Notizen zur Programmieraufgabe Hauptkomponentenanalyse

Johannes Hertrich

In dieser Übungsaufgabe, betrachten wir den MNIST Datensatz. Dieser besteht aus 60000 Grauwertbildern der Größe 28×28. Jedes dieser Bilder zeigt eine handgeschriebene Ziffer von 0 bis 9. Außerdem besitzt jedes der Bilder ein Label, das angibt, welche Ziffer auf dem Bild zu sehen ist. Im Folgenden möchten wir nun Bilder maschienell klassfizieren. Das heißt, wir möchten auf Basis des Datensatzes einen Algorithmus entwickeln, der ein Bild der Größe 28×28 als Eingabe nimmt und daraufhin vorhersagt, welche Ziffer darauf zu sehen ist.

Der MNIST-Datensatz ist in 50000 „Trainings-" und 10000 „Testdaten" unterteilt. Die Idee dabei ist, die Trainingsdaten (Bilder und Labels) zu verwenden, um unseren Algorithmus aufzustellen. Daraufhin verwenden wir die Testdaten um zu überprüfen, wie gut unsere Methode funktioniert. Das bedeutet, wir sagen für jedes Bild aus dem Testdatensatz vorher, welche Ziffer dieses zeigt, ohne das Label zu verwenden. Abschließend gleichen wir mit Hilfe der Labels ab, bei welchem Anteil der Bilder wir richtig lagen.

Der Algorithmus, den wir in dieser Programmieraufgabe betrachten, besteht aus zwei Schritten. Da der Raum $\mathbb{R}^{28\times 28}$ der Bilder Dimension $28^2 = 784$ hat, sind Rechenoperationen darin oft sehr rechenaufwändig. Deshalb reduzieren wir in einem ersten Schritt zunächst die Dimension des Problems mit der sogenannten Hauptkomponentenanalyse. Danach führen wir die eigentliche Klassifizierung der Bilder mit Hilfe des K-Means Algorithmus durch.

Bemerkung 1 (Einlesen des MNIST-Datensatzes in Python)**.** Die Bilder des Trainingsdatensatzes können folgendermaßen in Python eingelesen werden.

- `imgs = np.fromfile('train-images.idx3-ubyte', dtype=np.uint8)`
- `imgs = np.reshape(imgs[16:], [-1, 28, 28])`

Die zugehörigen Labels kann man mit

- `labs = np.fromfile('train-labels.idx3-ubyte', dtype=np.uint8)`
- `labs = labs[8:]`

einlesen. Der Trainingsdatensatz kann eingelesen werden indem im Dateinamen `train` durch `test` ersetzt wird.

1 Hauptkomponentenanalyse

Wir beschäftigen uns zunächst mit der Dimensionsreduktion. Dafür nehmen wir an, dass wir N Punkte $x_0, ..., x_{N-1} \in \mathbb{R}^D$ für großes D gegeben haben. Nun suchen wir einen affinen Teilraum von deutlich kleinerer Dimension $d \ll D$ sodass die Summe der quadratischen Abstände der Punkte x_n zu diesem Unterraum möglichst klein ist. Diese Begriffen möchten wir zunächst formal definieren.

Definition 2. Ein *affiner Teilraum* des $\mathbb{R}^D$ der Dimension d ist eine Menge

$$H_{A,b} \coloneqq \{At + b : t \in \mathbb{R}^d\},$$

wobei $A \in \mathbb{R}^{D\times d}$ eine Matrix mit orthonormalen Spalten (d.h. $A^{\mathrm{T}}A = I_d$) und $b \in \mathbb{R}^D$ beliebig ist.

Dementsprechend ist $H_{A,0}$ der Untervektorraum von $\mathbb{R}^D$, der durch die Orthogonalbasis aufgespannt wird, die durch die Spalten $(a_0, ..., a_{d-1})$ von A gegeben ist. Außerdem ist $H_{A,b}$ eine um b verschobene Version von $H_{A,0}$. Wir definieren den quadratischen Abstand von einem Punkt x zu einer Menge M durch

$$d^2(x, M) \coloneqq \inf_{y\in M} \|x - y\|^2.$$

Falls, M ein affiner Teilraum des $\mathbb{R}^D$ ist, können wir zeigen, dass das Infimum immer angenommen wird und der Minimierer eindeutig ist.

