IS Praktikum 3: Lineare und logistische Regression

Für viele Anwendungsfälle im Machine Learning wird ein überwachter Lernprozess verwendet. Dies kann beispielsweise zur Vorhersage der Preise von Wohnraum in Abhänigkeit vom räumlichen oder sozialen Umfeld, zur Identifikation von Spam-Mails oder auch zur Erkennung von unerlaubtem Zugriff auf Kreditkarten angewendet werden. Viele Probleme lassen sich dabei mit einem einfachen Algorithmus lösen.

In diesem Praktikum soll dazu die **lineare** und **logistische Regression** genauer betrachtet werden. Durch den vergleichsweise einfachen Algorithmus bietet es sich an, die grundlegenden Konzepte des Machine Learnings anhand von einfachen Beispielen zu erproben. Der grundlegende Regressions-Algorithmus lässt sich dabei zur numerischen Vorhersage, als auch zur Kategorisierung verwenden.

Die hierzu notwendigen Schritte lassen sich folgendermaßen zusammenfassen (vgl. Vorlesung):

- Festlegen der Input-Paare (Attribute, Zielvariable) für Training und Validierung
- Aufstellen der Hypothese zur Lösung der Aufgabenstellung
- Definieren der Cost-Funktion zur Beurteilung des Ergebnisses

Zur Lösung der folgenden Aufgaben steht jeweils Python-Code zur Verfügung, der um Kern-Konzepte ergänzt werden soll. Die notwendigen Ergänzungen werden in den Aufgaben beschrieben und sind jeweils im Code markiert (achten Sie auf die `TODOs`). Die ersten Teilaufgaben aus Aufgabe 1 und 2 behandeln dabei die Implementierung der grundlegenden Methodik, wohingegen die jeweils letzte Teilaufgabe auf die Anwendung anhand eines interessanten Datensatzes abzielt.

Hilfe zur Installation und Einarbeitung:

- Eine Hilfe zur Installation der benötigten Software findet sich im Verzeichnis der Vorlesung.
- Eine gute Einführung zu Python, Numpy und Matplotlib kann unter dem folgenden Link gefunden werden: Python Numpy Tutorial

Hinweis zur Abnahme und zum Protokoll:

Die Abnahme der Aufgaben erfolgt im Rahmen des Praktikumstermins.

Jede Praktikumsgruppe soll die Ergebnisse dieses Termins zudem in Form eines Protokolls festhalten. Dieses Protokoll soll innerhalb einer Woche (23:59 Uhr des Wochentages in der folgenden Woche) nach dem Praktikumstermin zusammen mit dem Python-Code per Email abgegeben werden. Erstellen Sie dazu bitte ein **Zip**-File mit dem folgenden Inhalt:

- Protokoll im PDF-Format
- Python-Skripte

Das Protokoll soll für jede Aufgabe mindestens die von Ihnen verwendeten Formeln, Graphen und die Beantwortung der jeweiligen Fragestellungen enthalten. Beschreiben Sie bitte kurz die Aufgabenstellung und Ihre Lösung in Form eines knappen Fließtextes.

1. Lineare Regression

In dieser Aufgabe soll die in der Vorlesung vorgestellte Methode zur linearen Regression implementiert werden. Dazu kann zunächst das Bestimmen der Gewichte mittels der abgeschlossenen Lösung erfolgen. Anschließend soll dazu *Stochastic Gradient Descent* (SGD) eingesetzt werden. Für das Trainieren und Testen beider Modelle steht jeweils ein Skript zur Verfügung. Darin enthalten ist Code zum Trainieren anhand eines Dummy-Datensatzes und zum anschließenden Plotten der Annäherung.

1.1 Training mittels abgeschlossener Lösung

Es gilt, die in der Vorlesung vorgestellte Formel der abgeschlossenen Form zur Lösung der linearen Regression zu implementieren. Dies kann nativ in Python erfolgen, eine Implementierung mittels **Numpy** ist allerdings effizienter.

Zur Lösung der Aufgabe kann das Skript `linear_regression_closed_form.py` verwendet werden. Das vorliegende Python-Skript beinhaltet die Klasse *LinearRegression* mit den folgenden Methoden:

- **fit():** Methode zum Training des Modells anhand von gegebenen Trainingsdaten
- **predict():** Methode zur numerischen Vorhersage der Zielvariable

Zum Training des Modells muss zunächst die Matrix an Trainingsdaten um den **Bias-Term** ergänzt werden, damit die abgeschlossene Lösung angewendet werden kann. Demnach kann der Vektor der Gewichte des Modells mit eben dieser abgeschlossenen Lösung bestimmt werden. Damit ist in diesem Fall das Training bereits abgeschlossen.

Schließlich soll die Vorhersage der Zielvariable mit den trainierten Gewichten implementiert werden. Evaluieren Sie die Annäherung ihres Modells anhand des Plots aus dem Skript `isp3_1-1.py` und dokumentieren Sie das Ergebnis.

Frage: Was ist der naheliegende Nachteil an der abgeschlossenen Lösung, sobald ein besonders großer Datensatz verwendet werden soll?

1.2 Training mittels Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) gilt als bewährte Methode um Modelle anhand von großen Datenmengen zu trainieren. In der Vorlesung wurde die Update-Regel für Gewichte mittels SGD hergeleitet. Diese soll nun anstelle der zuvor verwendeten abgeschlossenen Lösung eingesetzt werden, um ein Modell zur linearen Regression zu trainieren.

