Dimitri Meier, Saeed Shanidar, Andreas Berks

dimitri.meier@haw-hamburg.de,  
saeed.shanidar@haw-hamburg.de,  
andreas.berks@haw-hamburg.de

06.06.2017

ISP AUFGABE 3

Gruppe 3, Team 5

Inhaltsverzeichnis

[0. Allgemeine Definitionen 2](#_Toc484475278)

[0.1. Konkatenation 2](#_Toc484475279)

[0.2. Bias 3](#_Toc484475280)

[1. Lineare Regression 4](#_Toc484475281)

[1.1. Training mittels abgeschlossener Lösung 4](#_Toc484475282)

[1.1.1. Definition der Trainingsdaten 4](#_Toc484475283)

[1.1.2. Training des Modells 5](#_Toc484475284)

[1.1.3. Aufstellen der Hypothese 7](#_Toc484475285)

[1.1.4. Ergebnis 8](#_Toc484475286)

[1.1.5. Beantwortung der Fragen 9](#_Toc484475287)

[1.2. Training mittels Stochastic Gradient Descent (SGD) 10](#_Toc484475288)

[1.2.1. Definition der Trainingsdaten 10](#_Toc484475289)

[1.2.2. Training des Modells 11](#_Toc484475290)

[1.2.3. Aufstellen der Hypothese 14](#_Toc484475291)

[1.2.4. Ergebnis 15](#_Toc484475292)

[1.2.5. Beantwortung der Fragen 17](#_Toc484475293)

[1.3. Vorhersage von Immobilienpreisen mit linearer Regression 21](#_Toc484475294)

[2. Logistische Regression 23](#_Toc484475295)

[2.1. Training mittels SGD 23](#_Toc484475296)

[2.1.1. Definition der Trainingsdaten 23](#_Toc484475297)

[2.1.2. Training des Modells 23](#_Toc484475298)

[2.1.3. Aufstellen der Hypothese 26](#_Toc484475299)

[2.1.4. Ergebnis 27](#_Toc484475300)

[2.1.5. Beantwortung der Fragen 29](#_Toc484475301)

# 0. Allgemeine Definitionen

## 0.1. Konkatenation

Wir definieren eine Funktion https://latex.codecogs.com/gif.latex?Konkatenation wie folgt:

https://latex.codecogs.com/gif.latex?Konkatenation%3A%20%28%5Cmathbb%7BR%7D%5E%7Bm%2C%201%7D%29%5E%7Bn%7D%20%5Cto%20%5Cmathbb%7BR%7D%5E%7Bm%2C%20n%7D%2C%20x%20%5Cmapsto%20y

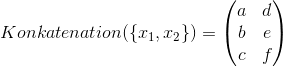
https://latex.codecogs.com/gif.latex?%5Cforall%20i%20%5Cin%20%5B1%2C%20m%5D%2C%20j%20%5Cin%20%5B1%2C%20n%5D%3A%20y_%7Bi%2Cj%7D%20%3D%20%28x_%7Bi%7D%29_%7Bj%7D

**Beispiel:**

Seien zwei Vektoren https://latex.codecogs.com/gif.latex?x_%7B1%7D und https://latex.codecogs.com/gif.latex?x_%7B2%7D gegeben:

Damit ergibt sich:



**Implementation in Python:**



## 0.2. Bias

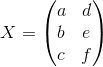
Wir definieren eine Funktion https://latex.codecogs.com/gif.latex?Bias wie folgt:

https://latex.codecogs.com/gif.latex?Bias%3A%20%5Cmathbb%7BR%7D%5E%7Bm%2C%20n%7D%20%5Cto%20%5Cmathbb%7BR%7D%5E%7Bm%2C%201%7D%2C%20x%20%5Cmapsto%20y

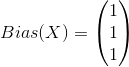
https://latex.codecogs.com/gif.latex?%5Cforall%20i%20%5Cin%20%5B1%2C%20m%5D%3A%20y_%7Bi%2C%201%7D%20%3D%201

**Beispiel:**

Sei eine Matrix https://latex.codecogs.com/gif.latex?X gegeben:



Damit ergibt sich:



**Implementation in Python:**

Wir definieren zuerst eine Hilfsfunktion, welche und die Anzahl der Zeilen und die Anzahl der Spalten der Matrix M liefert:



Nun definieren wir eine Funktion zum Erzeugen des Bias:



Weiterhin definieren wir eine Funktion zum Hinzufügen eines Bias:



# 1. Lineare Regression

## 1.1. Training mittels abgeschlossener Lösung

### 1.1.1. Definition der Trainingsdaten

Gegeben sind die folgenden Trainingsdaten:

|  |  |
| --- | --- |
| https://latex.codecogs.com/gif.latex?x_%7B1%7D | https://latex.codecogs.com/gif.latex?y |
| 0.86 | 2.49 |
| 0.09 | 0.83 |
| -0.85 | -0.25 |
| 0.87 | 3.1 |
| -0.44 | 0.87 |
| -0.43 | 0.02 |
| -1.1 | -0.12 |
| 0.4 | 1.81 |
| -0.96 | -0.83 |
| 0.17 | 0.43 |

Wir definieren nun eine Matrix https://latex.codecogs.com/gif.latex?X welche sämtliche Trainingsdaten beinhaltet.

Dazu konkatenieren wir sämtliche Spaten der Trainingsdaten:

https://latex.codecogs.com/gif.latex?Spalten%20%3D%20%5Cleft%20%5C%7B%20x_%7B1%7D%20%5Cright%20%5C%7D

https://latex.codecogs.com/gif.latex?X%20%3D%20Konkatenation%28Spalten%29

### 1.1.2. Training des Modells

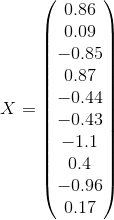
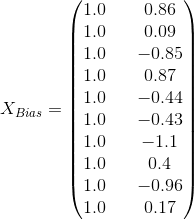
Im Folgenden definieren nun eine Methode https://latex.codecogs.com/gif.latex?fit%28X%2C%20y%29 zum Training des Modells  
anhand von gegebenen Trainingsdaten.

Zuerst fügen wir der Matrix https://latex.codecogs.com/gif.latex?X den Bias https://latex.codecogs.com/gif.latex?x_%7B0%7D hinzu.  
Die entstehende Matrix bezeichnen wir als https://latex.codecogs.com/gif.latex?X_%7BBias%7D .

https://latex.codecogs.com/gif.latex?x_%7B0%7D%20%3D%20Bias%28X%29

https://latex.codecogs.com/gif.latex?X_%7BBias%7D%20%3D%20Konkatenation%28%5Cleft%20%5C%7B%20x_%7B0%7D%20%5Cright%20%5C%7D%20%5Ccup%20Spalten%29

Die entsprechenden Matrizen https://latex.codecogs.com/gif.latex?X und https://latex.codecogs.com/gif.latex?X_%7BBias%7D stehen wie folgt aus:

Nun bestimmen wir die Gewichte https://latex.codecogs.com/gif.latex?%5Ctheta mittels abgeschlossener Lösung:

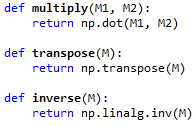
https://latex.codecogs.com/gif.latex?%5Ctheta%20%3D%20%28%7BX_%7BBias%7D%7D%5E%7BT%7D%7BX_%7BBias%7D%7D%29%5E%7B-1%7D%7BX_%7BBias%7D%7D%5E%7BT%7Dy

In diesem Beispiel entstehen dadurch die folgenden Gewichte:

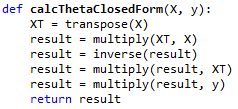
https://latex.codecogs.com/gif.latex?%5Ctheta%20%3D%20%5Cbegin%7Bpmatrix%7D%201.05881340999%20%5C%5C%201.61016841718%20%5Cend%7Bpmatrix%7D

**Implementation in Python:**

Wir definieren zuerst Hilfsfunktionen, um Matrizen zu multiplizieren,  
zu transponieren und zu invertieren:



Nun definieren wir eine Funktion zum Bestimmen der Gewichte:



Wir speichern uns die Werte von https://latex.codecogs.com/gif.latex?%5Ctheta und nutzen diese in der Definition von https://latex.codecogs.com/gif.latex?predict%28X_%7BTest%7D%29 .