Lemma 3. *Sei $x \in \mathbb{R}^D$ und sei $H_{A,b}$ ein affiner Teilraum des $\mathbb{R}^D$. Dann ist*

$$d^2(x, H_{A,b}) = \|\mathrm{proj}_{H_{A,b}}(x)-x\|^2, \quad \textit{wobei} \quad \mathrm{proj}_{H_{A,b}}(x) \coloneqq \operatorname*{arg\,min}_{y\in H_{A,b}} \|x-y\|^2 = AA^{\mathrm{T}}(x-b)+b.$$

Wir nennen $\mathrm{proj}_{H_{A,b}}(x)$ die orthogonale Projektion von x auf $H_{A,b}$.

Beweis. Sei $y = At + b \in H_{A,b}$ beliebig. Dann gilt $\|x - y\|^2 = \|y - x\|^2 = \|At + b - y\|^2$. Um einen Minimierer zu finden, setzen wir die Ableitung nach t zu 0. Das ergibt

$$0 = A^{\mathrm{T}}(A\hat{t} + b - x), \quad \Leftrightarrow \quad \underbrace{A^{\mathrm{T}}A}_{I_d}\hat{t} = A^{\mathrm{T}}(x - b).$$

Dass $\hat{t}$ wirklich ein Minimierer ist und dass dieser eindeutig ist, ist eine (freiwillige) Übungsaufgabe. Dementsprechend ist

$$\mathrm{proj}_{H_{A,d}}(x) = \operatorname*{arg\,min}_{y\in H_{A,d}} \|x - y\|^2 = A\hat{t} + b = AA^{\mathrm{T}}(x - b) + b$$

und der Beweis ist vollendet. □

Nun können wir unsere eigentliche Problemstellung formulieren.

Problem 4 (Hauptkomponentenanalyse)**.** Für gegebene Punkte $x_0, ..., x_{N-1} \in \mathbb{R}^D$ suchen wir einen affinen Teilraum $H_{A,b}$ sodass

$$(\hat{A}, \hat{b}) \in \operatorname*{arg\,min}_{A \in \mathbb{R}^{D\times d}, b\in\mathbb{R}^D} \sum_{i=0}^{N-1} \|\mathrm{proj}_{H_{A,b}}(x_i) - x_i\|^2. \tag{1}$$

Wir nennen das Problem bzw. eine Lösung $(\hat{A}, \hat{b})$, die Hauptkomponentenanalyse (engl. „Principal Component Analysis", PCA) von $x_0, ..., x_{N-1}$.

Wir lösen das Problem in der folgenden Proposition.

Proposition 5. *Ein optimalum $(\hat{A}, \hat{b})$ aus (1) ist gegeben durch den Mittelwert $\hat{b} = \frac{1}{N}\sum_{i=0}^{N-1} x_i$ und die Matrix $\hat{A} = (\hat{a}_0, ..., \hat{a}_{d-1})$, wobei die $\hat{a}_i$ die Eigenvektoren zu den d größten Eigenwerten der Matrix*

$$S = YY^{\mathrm{T}} = \sum_{i=1}^{N}(x_i - b)(x_i - b)^{\mathrm{T}}, \quad \textit{mit} \quad Y = (x_0 - b|...|x_{N-1} - b).$$

Die Matrix S heißt *empirische Kovarianzmatrix* von den Punkten x_i. Also umfasst die Matrix $\hat{A}$ die d Richtungen mit größter Varianz.

