Vektorquantisierung

von

Wolfgang Wiese
http://www.uni-erlangen.de/~unrzc9
unrzc9@rrze.uni-erlangen.de

im

Proseminar
„Grundlagen der automatischen Bild- und Sprachverarbeitung“
WS 1996/97

Übersicht:

1. Einleitung

Teil der automatischen Bild- und Sprachverarbeitung ist es, die aufgenommenen (eingehenden) Signale zu klassifizieren (z.B. den jeweils „richtigen" Symbol- oder

Lautgruppen zuzuordnen) und somit einer effizienten Weiterverarbeitung zugänglich zu machen.

Die Vektorquantisierung (VQ) ist ein Weg diese Aufgabe zu lösen.
Die VQ spielt bei einem Bild- und Sprachverarbeitungssystem die Rolle eines Moduls, welches eine Menge von Merkmalsvektoren als Eingabe erhält und für jeden dieser Vektoren einen Indexwert als Ergebnis liefert.

2. Der Vektorquantisierer

2.1. Begriffe und Grundlagen

Der Vektorquantisierer wird durch ein Codebuch charakterisiert.
Das Codebuch besteht aus Prototypvektoren, welche durch einen Index nummeriert sind.
Die Prototypvektoren bilden Repräsentanten für die Merkmalsvektoren, die durch ein Abstandsmaß auf diese Vektoren abgebildet werden.
Ziel in der VQ ist es, eine möglichst gute Repräsentation der Merkmalsvektoren zu erreichen.

Ein Vektorquantisierer ist also ein Operator, der jedem einkommenden Merkmalsvektor einem Codebuch-Vektor zuweist und den Indexwert des Codebuch-Vektors zurückgibt.
Dabei wird der ganze Merkmalsraum, ein D-dimensionaler reeler Raum, auf eine begrenzte Menge von Codebuch-Vektoren, den Prototypvektoren, abgebildet:

$$q: \begin{cases} \Re^D \rightarrow Z=\{Z_1, Z_2, \ldots, Z_k\} \\ x \rightarrow q(x) \end{cases}$$

Der Merkmalsraum $\Re^D$ wird somit durch das Codebuch in k Partitionen von Z aufgeteilt. Diese Aufteilung bewirkt eine effiziente Darstellung der Eingangsinformationen durch Indexwerten.

Ein Beispiel macht dies besonders deutlich: Ein Sprachsignal, welches mit einer Bandbreite von 10 kHz und mit 16 Bit aufgenommen wurde, also 160 kbps, kann durch die VQ reduziert werden auf eine Bitrate von 1000 bps!

2.2. Elemente eines Vektorquantisierers

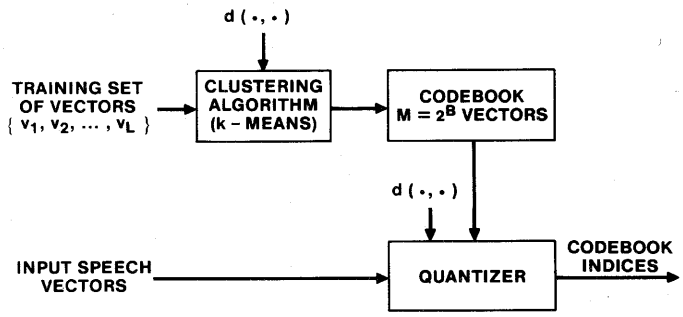

Figure 3.40 Block diagram of the basic VQ training and classification structure.

Bild 1 zeigt den prinzipiellen Aufbau eines VQs-Systems. Man erkennt, daß ein Vektorquantisierer im wesentlichen aus 3 Teilen besteht:

I. **Ein Satz an Traingsvektoren**. Diese Traingsvektoren werden benötigt, um einen „optimalen" Satz an Codebuch-Vektoren zu erstellen. Experimente brachten die Erkenntnis, daß bei einer Codebuch-Größe von M (zum Beispiel: $M = 2^B$) Vektoren eine Anzahl von L = 10 * M Trainingsvektoren nötig sind um ein gut funktionierendes, d.h. ein robustes VQs-System zu erhalten.
II. **Ein Abstandsmaß.** Ein Maß, das festlegt, welcher Merkmalsvektor zu welchem Prototypvektor zuzuordnen ist. In der Regel wird hier die Minimumsabstandsregel, die schon beim NN eingeführt wurde benutzt.
III. **Ein Algorithmus zur Codebuchgenerierung**. Eine Prozedur, mit dessen Hilfe aus Trainingsvektoren und dem vorhandenen Codebuch ein verbessertes Codebuch erstellt werden kann.