Damit ist die Definition der Methode https://latex.codecogs.com/gif.latex?fit%28X%2C%20y%29 zum Training des Modells  
anhand von gegebenen Trainingsdaten abgeschlossen.

### 1.1.3. Aufstellen der Hypothese

Weiterhin definieren wir eine Methode https://latex.codecogs.com/gif.latex?predict%28X_%7BTest%7D%29 zur numerischen Vorhersage der Zielvariable.

Es gilt für https://latex.codecogs.com/gif.latex?X_%7BTest%7D :

https://latex.codecogs.com/gif.latex?X_%7BTest%7D%20%3D%20Konkatenation%28%5Cleft%20%5C%7B%7B%7Bx_%7BTest%7D%7D_%7B1%7D%7D%20%5Cright%20%5C%7D%29

https://latex.codecogs.com/gif.latex?Spalten_%7BTest%7D%20%3D%20%5Cleft%20%5C%7B%20%7B%7Bx_%7BTest%7D%7D_%7B1%7D%7D%20%5Cright%20%5C%7D

Zuerst fügen wir der Matrix https://latex.codecogs.com/gif.latex?X_%7BTest%7D den Bias https://latex.codecogs.com/gif.latex?%7Bx_%7B0%7D%7D_%7BTest%7D hinzu.  
Die entstehende Matrix bezeichnen wir als https://latex.codecogs.com/gif.latex?%7BX_%7BTest%7D%7D_%7BBias%7D .

https://latex.codecogs.com/gif.latex?%7B%7Bx_%7BTest%7D%7D_%7B0%7D%7D%20%3D%20Bias%28X_%7BTest%7D%29

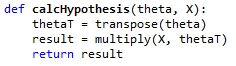
https://latex.codecogs.com/gif.latex?%7BX_%7BTest%7D%7D_%7BBias%7D%20%3D%20Konkatenation%28%5Cleft%20%5C%7B%20%7BX_%7BTest%7D%7D_%7BBias%7D%20%5Cright%20%5C%7D%20%5Ccup%20Spalten_%7BTest%7D%29

Nun berechnen wir die Hypothese https://latex.codecogs.com/gif.latex?%7Bh%28%7BX_%7BTest%7D%7D_%7BBias%7D%29%7D :

https://latex.codecogs.com/gif.latex?h%28%7BX_%7BTest%7D%7D_%7BBias%7D%29%20%3D%20%7BX_%7BTest%7D%7D_%7BBias%7D%5Ctheta

https://latex.codecogs.com/gif.latex?%7Bh%28%7BX_%7BTest%7D%7D_%7BBias%7D%29%7D_%7Bi%2C%20k%7D%20%3D%20%5Csum_%7Bj%3D0%7D%5E%7Bn%7D%20%7BX_%7BTest%7D%7D_%7BBias%7D_%7Bi%2C%20j%7D%20%5Ccdot%20%5Ctheta_%7Bj%2C%20k%7D

**Implementation in Python:**



Das Ergebnis der Hypothese https://latex.codecogs.com/gif.latex?%7Bh%28%7BX_%7BTest%7D%7D_%7BBias%7D%29%7D geben wir als Ergebnis zurück.

Damit ist die Definition der Methode https://latex.codecogs.com/gif.latex?predict%28X_%7BTest%7D%29 zur numerischen Vorhersage der Zielvariable abgeschlossen.

### 1.1.4. Ergebnis

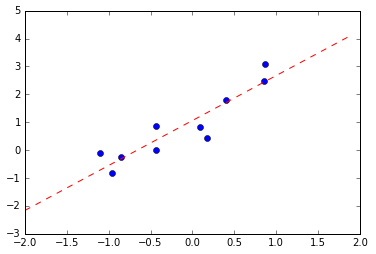
Zur Erinnerung: Gegeben waren die folgenden Trainingsdaten:

|  |  |
| --- | --- |
| https://latex.codecogs.com/gif.latex?x_%7B1%7D | https://latex.codecogs.com/gif.latex?y |
| 0.86 | 2.49 |
| 0.09 | 0.83 |
| -0.85 | -0.25 |
| 0.87 | 3.1 |
| -0.44 | 0.87 |
| -0.43 | 0.02 |
| -1.1 | -0.12 |
| 0.4 | 1.81 |
| -0.96 | -0.83 |
| 0.17 | 0.43 |

Das Ergebnis der Hypothese sieht wie folgt aus:

y



x1

### 1.1.5. Beantwortung der Fragen

**1) Was ist der naheliegende Nachteil an der abgeschlossenen Lösung, sobald ein besonders großer Datensatz verwendet werden soll?**

In diesem Fall ist die abgeschlossene Lösung sehr speicheraufwendig und in der Regel nur selten anwendbar.  
  
