Praktikum Maschinelles Lernen und Datenanalyse

Sommersemester 2011

Fachgebiet Maschinelles Lernen
Institut für Softwaretechnik und
theoretische Informatik
Fakultät IV, Technische Universität Berlin
Prof. Dr. Klaus-Robert Müller
Email: klaus-robert.mueller@tu-berlin.de

Übungsblatt 4: Grundlagen der Klassifikation

Abgabeschluss: Siehe website.

Für dieses Aufgabenblatt sind sowohl Code als auch eine schriftliche Ausarbeitung abzugeben. Der Code und eine elektronische Version der Ausarbeitung (als PDF) muss über PASS abgegeben werden (siehe link auf der Website).

Aufgaben

Teil 1: Implementation

Aufgabe 1 (5 Punkte)

Implementiere Kreuzvalidierung als allgemeine Funktion, die für unterschiedliche Methoden und Gütekriterien verwendet werden kann:

```
[ C, opt_param ] = cv(X, y, classifier_handle, { param_name, value_range, ... },
                      nfolds, nrepetitions, loss_function)
```

Die Argumente haben die folgende Bedeutung.

1. `X` ist die $(d \times n)$-Matrix der Daten.
2. `y` ist ein $(1 \times n)$-Vektor, welcher die labels oder regression targets zu jedem Datenpunkt enthaelt.
3. `classifier_handle` ist der handle des zu verwendenden Classifiers: die Funktion `classifier_handle` soll folgende Signatur haben,

   ```
   C = classifier_handle(X, y, param1, param2, ...)
   ```

   und den Klassikator auf den gegebenen Trainingsdaten `X`, `y` und Parametern trainieren und sämtliche zur Anwendung notwendige Information in der Struktur `C` zurueckliefern.

   Insbesondere enthält die Struktur `C` das Feld `applyfunc`, die einen Handle auf die Auswertungsfunktion `y = feval(C.applyfunc, C, X)` enthält, welche den Klassifikator `C` auf die Daten `X` anwendet und die labels (oder regression targets) `y` berechnet. (Hinweis: Der Functionhandle kann auch auf eine lokale Funktion zeigen. Implementiere die Auswertungsfunktion also als lokale Funktion immerhalb des Classifiers.)
4. Eine Liste bestehend aus Parameternamen (`param_name`) und Wertebereichen (`value_range`). Die Kreuzvalidierung wird für alle so gegebenen Kombinationen von Parameterwerten durchgeführt. Die Reihenfolge der Parameter in dieser Liste muss ihrer Reihenfolge im Handle `classifier_handle` entsprechen.
5. `nfolds` ist die Anzahl der Partitionen (m im Skript). Dieser Parameter ist optional mit dem Standardwert 10.
6. `nrepetitions` ist die Anzahl der Wiederholungen (r im Skript). Dieser Parameter ist optional mit dem Standardwert 5.
7. `loss_function` ist der Namen der zu verwendenden Loss-Function. Diese hat die Signatur `l = loss_function(X, y_true, y_pred)` wobei `X` die Daten sind, `y_true` die echten Labels und `y_pred` die berechneten. Schreibe eine Loss-Function mit den Namen `zero_one_loss`, die den Klassifikationsfehler zwischen 0 und 1 zurückliefert. Dieser Parameter ist optional mit `'zero_one_loss'` als Standardwert.

Das Ergebnis von `cv` ist

1. `C`, der Klassifikator trainiert mit den besten Parametern.
2. `opt_param`, eine Struktur welche die optimalen Parameter enthält. Für jeden Parameter gibt es ein gleichnamiges Attribut (siehe `param_name`), das seinen Wert enthaelt.

Die Funktion soll auf der Kommandozeile ueber den Fortschritt berichten und dabei auch eine Schätzung für die verbleibende Laufzeit angeben (siehe `tic` und `toc`)

Aufgabe 2 (5 Punkte)

Implementiere Kernel Ridge Regression

```
C = krr(X, y, kernel, kernelparameter, regularization)
```

Die folgenden Kerne (mit ihren dazugehörigen Parametern) sollen implementiert werden:

Name	Kern	Parameter
`linear`	$k(x,z) = \langle x, z\rangle$	(keiner)
`polynomial`	$k(x,z) = (\langle x, z\rangle + 1)^p$	Grad $p \in \{1, 2, 3, \ldots\}$
`gaussian`	$k(x,z) = \exp(-\|x-z\|^2/2dw^2)$	Kernbreite w

Hier ist d die Dimension der X.

Der Parameter `regularization` ist die Regularisierungskonstante c in $\hat{\alpha} = (K + cI)^{-1}y$. Falls `regularization` null ist, führe Leave-One-Out Crossvalidierung über c effizient durch. Verwende als Kandidaten für c die Eigenwerte der Kernmatrix K.

Verwende die "noisy sinc function" (siehe Homepage) zum Testen der Funktion.

Teil 2: Anwendung

Aufgabe 3 (5 Punkte)

Schreibe eine Funktion

```
roc_curve(n)
```

welche fuer das eindimensionale binaere Klassifikationsproblem mit Gaussschen Klassen und identischen class priors,

$$\begin{aligned} p(x|y=-1) &\sim \mathcal{N}(\mu = 0, \sigma^2 = 1) \\ p(x|y=+1) &\sim \mathcal{N}(\mu = 2, \sigma^2 = 1) \\ p(y=-1) &= 0.5 \\ p(y=+1) &= 0.5 \end{aligned}$$

die ROC-Curve (siehe Skript) des linearen Klassifikators f_{x_0} gegeben durch

$$f_{x_0}(x) = \begin{cases} -1 & : \quad x \leq x_0 \\ +1 & : \quad x > x_0 \end{cases}$$

plottet. Die Funktion `roc_curve` soll zwei subplots in einer figure erstellen:

1. Die analytisch berechnete ROC-Curve. Verwende dazu die gegebenen Dichten.
2. Die ROC-Curve ermittelt durch Simulation mit `n` Datenpunkten. Generiere Dir dazu zufaellige Daten aus den gegebenen Verteilungen.

Versehe beide Plots mit Titel und Achsenbeschriftungen.

Aufgabe 4 (5 Punkte)

Auf unserer Homepage findest Du verschiedene Datensätze für Kernel-Ridge-Regression. Wende die Algorithmen jeweils auf die Datensätze an und bestimme per Cross-validiation den besten Classifier. Gib jeweils den besten Klassifier (d.h. die Struktur `C`) und die vorhergesagten Labels auf dem Testdatensatz ab. Lege hierfür eine Struktur `results` an, die folgende Struktur hat:

```
result.krr.banana_predictor         % der gelernte Predictor für KRR
result.krr.banana_predicted_labels  % die Label auf dem Testdatensatz
result.krr.diabetis_predictor       % dasselbe für den Diabetis-Datensatz
result.krr.diabetis_predicted_labels
```

und speichere die Struktur `results` in der Datei `result.mat`. Plotte die resultierenden ROC-Kurven fuer Kernel-Ridge-Regression, indem Du den bias-term verschiebst.