I. Die Trainingsvektoren.

Mit den Traingsvektoren werden die Codebuch-Vektoren festgelegt. Dementsprechend sollten sie ein großes Gebiet an Variationen aufspannen.
Sie sind charakterisiert durch

- dem Sprecher: Alter, Geschlecht, Akzent, Sprechgeschwindigkeit, Tonlagen und weitere sprecherabhängige Variablen.
- den Sprechbedingungen: Unterschiede zwischen Raumgegebenheiten spielen ebenso eine Rolle wie Hintergrundgeräusche und andere Faktoren.
- den benutzten technischen Geräten: Unterschiedliche Mikrophone; Einflüße aus den Übertragungsmedien, wie z.B. ob Breitbandkabel verwendet werden, oder einfache Kupferkabel.
- dem Unterschied von Schriftsprache und gesprochener Sprache (Sprechsprache): Spontane Sprache unterliegt nicht so strengen grammatikalischen Regeln wie die Schriftsprache. So werden zum Beispiel beim spontanen Sprechen Pausen an Stellen gemacht, in denen bei der Schriftsprache keine Pausen gemacht werden würden.

Man erhält einen kleinen Quantisierungsfehler in einem VQs-System, wenn die eingehenden Merkmalsvektoren den Trainingsvektoren ähnlich sind.

Wurde zum Beispiel ein Codebuch erstellt, dessen Trainingsvektoren nur von jungen Sprechern bestimmt wurde, und kommt dann die Situation, das ein älterer Sprecher in das VQs-System spricht, wird der Fehler groß.
Somit wird deutlich, dass bereits die Trainingsvektoren eine große Spanne an Variabilität aufweisen müssen, damit das spätere System für Anwendungen auf vielen unterschiedlichen Gebieten (Z.B.: Auskunft, Anrufbeantwortet, Computer,..) für möglichst viele Menschen gleich gut funktionieren kann.

II. Das Abstandsmaß

Bei der VQ bedient man sich hier in den meisten Fällen beim des einfachen geometrischen Abstandes. Ein gebräuchliche Realisierung sieht folgendermaßen aus:

wenn $|\mathbf{c} - \mathbf{a}_l|^2 \leq |\mathbf{c} - \mathbf{a}_k|^2$ für alle $k \neq 1$ dann ist $\mathbf{c} \in V_l$

wobei

$\mathbf{a}_l \in \Re^D$, $l = 1,...,K$ die Prototypvektoren sind und
$\mathbf{c}$ der zu testende Merkmalsvektor ist.

In der programmtechnischen Realisierung wird also beim Vektorquantisierer der einkommende Merkmalvektor $\mathbf{c}$ mit jedem der Codebuch-Vektoren auf diese Weise verglichen, so daß zum Schluß der Codebuch-Vektor, der den geringsten Abstand zum Merkmalsvektor hat, als Repräsentant genommen wird.

In der Praxis wird außerdem oft ein Gleichheitsmaß benötigt. Dieses Maß wird verwendet um zwei Merkmalsvektoren $\mathbf{v}_i$ und $\mathbf{v}_j$ vergleichen zu können, nachdem sie durch den VQ auf Indices reduziert wurden:

$$d(\mathbf{v}_i,\mathbf{v}_j) = d_{ij} \begin{cases} = 0 \text{ falls } \mathbf{v}_i = \mathbf{v}_j \\ > 0 \text{ sonst} \end{cases}$$

Bei der VQ reduziert sich der Vergleich zweier Merkmalvektoren somit im Prinzip auf einen „table-lookup“, bei dem die Indices verglichen werden.

III. Algorithmen zur Codebuch-Erstellung

Es gibt mehrere Algorithmen, die die Erstellung eines Codebuches erzielen. Die am häufigst benutzten werden im folgenden kurz vorgestellt. Bei allen gleich sind die Startvorgaben: Wir haben Trainingsvektoren und wollen das optimale Codebuch erhalten.
Folgende Algorithmen stehen uns zur Auswahl:

- Lloyd-Algorithmus (1D-Variante; siehe „Digitalisierung von Signalen“)
- LBG-Algorithmus
- K-means Iteration
- SYSDATA

a) Der LBG-Algorithmus

Der LBG-Algorithmus (nach Linde, Buzo und Gray) beruht darauf, daß die jeweiligen Codebuch-Vektoren bei jedem Durchlauf von allen Trainingsvektoren neu berchnet werden. Dies geschiet durch eine

Schwerpunktsberechnung über die Koordinaten der zugehörigen Trainingsvektoren.
Folgendes Schema beschreibt das Vorgehen:

- Wähle die Anzahl K der Codebuchklassen
- Wähle initiale Prototypvektoren $Z^{(0)} = (Z_k^{(0)} \mid k= 1, .. ,K)$
- Definiere ein Abbruchkriterium
- Für i=1,2,3,...