**Beispiel:**  
Anzahl der Samples = 1.000.000

Anzahl der Features = 100.000

Für die Matrix https://latex.codecogs.com/gif.latex?M%20%3D%20X%5E%7BT%7DX enthält in diesem Fall 100.000 \* 100.000 = 1010 Elemente. Sind diese Elemente vom Datentyp **float** (64 Bit) wird etwa 80 Gigabyte Hauptspeicher benötigt.

## 1.2. Training mittels Stochastic Gradient Descent (SGD)

### 1.2.1. Definition der Trainingsdaten

Gegeben sind die folgenden Trainingsdaten:

|  |  |
| --- | --- |
| https://latex.codecogs.com/gif.latex?x_%7B1%7D | https://latex.codecogs.com/gif.latex?y |
| 0.86 | 2.49 |
| 0.09 | 0.83 |
| -0.85 | -0.25 |
| 0.87 | 3.1 |
| -0.44 | 0.87 |
| -0.43 | 0.02 |
| -1.1 | -0.12 |
| 0.4 | 1.81 |
| -0.96 | -0.83 |
| 0.17 | 0.43 |

Wir definieren nun eine Matrix https://latex.codecogs.com/gif.latex?X welche sämtliche Trainingsdaten beinhaltet.

Dazu konkatenieren wir sämtliche Spaten der Trainingsdaten:

https://latex.codecogs.com/gif.latex?Spalten%20%3D%20%5Cleft%20%5C%7B%20x_%7B1%7D%20%5Cright%20%5C%7D

https://latex.codecogs.com/gif.latex?X%20%3D%20Konkatenation%28Spalten%29

### 1.2.2. Training des Modells

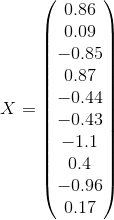
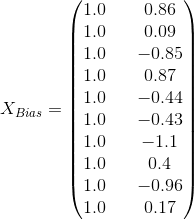
Im Folgenden definieren nun eine Methode https://latex.codecogs.com/gif.latex?fit%28X%2C%20y%29 zum Training des Modells  
anhand von gegebenen Trainingsdaten.

Zuerst fügen wir der Matrix https://latex.codecogs.com/gif.latex?X den Bias https://latex.codecogs.com/gif.latex?x_%7B0%7D hinzu.  
Die entstehende Matrix bezeichnen wir als https://latex.codecogs.com/gif.latex?X_%7BBias%7D .

https://latex.codecogs.com/gif.latex?x_%7B0%7D%20%3D%20Bias%28X%29

https://latex.codecogs.com/gif.latex?X_%7BBias%7D%20%3D%20Konkatenation%28%5Cleft%20%5C%7B%20x_%7B0%7D%20%5Cright%20%5C%7D%20%5Ccup%20Spalten%29

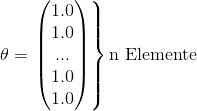
Die entsprechenden Matrizen https://latex.codecogs.com/gif.latex?X und https://latex.codecogs.com/gif.latex?X_%7BBias%7D stehen wie folgt aus:

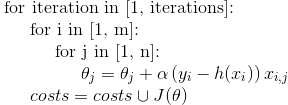
Nun bestimmen wir die Gewichte https://latex.codecogs.com/gif.latex?%5Ctheta mittels Stochastic Gradient Descent:

Zuerst initialisieren wir eine Menge für https://latex.codecogs.com/gif.latex?costs zu speichernden Werte der Kosten und  
initialisieren zusätzlich die Gewichte https://latex.codecogs.com/gif.latex?%5Ctheta mit dem Wert 1.0:

https://latex.codecogs.com/gif.latex?costs%20%3D%20%5Cleft%20%5C%7B%20%5Cright%20%5C%7D



Anschließend werden nach und nach die Gewichte in einer Schleife angenähert:



