu algoritmo.

4 Classificação de dígitos manuscritos

Vamos agora trabalhar com uma aplicação realística, de maior porte, que pode ser resolvida de forma razoavelmente eficaz pelo método estudado: a classificação de dígitos manuscritos.

Considere que temos uma imagem com um dígito manuscrito, como a ilustrada abaixo, obtida da base de dígitos manuscrito MNIST (`http://yann.lecun.com/exdb/mnist/`).



Nosso objetivo é "ensinar uma máquina" a descobrir que número aparece na imagem. Veja em `en.wikipedia.org/wiki/MNIST_database` alguns outros métodos de classificação. Aprendizagem de máquina é uma área de pesquisa atual, com grande potencial de aplicações. Nós vamos usar um método diferente desses para classificar os dígitos, baseado na fatoração não negativa de matrizes.

Incialmente vamos descrever um método para avaliar se o que aparece na imagem deve ou não ser um certo dígito d. Depois descreveremos como identificar qualquer dígito. Para tanto vamos precisar primeiro "treinar" nossa máquina, com uma base de dados conhecida, e depois podemos usar nossa máquina para tentar classificar dígitos não conhecidos (etapa de teste).

4.1 Treinamento de um dígito d

Primeiro usamos uma base de imagens que contenha apenas o dígito de interesse, d, para treinar nossa máquina de classificação. Vamos considerar apenas o caso de imagens em preto e branco (escala de cinza).

Pré-processamento das imagens

Cada imagem é uma matrix $n_x \times n_y$ (no caso acima, 28×28) com valores entre 0 e 255 (escala RGB), que representam escalas de cinza, sendo 0 o preto e o 255 o branco.

Há algumas etapas importantes de pré-processamento das imagens para que estas possam ser representadas por uma única matriz A, que será usada no método de aprendizagem:

Leitura de m imagens originais, cada uma com tamanho $n_x \times n_y$ de uma base de dados fornecida.

Cada imagem deve ser redimensionada para se tornar um vetor coluna com $n = n_x n_y$ linhas. No exemplo dado, cada imagem é convertida em um vetor coluna de tamanho $n = 28^2 = 784$. Essa transformação pode ser feita simplesmente "empilhando" as colunas da matrix que define a imagem.

Juntam-se os m vetores colunas gerados para formar uma matrix A de tamanho $n \times m$, onde cada coluna representa uma imagem.

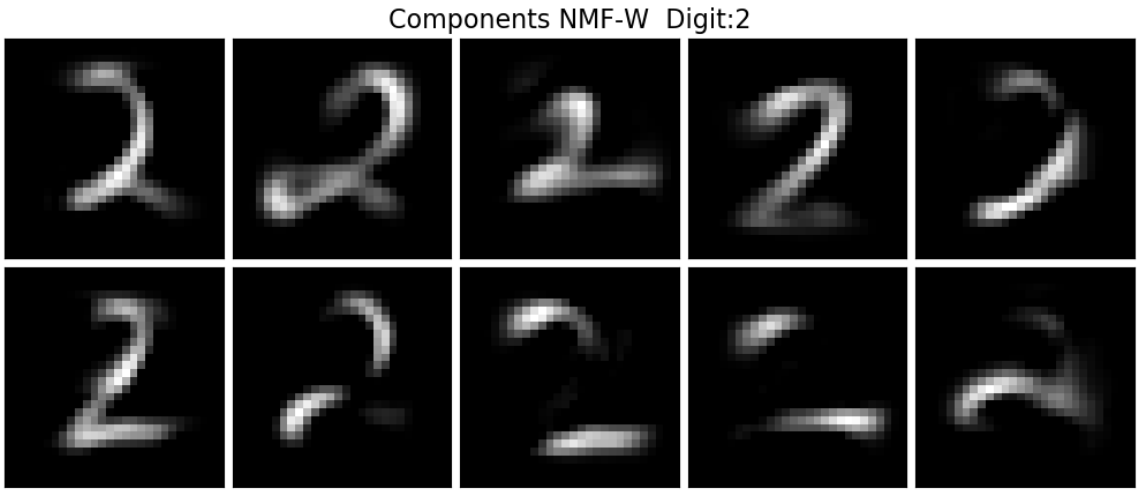
Por fim, é conveniente normalizarmos os valores da matriz forçando com que fiquem entre 0 e 1. O mais simples neste caso é simplesmente dividir cada elemento da matriz por 255, que é o maior valor da escala RGB.