Beweis. Die Zielfunktion in (1) ist gegeben durch

$$\sum_{i=0}^{N-1} \|\mathrm{proj}_{H_{A,b}}(x_i) - x_i\|^2 = \sum_{i=0}^{N-1} \|AA^{\mathrm{T}}(x_i - b) + b - x_i\|^2 = \sum_{i=0}^{N-1} \|(I_D - AA^{\mathrm{T}})(x_i - b)\|^2.$$

Wir setzen die Ableitung auf null um kritische Punkte auszurechnen. Dass es sich bei diesen um globale Minima handelt, verbleibt wieder als (freiwillige) Übungsaufgabe. Das ergibt

$$0 = \sum_{i=1}^{N}(I_D - AA^{\mathrm{T}})^{\mathrm{T}}(I_D - AA^{\mathrm{T}})(x_i - \hat{b}) = -N(I_D - AA^{\mathrm{T}})\hat{b} + \sum_{i=1}^{N}(I_D - AA^{\mathrm{T}})x_i.$$

Durch Umstellen wird das zu

$$(I_D - AA^{\mathrm{T}})\hat{b} = (I_D - AA^{\mathrm{T}})\Big(\frac{1}{N}\sum_{i=1}^{N} x_i\Big).$$

Unabhängig von A ist diese Gleichung ist offenbar durch $\hat{b} = \frac{1}{N}\sum_{i=1}^{N} x_i$ erfüllt und wir erhalten die Aussage für $\hat{b}$.

Für die Minimierung nach A für festes $b = \hat{b}$, schreiben wir $y_i = x_i - b$ und $Y = (y_0|...|y_{N-1}) \in \mathbb{R}^{D\times N}$. Dann können wir die Zielfunktion umschreiben als

$$\hat{A} \in \operatorname*{arg\,min}_{A \in \mathbb{R}^{D\times d}, A^{\mathrm{T}}A = I_d} \sum_{i=0}^{N-1} \|(I_D - AA^{\mathrm{T}})y_i\|^2$$

Nun gilt

$$\begin{aligned}\|(I_D - AA^{\mathrm{T}})y\|^2 &= y^{\mathrm{T}}(I_D - AA^{\mathrm{T}})^{\mathrm{T}}(I_D - AA^{\mathrm{T}})y = y^{\mathrm{T}}(I_D - AA^{\mathrm{T}})y \\ &= y^{\mathrm{T}}y - y^{\mathrm{T}}AA^{\mathrm{T}}y = y^{\mathrm{T}}y - \sum_{j=0}^{d-1} y^{\mathrm{T}}a_j a_j^{\mathrm{T}}y = \|y\|^2 - \sum_{i=0}^{d-1} a_j^{\mathrm{T}}yy^{\mathrm{T}}a_j\end{aligned}$$

Damit erhalten wir

$$\begin{aligned}\hat{A} &\in \operatorname*{arg\,min}_{A\in\mathbb{R}^{D\times d}, A^{\mathrm{T}}A=I_d} \sum_{j=1}^{d}\sum_{i=0}^{N-1} \underbrace{\|y_i\|^2}_{\text{konstant}} - a_j^{\mathrm{T}}y_i y_i^{\mathrm{T}}a_j \\ &= \operatorname*{arg\,min}_{A\in\mathbb{R}^{D\times d}, A^{\mathrm{T}}A=I_d} -\sum_{j=1}^{d} -a_j^{\mathrm{T}}YY^{\mathrm{T}}a_j,\end{aligned}$$

wobei $Y = (y_0|...|y_{N-1})$. □

Numerisch ist es stabiler, die Eigenwerte von S über die Singulärwertzerlegung von Y mit Hilfe des folgenden Lemmas zu berechnen. Die Singulärwertzerlegung ist in Numpy bereits als `numpy.linalg.svd` implementiert.

Lemma 6. *Sei $S \in \mathbb{R}^{D\times D}$ gegeben durch $S = YY^{\mathrm{T}}$ für ein $Y \in \mathbb{R}^{D\times N}$ mit Singulärwertzerlegung $U\Sigma V^{\mathrm{T}}$. Dann ist die Eigenwertzerlegung von S gegeben durch*

$$S = U\Sigma^2 U^{\mathrm{T}}.$$

Das heißt, die Eigenvektoren von S sind die linken Singulärvektoren von Y.