Als Kostenfunktion nutzen wir die sum-of-squares-errors-Funktion:

https://latex.codecogs.com/gif.latex?J%28%5Ctheta%29%20%3D%20%5Cfrac%7B1%7D%7B2%7D%20%5Csum_%7Bm%7D%5E%7Bi%3D1%7D%7B%28h%28x_%7Bi%7D%29%20-%20y_%7Bi%7D%29%7D%5E%7B2%7D

Seien nun die folgenden Parameter für das Training festgelegt:

https://latex.codecogs.com/gif.latex?alpha%20%3D%200.001%2C%20iterations%20%3D%201000

In diesem Beispiel entstehen dadurch die folgenden Gewichte:

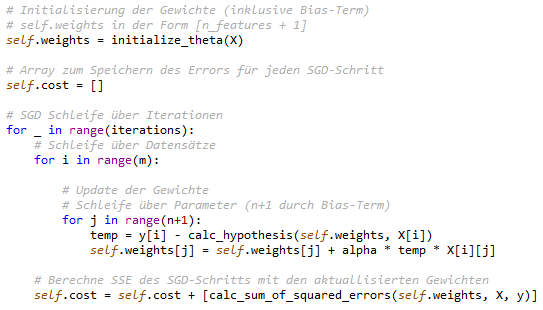
https://latex.codecogs.com/gif.latex?%5Ctheta%20%3D%20%5Cbegin%7Bpmatrix%7D%201.05350947347%20%5C%5C%201.60561505101%20%5Cend%7Bpmatrix%7D

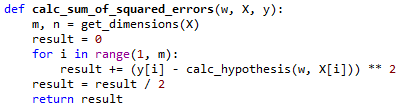
**Implementation in Python:**

Wir definieren zuerst Hilfsfunktionen, um die Gewichte zu initialisieren:



Nun definieren wir eine Funktion zum Bestimmen der Gewichte:



Dabei ist die Kostenfunktion wie folgt definiert:  


Wir speichern uns die Werte von https://latex.codecogs.com/gif.latex?%5Ctheta und nutzen diese in der Definition von https://latex.codecogs.com/gif.latex?predict%28X_%7BTest%7D%29 .

Damit ist die Definition der Methode https://latex.codecogs.com/gif.latex?fit%28X%2C%20y%29 zum Training des Modells  
anhand von gegebenen Trainingsdaten abgeschlossen.

### 1.2.3. Aufstellen der Hypothese

Weiterhin definieren wir eine Methode https://latex.codecogs.com/gif.latex?predict%28X_%7BTest%7D%29 zur numerischen Vorhersage der Zielvariable.

Es gilt für https://latex.codecogs.com/gif.latex?X_%7BTest%7D :

https://latex.codecogs.com/gif.latex?X_%7BTest%7D%20%3D%20Konkatenation%28%5Cleft%20%5C%7B%7B%7Bx_%7BTest%7D%7D_%7B1%7D%7D%20%5Cright%20%5C%7D%29

https://latex.codecogs.com/gif.latex?Spalten_%7BTest%7D%20%3D%20%5Cleft%20%5C%7B%20%7B%7Bx_%7BTest%7D%7D_%7B1%7D%7D%20%5Cright%20%5C%7D

Zuerst fügen wir der Matrix https://latex.codecogs.com/gif.latex?X_%7BTest%7D den Bias https://latex.codecogs.com/gif.latex?%7Bx_%7B0%7D%7D_%7BTest%7D hinzu.  
Die entstehende Matrix bezeichnen wir als https://latex.codecogs.com/gif.latex?%7BX_%7BTest%7D%7D_%7BBias%7D .

https://latex.codecogs.com/gif.latex?%7B%7Bx_%7BTest%7D%7D_%7B0%7D%7D%20%3D%20Bias%28X_%7BTest%7D%29

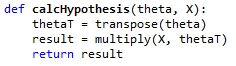
https://latex.codecogs.com/gif.latex?%7BX_%7BTest%7D%7D_%7BBias%7D%20%3D%20Konkatenation%28%5Cleft%20%5C%7B%20%7BX_%7BTest%7D%7D_%7BBias%7D%20%5Cright%20%5C%7D%20%5Ccup%20Spalten_%7BTest%7D%29

