Randolf Altmeyer

BZQ II: Stochastikpraktikum
Wintersemester 2016
Humboldt-Universität zu Berlin

Projektaufgaben Block 3

1 SPAM vs. HAM: Naive Bayes (13P)

1. Extrahiere die Datei *preparedFeatures.zip* von der Webseite. Lies die Trainingsdateien (*train-features.txt, train-labels.txt*) als Matrizen ein. Verwende dafür den Befehl **read.table**. Die Spalten der Features sind Dokument-ID, Wort-ID und Worthäufigkeit des Wortes im angegebenen Dokument. In der Datei *dictionary.txt* findet man die Zuordnung zwischen einem Wort und seiner Wort-ID. Insgesamt verwenden wir nur die 2500 häufigsten Wörter in allen Emails (Trainings- und Testdaten). Die Labels 1 und 0 entsprechen den Klassen SPAM und HAM (= kein SPAM).

2. Erzeuge eine Featurematrix, wobei jede Zeile einem Dokument entspricht und die Worthäufigkeiten aller möglichen Worte im Dokument enthält. Beachte, dass die meisten Einträge in einer Zeile gleich 0 sein werden, da jede Email nur einen kleinen Teil aller Wörter im Wörterbuch enthält. Benutze dafür die Funktion sparseMatrix aus dem Package *Matrix,* um eine dünnbesetzte Matrix zu erzeugen (und um damit Speicherplatz zu sparen).

3. Erzeuge die Wahrscheinlichkeiten für den Naive-Bayes-Classifier aus der Vorlesung. Beachte dabei Folgendes. Ein Dokument entspricht einem Vektor $x \in \{1,\dots,2500\}^p$, wobei $1 \le p \le p_{\max}$ und $p_{\max}$ ist die maximale Länge aller Emails, die wir betrachten. Die Variablen (X_j, Y_j) aus der Vorlesung enthalten dann für das j-te Dokument die Klasse $Y_j \in \{0,1\}$ und den Vektor $X_j \in \{1,\dots,2500\}^{p_j}$, wobei $1 \le p_j \le p_{\max}$ die Länge des j-ten Dokuments ist, und es gilt $X_j^{(k)} = i$, wenn das k-te Wort im Dokument dem i-ten Wort im Wörterbuch entspricht. Insbesondere ist die Verteilung $\mathbb{P}^X$ von X auf der Menge aller möglichen Dokumente definiert. Unsere Features enthalten allerdings keine Informationen über die Wortpositionen innerhalb einer Email. Denn wir machen die *naive* Annahme, dass alle Wortpositionen in einer Email, gegeben die Klasse Y, unabhängig voneinander sind. Daher ignorieren wir die Position und schätzen für $x \in \{1,\dots,2500\}^p$

$$\mathbb{P}(X = x \,|\, Y = 1) = \Phi_{x|1} = \prod_{k=1}^{p} \Phi_{x_k|1}$$

und für alle $i \in \{1,\dots,2500\}$ *unabhängig von k (!)*

$$\Phi_{i|1} \approx \frac{\sum_{j=1}^{n} \sum_{m=1}^{p_j} \mathbf{1}(X_j^{(m)} = i, Y_j = 1) + 1}{\sum_{j=1}^{n} p_j \mathbf{1}(Y_j = 1) + |V|}.$$

Hierbei entspricht i dem Wort x_k an der k-ten Stelle. Allerdings ist $\Phi_{i|1}$ für alle k gleich. $|V|$ ist die Größe des Wörterbuchs. $\Phi_{x|0}$ und die $\Phi_{i|0}$ erhält man ähnlich. Beachte, dass Zähler und Nenner leicht angepasst wurden im Vergleich zur Vorlesung. Gib eine passende Erklärung für diese Anpassung an.

4. Lies die Testdateien (*testFeatures.txt, test-label.txt*) als Matrizen ein wie oben. Ermittle mit den geschätzten Parametern aus 3. für jedes Dokument eine der Klassen 0 oder 1 (beachte die Verwendung von Logarithmen, wie in der Vorlesung angegeben). Vergleiche mit den angegebenen Labels in *test-labels.txt*. Diskutiere das Ergebnis. Welche Wörter sind besonders gute Indikatoren für SPAM oder HAM?

   Hinweis: Es sollten 6 falsche Klassifikationen erfolgen.

5. Betrachte die folgenden Modifikationen:

   (a) Kann man die Ergebnisse eventuell verbessern, wenn man die Features anpasst? Diskutiere, implementiere und teste verschiedene Möglichkeiten.

   Hinweise: Entferne bedeutungslose Wörter aus dem Wörterbuch. Erzeuge zusätzliche Features, z.B. die Anzahl der Wörter in einem Dokument. Erkläre für neue Features auch wie sich die Variablen (X_j, Y_j) und die Parameter $\Phi_{x|1}$ von oben ändern (unter der Annahme von Unabhängigkeit aller Features).