Beweis. Es gilt

$$S = YY^{\mathrm{T}} = U\Sigma V^{\mathrm{T}}(U\Sigma V^{\mathrm{T}}) = U\Sigma V^{\mathrm{T}}V\Sigma^{\mathrm{T}}U^{\mathrm{T}}.$$

Mit der Orthogonalität von V (d.h. $V^{\mathrm{T}}V$) und Diagonalität von Σ (sodass $\Sigma = \Sigma^{\mathrm{T}}$) erhalten wir die Aussage. □

Sobald $\hat{A}$ und $\hat{b}$ nun berechnet sind, können wir die Dimension des Problems reduzieren, indem wir alle Punkte $x \in \mathbb{R}^d$ auf $H_{\hat{A},\hat{b}}$ projezieren und nur noch mit den Koeffizienten $t \in \mathbb{R}^d$ aus der Darstellung $H_{\hat{A},\hat{b}} = \{\hat{A}t + \hat{b} : t \in \mathbb{R}^d\}$ rechnen. Für $x \in \mathbb{R}^D$ kann das zugehörige $t \in \mathbb{R}^d$ durch $t = \hat{A}^{\mathrm{T}}(x - \hat{b})$ bestimmt werden. Andersherum erhalten wir x aus t durch $x = \hat{A}t + \hat{b}$.

2 Der K-Means Algorithmus

Als nächstes betrachten wir einen einfachen Algorithmus zum Klassifizieren. Dafür nehmen wir an, dass Punkte $x_0, ..., x_{N-1} \in \mathbb{R}^d$ gegeben sind. Wir möchten diese Daten in K Klassen einteilen. Diese werden jeweils durch ein Zentrum $r_k \in \mathbb{R}^d$ für $k = 0, ..., K-1$

repräsentiert Die Hauptidee ist, dass ein Punkt x_i immer dem nächsten Zentrum zugeordnet wird. Gleichzeitig soll jedes Zentrum der Mittelwert der zugehörigen Punkte sein.

Um die Zentren $r_0, ..., r_{K-1}$ zu finden, verwenden wir einen iterativen Algorithmus, der abwechelnd

- alle Punkte $x_0, ..., x_{N-1}$ dem nächstgelegenen Zentrum zuordnet,
- die Zentren r_k auf den Mittelwert aller ihm zugeordneter Punkte setzt.

Damit erhalten wir den folgenden Algorithmus.

Algorithm 1 K-Means Algorithmus

Eingabe: Punkte $x_0, ..., x_{N-1} \in \mathbb{R}^d$
Initialisierung: Zentren $r_1, ..., r_{K-1}$
while Abbruchkriterium nicht erfüllt **do**
  Setze $C_k = \emptyset$ für $k = 0, ..., K-1$.
  for $i = 0, ..., N-1$ **do**
    Berechne

$$\hat{k} = \underset{k=0,...,K-1}{\arg\min} \|x_i - r_k\|^2$$

    Setze $C_{\hat{k}}$ auf $C_{\hat{k}} \cup \{i\}$.
  end for
  for $k = 0, ..., K-1$ **do**
    Setze

$$r_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i$$

  end for
end while

Als Initialisierung kann für r_k ein zufälliger Punkt x_i gewählt werden. Ein typisches Abbruchkriterium ist, dass sich die Zentren r_k nicht mehr verändern. Um eine zu lange Laufzeit zu verhindern, sollte dabei immer eine Maximalanzahl an Iterationen implementiert werden.

Bemerkung 7. Für den K-Means Algorithmus benötigen wir die Labels der Trainingsbilder nicht. Diese Voraussetzung wird auch „unüberwachtes Lernen" (engl. „unsupervised learning") genannt. Algorithmen, die die Labels der Trainingsbilder verwenden werden hingegen „überwachtes Lernen" (engl. „supervised learning") genannt.