Nun berechnen wir die Hypothese https://latex.codecogs.com/gif.latex?%7Bh%28%7BX_%7BTest%7D%7D_%7BBias%7D%29%7D :

https://latex.codecogs.com/gif.latex?h%28%7BX_%7BTest%7D%7D_%7BBias%7D%29%20%3D%20%7BX_%7BTest%7D%7D_%7BBias%7D%5Ctheta

https://latex.codecogs.com/gif.latex?%7Bh%28%7BX_%7BTest%7D%7D_%7BBias%7D%29%7D_%7Bi%2C%20k%7D%20%3D%20%5Csum_%7Bj%3D0%7D%5E%7Bn%7D%20%7BX_%7BTest%7D%7D_%7BBias%7D_%7Bi%2C%20j%7D%20%5Ccdot%20%5Ctheta_%7Bj%2C%20k%7D

**Implementation in Python:**



Das Ergebnis der Hypothese https://latex.codecogs.com/gif.latex?%7Bh%28%7BX_%7BTest%7D%7D_%7BBias%7D%29%7D geben wir als Ergebnis zurück.

Damit ist die Definition der Methode https://latex.codecogs.com/gif.latex?predict%28X_%7BTest%7D%29 zur numerischen Vorhersage der Zielvariable abgeschlossen.

### 1.2.4. Ergebnis

Zur Erinnerung: Gegeben waren die folgenden Trainingsdaten:

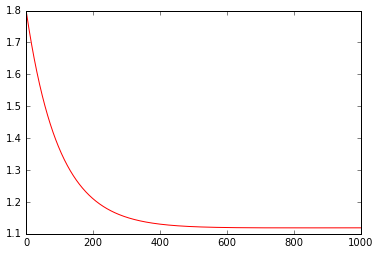
 

|  |  |
| --- | --- |
| https://latex.codecogs.com/gif.latex?x_%7B1%7D | https://latex.codecogs.com/gif.latex?y |
| 0.86 | 2.49 |
| 0.09 | 0.83 |
| -0.85 | -0.25 |
| 0.87 | 3.1 |
| -0.44 | 0.87 |
| -0.43 | 0.02 |
| -1.1 | -0.12 |
| 0.4 | 1.81 |
| -0.96 | -0.83 |
| 0.17 | 0.43 |

Seien nun die folgenden Parameter für das Training festgelegt:

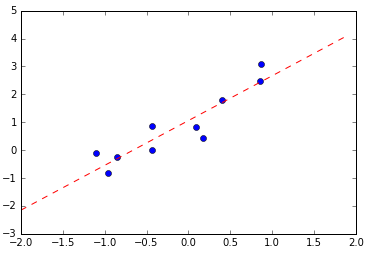
https://latex.codecogs.com/gif.latex?alpha%20%3D%200.001%2C%20iterations%20%3D%201000

Die Fehlerkurve sieht wie folgt aus:



Das Ergebnis der Hypothese sieht wie folgt aus:

y



x1

### 1.2.5. Beantwortung der Fragen

**1) Probieren sie unterschiedliche Werte für die Parameter alpha und iterations in der  
predict()-Funktion aus und betrachten Sie die Auswirkung auf den Graphen des CostWertes.**

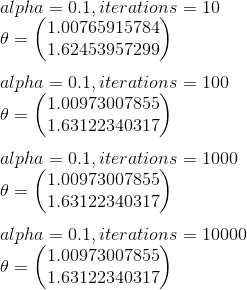
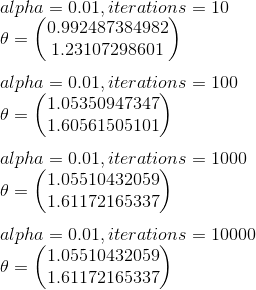
An den folgenden Testdaten lässt sich folgendes ableiten:

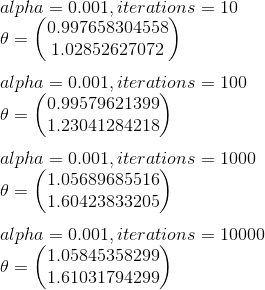
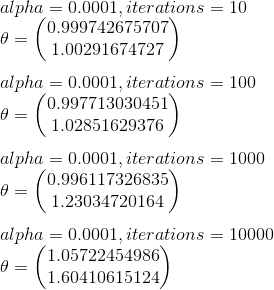
* Je kleiner der Wert für alpha, desto genauer, jedoch auch langsamer die Annährung
* Je größer der Wert für iterations, desto genauer die Annährung, jedoch wird  
  irgendwann Konvergenz erreicht