   (b) Die Trainingsdaten umfassen 700 Dokumente, die Testdaten nochmal 300. Vermutlich werden die Ergebnisse besser, wenn man mehr Trainingsdaten verwendet und schlechter, wenn man weniger verwendet. Überprüfe wie sich die Ergebnisse mit weniger bzw. mehr Daten verändern. Gibt es einen systematischen Weg dies zu tun?

   Hinweis: Recherchiere und erkläre den Begriff *m-fold cross-validation* (2 Extrapunkte für Implementierung und Testen).

Freiwillig (bis zu 5 Extrapunkte): Extrahiere die Datei *rawFeatures.zip* von der Webseite. Dort findet man die ursprünglichen Emails, nachdem bereits eine Vorverarbeitung stattgefunden hat (keine Artikel, nur Infinitive, usw.). Lies die Dateien ein und erzeuge die Features aus 1. selber. Wenn du die 2500 häufigsten Wörter verwendest, solltest du dieselben Dateien wie in *preparedFeatures.zip* erhalten. Betrachte außerdem neue Features, die auch die Wortpositionen miteinbeziehen. Erläutere wie in (5a) wie sich die Variablen und Parameter ändern.

Hinweis: Setze das Arbeitsverzeichnis (mit **setwd**) so, dass sich alle notwendigen Dateien von der Webseite im Ordner *../data* befinden. Bitte keine Emails oder sonstige Dateien abgeben, die nicht zu den R-Codes oder der Doku gehören.

2 SPAM vs. HAM: Lineare Regression & Lasso (7P)

1. Betrachte den Kleinste-Quadrate-Schätzer $\hat{\beta}$ und den Ridge-Regression-Schätzer $\hat{\beta}^{RR}$ aus der Vorlesung für $\lambda > 0$.

   (a) Bestimme den Schätzer $\hat{\beta}^{RR}$ explizit.

   (b) Bestimme Erwartungswert und Varianz von $\hat{\beta}$ und $\hat{\beta}^{RR}$. Was passiert, wenn $\lambda \to 0$ bzw. $\lambda \to \infty$? Beachte, dass $\hat{\beta}^{RR}$ auch wohldefiniert ist, wenn $X^T X$ nicht vollen Rang hat.

   Hinweis: Die Varianz einer vektorwertigen Zufallsvariablen $Y \in \mathbb{R}^p$ ist definiert als $\mathrm{Var}Y = \mathbb{E}[(Y - \mathbb{E}[Y])(Y - \mathbb{E}[Y])^\top)]$.

   (c) Bestimme den mittleren quadratischen Fehler von $\hat{\beta}$, d.h. berechne $\mathbb{E}[\|\hat{\beta} - \beta\|^2] = \mathbb{E}[(\hat{\beta} - \beta)^\top(\hat{\beta} - \beta)]$, und genauso bezüglich $\hat{\beta}^{RR}$. Erkläre im Spezialfall von orthogonalem Design, d.h. für $X^T X = I_p$, warum die Koeffizienten von $\hat{\beta}^{RR}$ zur 0 tendieren. Erkläre damit kurz (!) die Hauptidee von Regularisierung im linearen Modell und in welchem Sinne $\hat{\beta}^{RR}$ „besser“ als $\hat{\beta}$ ist.

   Hinweis: Verwende eine Bias-Varianz-Zerlegung für vektorwertige Zufallsvariablen.

2. Wir nehmen an, dass die Klassifikationen in der letzten Aufgabe mit einem linearen Modell $Y = X\beta + \varepsilon$ modelliert werden können, wobei die Zeilen der Matrix $X \in \mathbb{R}^{n \times p}$ für ein Dokument den Wortfrequenzen der Wörter im Wörterbuch entsprechen, d.h. X_{ij} gibt an wie oft das j-te Wort im Wörterbuch im i-ten Dokument vorkommt. Wie vorher enthält $Y \in \{0,1\}^n$ die Klassifikationen in die Klassen SPAM und HAM und $\varepsilon \in \mathbb{R}^n$ beschreibt die Fehler, die wir bei der Klassifikation machen. Nimm an, dass die Komponenten von ε unabhängig sind.

   (a) Implementiere einen Klassifizierer für die Daten aus der letzten Aufgabe mit Hilfe des Lasso-Schätzers. Verwende dafür die Methoden **glmnet** und **predict** (inklusive Kreuzvalidierung) wie in der Datei *testLinModel.R* auf der Webseite. Beachte dabei, dass die Klassen nur 0 oder 1 sein sollen.

   (b) Vergleiche und kommentiere (kurz!) die Hauptergebnisse und Unterschiede zur Naive Bayes-Methode.

3 PCA & Eigenfaces (Freiwillig, 5 Extrapunkte)

Hinweise zur Abgabe:

Alle Dateien (pdf der Auswertung, R-Code für jede einzelne Aufgabe in eigener Datei, aber OHNE Daten!!) „gepackt“ (z.B. mit zip) mit dem Namen *UE3_ Student1_ Student2* bis zum 10.01. per Email an *altmeyrx@math.hu-berlin.de* schicken.