A normalização indicada é importante pois vamos decompor a matrix em WH com W normalizada, portanto, para que possamos interpretar W, a matriz A deve ter a mesma ordem de grandeza. Outras normalizações são possíveis, mas neste exercício trabalharemos apenas com essa mais simples.

4.1.1 Aprendizagem

A partir da matriz A, que contém informações referentes a um único dígito, podemos realizar a fatoração não negativa desta matriz em p componentes.

Por exemplo, usando a base de imagens MNIST com 5958 imagens manuscritas do número 2, encontramos $p = 10$ componentes W que estão representadas na figura abaixo. Cada imagem abaixo representa uma coluna da matriz W. Estas foram novamente escritas na forma 28×28 e reescaladas, multiplicando por 255, para a visualização (sempre que você quiser visualizar uma imagem será necessário este procedimento). Cada uma representa um "perfil típico" de um 2.



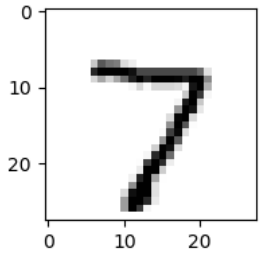
A matrix W representa o que foi aprendido sobre o dígito manuscrito 2, a partir dos dados disponíveis de aprendizagem e será usada na classificação de dígitos.

4.2 Classificador de dígitos

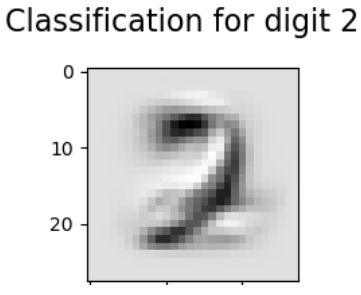
Uma vez que tenhamos treinado o classificador do dígito d (a matriz W_d), podemos usá-lo para avaliar se um dígito é de fato d ou não.

Consideremos uma nova imagem, já escrita na forma de vetor coluna e normalizada, que denotaremos por v. Vamos aproximar esse vetor (essa imagem) por uma combinação linear das colunas da matriz W_d. Para tanto usamos o método de mínimos quadrados, resolvendo o sistema sobredeterminado $W_d x = v$. Em função do erro quadrático nesta aproximação iremos avaliar se esta nova imagem deve ser classificada como um dígito d.

Por exemplo, consideremos a imagem seguinte, que percebemos ser claramente um dígito 7 (este é o nosso v. **Obs:** Nesta e nas próximas figuras, o código RGB está invertido).



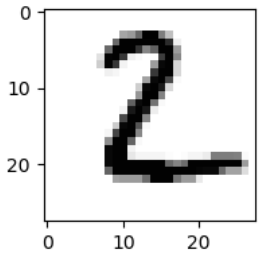
Iremos testar se ela é um dígito 2, usando o classificador W_2 calculado na fase de aprendizagem. Resolvendo o sistema $W_2 x = v$, obtemos a sua aproximação nas componentes do dígito 2, isto é $W_2 x$, como ilustrado abaixo.



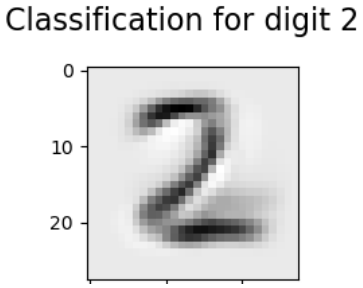
Há pouca semelhança, traduzida num erro quadrático significativo.

Agora vamos tentar classificar se a seguinte figura, que não fazia parte de nosso conjunto de treinamento, é

um número 2.



Ao fazemos a mesma classificação a partir desse nova imagem obtemos a seguinte reconstrução W_2x, que leva a um erro quadrático bem menor.



A seguir iremos ver como explorar os classificadores, na tentativa de classificar dígitos quaisquer. Dada a imagem de um dígito, e um conjunto de classificadores W_d, $d = 0, 1, 2, ..., 9$ previamente calculados numa fase de aprendizagem, iremos determinar qual deles leva ao menor erro quadrático, desta forma obtendo uma indicação de qual dígito nossa imagem representa. Os detalhes sobre como fazer isso estão na próxima e mais interessante tarefa deste projeto.