(1) Klassifiziere alle Trainingsvektoren $x \in \omega$ und bestimme daraus die Partitionierung

$$Y_k^{(i)} = Y_k(Z^{(i-1)}), k = 1,...,K$$

(2) Berechne das neue Codebuch Z(i) mit den Klassenzentroiden

$$Z_k^{(i)} = (1 / N_k^{(i)}) \sum x$$

$N_k^{(i)}$ ist dabei die Anzahl der Trainingsvektoren in der Partition $Y_k^{(i)}$

(3) Wenn das Abbruchkriterium erfüllt ist, beende. Sonst setze i=i+1 und fange wieder bei (1) an.

Das gesuchte Codebuch ist $Z^{(i)}$, und der Vektorquantisierer entscheidet gemäß der Minimumsabstandsregel.

Der LBG-Algorithmus hat allerdings einen nicht unwesentlichen Nachteil.
Bei der Initialisierung müssen initiale Prototypvektoren gewählt werden. Dies bedingt eine Art Vorgriff auf ein Codebuch, das erst gemacht werden soll.
Dies kann gefährlich sein, wenn man zu sehr „daneben tippt“, da dann alle Trainingsvektoren Prototypvektoren zugeordnet werden könnten, die falsch sind.
Zudem erfolgt die Klassifikation durch den minimalen Abstand zu den Prototypvektoren. Dies führt aber nur dann zu ordentlichen Ergebnissen führen, wenn bereits gute initiale Prototypvektoren vorliegen. Aber wie kann man diese vorraussetzen, wenn man diese als Ziel hat?
Es ist also ein Problem derart, was zuerst da war: Die Henne, oder das Ei.

b) K-means

K-means verläuft analog zu LBG. Der einzige Unterschied besteht darin, daß Punkt (2) nach jedem neuen Trainingsvektor durchgeführt wird, und nicht erst nachdem ein ganzer Satz an Trainingsvektoren aufgenommen wurde.

b) SYSDATA

SYSDATA erweitert den LBG-Algorithmus, vermeidet aber durch eine zweite Iteration das Problem der initialien Codebuch-Vektoren:

(1) Erstellung eines 1-Vektor Codebuchs. Dies ist dann das Zentroid (das Klassenzentrum) der gesamten Satz an Trainingsvektoren.
(2) Verdopplung der Codebuch-Größe durch Splitten der Zentroide in 2 Teile.

(3) Ausführung des K-means-Algorithmus (oder LBG) zur Erstellung des besten Satzes an Codebuch-Vektoren.
(4) Iterative Wiederholung von (2) und (3) bis das Codebuch die gewünschte Größe erreicht hat.

Das Bild (aus „Fundamentals of Speech Recognition“) zeigt dies in besonders einsichtiger Form:

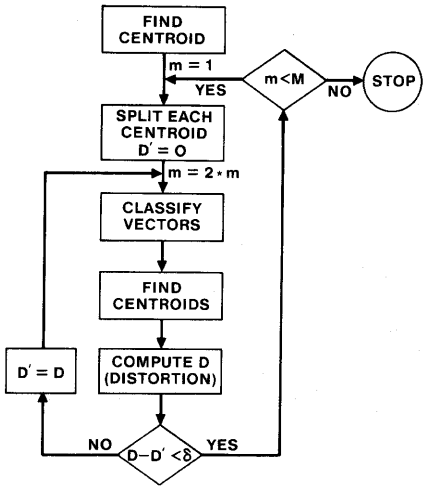

Figure 3.42 Flow diagram of binary split codebook generation algorithm.

2.3. Der mittlere quadratische Fehler

Wie oben bereits bei der Beschreibung der Trainingsvektoren angedeutet, ist mit jedem Codebuch alleine aufgrund der Sammlung der Trainingsvektoren ein gewisser Fehler verbunden. Dieser Fehler läßt sich mit Hilfe folgender Formel berechnen:

$$E = \sum_{l=1}^{k} \int_{\mathbf{c}\in\mathbf{a}_l} |\mathbf{c} - \mathbf{a}_l|^2 P(\mathbf{c})\, dc$$

wobei:

c der Merkmalvektor,
$\mathbf{a}_l$ der Prototypvektor und
$P(\mathbf{c})$ die kontinuierliche Dichte ist, mit der **c** verteilt ist.