Zur Lösung der Aufgabe kann das Python-Skript `linear_regression_sgd.py` verwendet werden. Die `predict()`-Methode kann dabei aus 1.1 übernommen werden. In der `fit()`-Methode muss hingegen SGD eingesetzt werden. Dazu müssen die folgenden Schritte durchgeführt werden:

- Initialisierung der Gewichte mit einem passenden Anfangswert (Bias-Term beachten)
- Aufstellen der Hypothese $h(x)$

- Aufstellen der Cost-Funktion $J(w)$ als **Sum-of-squared-Errors** (SSE), *optional auch MSE*

Probieren sie unterschiedliche Werte für die Parameter `alpha` und `iterations` in der `predict()`-Funktion aus und betrachten Sie die Auswirkung auf den Graphen des Cost-Wertes. Dokumentieren Sie knapp Ihre Beobachtungen.

Frage: Warum wird in der Praxis *Stochastic Gradient Descent* (SGD) gegenüber *Batch Gradient Descent* (s. Vorlesung) bevorzugt?

1.3 Vorhersage von Immobilienpreisen mit linearer Regression

In dieser Aufgabe soll eine der vorangehenden Lösungen dazu verwendet werden, Immobilienpreise von Wohnungen in *Portland, Oregon* vorherzusagen. Im Gegensatz zu den bisher verwendeten Dummy-Daten, sind in diesem Datensatz drei Merkmale enthalten (Größe, Anzahl der Zimmer, Preis), wobei der Preis als Zielvariable vorhergesagt werden soll.

Die Werte der verschiedenen Merkmale befinden sich dabei auf individuellen Skalen und daher sehr unterschiedlichen Wertebereichen. Damit das Gradientenverfahren schneller konvergiert, sollten die Daten daher Standardisiert werden. Dies können Sie tun, indem Sie den Mittelwert und die Standartabweichung pro Merkmal berechnen. Eine bewährte Methode ist die Standardisierung auf $\mu = 0$ und $\sigma = 1$ Dazu kann die folgende Formel verwendet werden:

$$x'_i = \frac{x_i - \mu}{\sigma}$$

Verwenden Sie das Skript `isp3_1-3.py` zur Lösung der Aufgabe. Das Skript enthällt bereits Code zum Laden des Datensatzes mit **Pandas** und zum Plotten der Ergebnisse. Pandas ermöglicht Ihnen eine sehr einfache Vorverarbeitung der Daten. Dokumentieren Sie kurz, welche Implementierung der linearen Regression Sie verwendet haben und nehmen Sie einen passenden Plot in das Protokoll auf.

2. Logistische Regression

Der Regressions-Algorithmus lässt sich relativ leicht dazu nutzen, eine binäre Klassenzugehörigkeit vorherzusagen. Dazu wird die Hypothese $h_w(x)$ um die **Sigmoid**-Funktion erweitert. In der Vorlesung wurde dazu hergeleitet, dass die Formel zum Gradientenverfahren identisch zur linearen Regression bleibt (bis auf $h_w(x)$). Beachten Sie allerdings, dass die **Cost**-Funktion deutlich anders ist.

Für diese Aufgabe kann die zuvor in 1.2 aufgestellte Formel zum Training mittels SGD verwendet werden.

2.1 Training mittels SGD

Implementieren Sie mithilfe Ihrer Lösung aus Aufgabe 1.2 einen Klassifikator auf Basis der logistischen Funktion mittels SGD. Verwenden Sie dazu das Skript `logistic_regression.py` und erweitern Sie es um die folgenden Konzepte in den Funktionen `fit()` und `predict()`:

- Sigmoid-Funktion $\sigma(x)$
- Hyothese der logistischen Regression $h_w(x)$
- Cost-Funktion der logistischen Regression
- Update-Regel für SGD

Es reicht in dieser Aufgabe, dass die Funktion `predict()` die Wahrscheinlichkeit $P(y = 1)$ ausgibt.

Verwenden Sie das Skript `isp3_2-1.py` zum Laden und Plotten der Trainingsdaten. Das Skript enthällt ebenfalls Code zum Plotten der *Decision-Boundary* des Klassifikators, indem ein Grid an Testdaten erzeugt und mit dem trainierten Klassifikator kategorisiert wird. Überprüfen Sie, welche Auswirkung die Veränderung des Wertes `iterations` und `alpha` in der Funktion `fit()` aus die Qualität der *Decision-Boundary* haben und dokumentieren Sie ihre Ergebnisse im Protokoll.

Frage: Was müssten Sie tun, damit Sie mehr als zwei Klassen vorhersagen können?

2.2 Überleben der Titanic (optional)

Der Titanic-Datensatz ist ein besonders schöner Datensatz zum Erproben von Techniken zur Vorverarbeitung und Klassifikation. Hier finden Sie besonders viele hochwertige Tutorials, die Ihnen am Beispiel der Daten zeigen, welche Art der Gewinnung von Informationen aus Daten möglich ist (*Feature Engineering*) und wie sich dies auf die praktische Zielsetzung auswirken kann.

Das Skript `isp3_2-2.py` stellt Ihnen den Datensatz zur Verfügung. Das Ziel ist dabei, einen Klassifikator zu trainieren, der mit hinreichender Genauigkeit vorhersagen kann, ob eine Person das Titanic-Unglück überlebt. Auf der Plattform **Kaggle** finden Sie dazu viele ausführliche Tutorials:

- Überblick: Titanic: Machine Learning from Disaster
- Python-Beispiel: An Interactive Data Science Tutorial

Viel Erfolg bei der Bearbeitung der Aufgaben!