4.3 Tarefa Principal: Classificação de dígitos de uma base de dados reais

Iremos trabalhar com dados extraídos da base de dados MNIST (`http://yann.lecun.com/exdb/mnist/`, também disponível em PNG `https://github.com/myleott/mnist_png`). Já adiantamos parte do trabalho, fazendo o pré-processamento dos dados que serão utilizados em sua tarefa. Em `https://www.ime.usp.br/~pedrosp/dados_mnist.zip` você encontra: dez arquivos train_digX.txt contendo dados para a fase de aprendizagem. Cada um destes arquivos contém os dados referentes a imagens dos dígitos correspondentes para a fase de treinamento. Os arquivos têm 784 linhas, com pelo menos 5300 entradas por linha. Isto significa que há dados suficientes para executar o treinamento com até 5300 amostras de cada dígito. Além destes arquivos são fornecidos arquivos para a fase de testes dos classificadores que serão desenvolvidos na fase de treinamento. O arquivo test_images.txt tem 784 linhas e 10000 entradas por linha, correspondendo a 10000 imagens distintas de dígitos. Através dos dados deste arquivo iremos classificar o índice de acerto dos classificadores que serão desenvolvidos. Um arquivo extra, test_index.txt, traz em suas 10000 linhas a informação sobre os dígitos que estão representados em test_images.txt. Através desta informação você será capaz de calcular os índices de acerto na classificação de cada dígito.

A sua tarefa é a seguinte:

1. Treinar os classificadores calculando as matrizes W_d para cada dígito $d = 0, 1, ..., 9$, usando para tal um número determinado de digitos para treinamento *ndig_treino*, a serem lidos dos arquivos correspondentes train_digX.txt, com X variando de 0 a 9.

   Para cada dígito d é lida do arquivo a matriz A com $n = 784$ linhas e o número de colunas m especificado pelo número *ndig_treino*.

   Cada classificador deve ser armazenado em uma matriz W_d de tamanho $n \times p$ (p é o número de componentes desejado, por exemplo 10). A matriz W_d é calculada pela decomposição da matriz A correspondente ao dígito d num produto W_dH, como descrito anteriormente. As matrizes W_d devem ser todas armazenadas, enquanto que a matriz H correspondente pode ser descartada.

2. Após a fase de treinamento com *ndig_treino* deve-se usar o conjunto de testes para se aferir o porcentual de acerto dos classificadores.

   As imagens que serão usadas para os testes devem ser lidas do arquivo test_images.txt e armazenadas em uma matriz A com 784 linhas e n_test colunas, sendo 10000 o número de imagens neste arquivo, e portanto o máximo valor possível para o número de imagens n_test que se deseja testar.

   Para cada dígito devemos resolver o sistema $W_d H = A$, no sentido de mínimos quadrados, onde H é uma matriz $p \times n\_test$ (através do método de fatoração QR da matriz W_d correspondente - veja a seção sobre sistemas simultâneos).

   Calculamos então, para cada coluna c_j de $A - W_d H$ sua norma euclidiana ($||c_j|| = \sqrt{\sum_{i=1}^{784} c_{i,j}^2}$). Este valor nos permitirá avaliar se a j-ésima imagem da matriz de testes deve ou não ser um dígito d. Classificaremos a j-ésima imagem como um dígito d se o valor de erro correspondente $e_j = ||c_j||$ for menor que o valor de erro obtido para os outros dígitos.

   Para tanto defina dois vetores de tamanho n_test. Em um deles você armazenará qual o dígito mais provável correspondente à imagem j (dentre os dígitos já testados) e no outro o valor do erro e_j correspondente. A cada novo dígito testado, obtemos os erros correspondentes a cada imagem. Se o novo valor de e_j for menor que o anterior, armazenamos este novo valor de erro - mínimo até então - e o novo dígito como o mais provável. Se o erro novo for maior que o erro armazenado, concluímos que a imagem j não corresponde a este novo dígito.

   Após percorrermos o procedimento para todos os dígitos de 0 a 9, teremos armazenada uma classificação para cada uma das n_test imagens teste.

3. No arquivo test_index.txt temos o verdadeiro dígito de cada imagem do arquivo test_images.txt. Como classificamos as n_test imagens como determinados dígitos, estamos agora em condição de avaliar a qualidade da classificação, em termo do número de acertos. Calcule os seguintes índices:

   a) Porcentual total de acertos.

   b) Para cada dígito, avalie quantas classificações foram corretas dentre o total de ocorrências deste dígito no arquivo de testes e o correspondente porcentual de acerto para este dígito.

4. Execute estes testes variando

   $ndig\_treino = 100, 1000, 4000$

   $n\_test = 10000$

   $p = 5, 10, 15$

5. Discuta os resultados no seu relatório, descrevendo exemplos, taxas de erros (em função do tamanho dos conjuntos de treinamento) e suas impressões sobre a tarefa. A implementação do método de Householder como alternativa às rotações de Givens (e a comparação dos dois métodos) será valorizada.

6. Seu programa deve imprimir os tempos computacionais (CPU time) para cada caso.

Divirta-se!