Der Fehler beschreibt die Ungenauigkeit, die zustandekommt, wenn die Merkmalvektoren durch die Codebuch-Vektoren repräsentiert werden. Durch die in 2.1.II. beschriebene Klassifikationsvorschrift werden die Merkmalvektoren durch den Codebuch-Vektor repräsentiert der diesem am nächsten ist. Eine Aussage darüber, wie weit dieser Abstand allerdings wirklich ist, wird dabei nicht gemacht.
Der Fehler wird natürlich dann am geringsten werden, wenn die Codebuch-Größe die Größe des Merkmalsraumes annähert.

2.4. Erweiterungen des Vektorquantisierers

Der Vektorquantsierer läßt sich aufgrund ihres modularen Aufbaus einfach erweitern. Dementsprechend groß ist die Anzahl der realisierten Erweiterungen des VQs-Systeme. Folgende Möglichkeiten haben sich dabei als besonders günstig hervorgetan:

- Nutzung mehrerer Codebücher innerhalb eines Systems. Dem System stehen mehrere unterschiedliche Codebücher zur Auswahl. Bei der Analyse von Merkmalvektoren wird dann das Codebuch ausgewählt, in der der geringste Quantisierungsfehler vorkommt.
- K-Tupel Quantisierer, bei dem auch der zeitliche Verlauf der Merkmalvektoren in festen Zeitabschnitten betrachtet wird.
- Matrix-Quantisierer, bei dem die Daten auch in ihrer unterschiedlichen Länge betrachtet werden. Im Unterschied zum K-Tupel Quantisierer sind hier die Zeiteinheiten nicht fest, sondern variabel.
- Trellis Codes. Bei den Trellis Codes wird zu den normalen Codebüchern eine Wahrscheinlichkeit über die Aufeinanderfolge von Vektoren hinzugezogen. So würde zum Beispiel bei einem deutschem Codebuch die Wahrscheinlichkeit das nach einem „k“ ein „e“ kommt sehr viel höher sein, als daß nach einem „k“ ein „g“ kommt.

3. Zusammenfassung

Die Vektorquantisierung erweist sich als gutes Mittel bei der Verarbeitung von Sprache und Bildern. Dennoch hat sie auch Nachteile.
Die Hauptvorteile der VQ sind:

- Durch die Repräsentation von Eingangssignalen durch Codebuch-Indices verfügt man über eine effiziente und Speicherplatzsparende Möglichkeit Informationen zu verarbeiten.
- Der Vergleich zweier Merkmalvektoren reduziert sich auf ein „table-lookup“ und ist damit sehr schnell.
- Durch Codebuch-Indices erhält man eine diskrete Repräsentation der Phoneme, welche in der Sprachverarbeitung von großer Bedeutung sind.

Die Nachteile der VQ:

- Es bleibt ein Fehler bei der Repräsentation von Merkmalsvektoren durch Codebuch-Vektoren.
- Der Speicherplatzverbrauch eines Codebuches ist von nicht unbeträchtlicher Bedeutung. Je größer das Codebuch, desto größer ist auch der Speicherverbrauch. Außerdem steigt mit der Größe des Codebuchs auch der Rechenaufwand für die Klassifikation von Merkmalvektoren. Es folgt also ein Problem der Balancefindung zwischen Codebuch, Quantisierungsfehler und Berechnungsdauer. Diese Balance zu finden ist nicht trivial.

Da sich die Nachteile der VQ im Prinzip „nur“ aus dem Problem der Balancefindung zusammensetzt und die Vorteile groß sind, gehört sie zu einem sehr wichtigen Verfahren bei der Bild- und Sprachverarbeitung. Hinzu kommt die große Flexibilität und Erweiterbarkeit der VQ.
Die Programmiertechnische Umsetzung ist zudem sehr einfach zu realisieren.

Die VQ ist deswegen ein gutes Hilfsmittel in der Mustererkennung.

4. Literaturübersicht

Folgende Literatur wurde verwendet und kann für das Thema empfohlen werden:

- „Fundamentals of Speech Recognition“, Rabiner & Juang, PTR Prentice Hall, T80 2N23
- „Digital Image Processing“, Pratt, J.Wiley & Sons, T80 2N22
- „Pattern Analysis and Understanding“, Niemann,Springer-Verlag, T80 2N..
- Skripten zur KI (Einführung in die Sprachverarbeitung, Görz) und Mustererkennung.