Ergebnis der geschlossenen Form:

https://latex.codecogs.com/gif.latex?%5Ctheta%20%3D%20%5Cbegin%7Bpmatrix%7D%201.05881340999%20%5C%5C%201.61016841718%20%5Cend%7Bpmatrix%7D

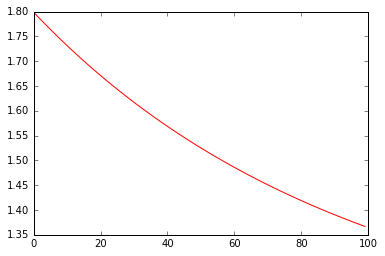
Auswirkungen auf das Ergebnis der Gewichte:

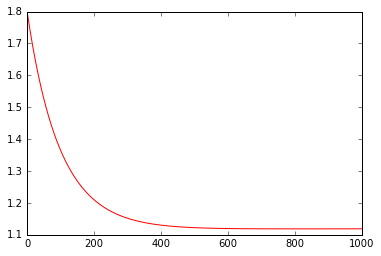
 

Auswirkungen auf den Graphen des CostWertes:

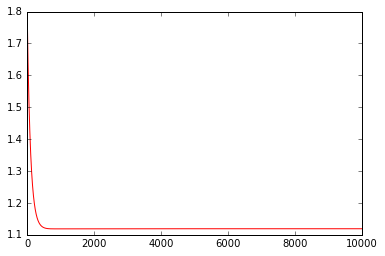
https://latex.codecogs.com/gif.latex?alpha%20%3D%200.001%2C%20iterations%20%3D%20100



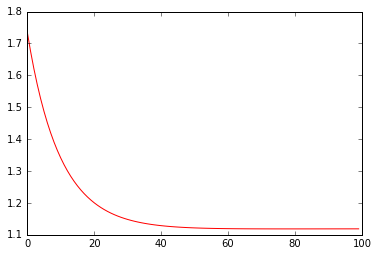
https://latex.codecogs.com/gif.latex?alpha%20%3D%200.001%2C%20iterations%20%3D%201000



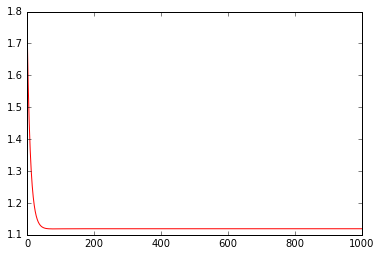
https://latex.codecogs.com/gif.latex?alpha%20%3D%200.001%2C%20iterations%20%3D%2010000



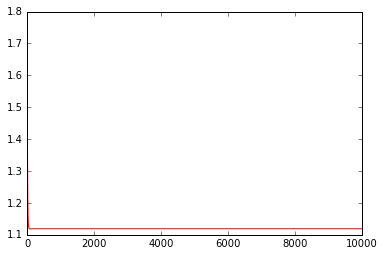
https://latex.codecogs.com/gif.latex?alpha%20%3D%200.01%2C%20iterations%20%3D%20100



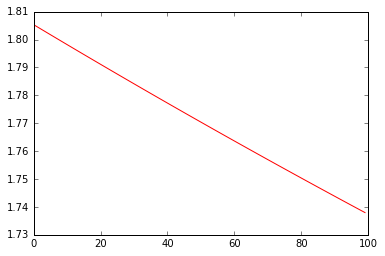
https://latex.codecogs.com/gif.latex?alpha%20%3D%200.01%2C%20iterations%20%3D%201000



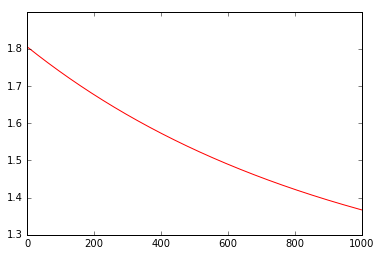
https://latex.codecogs.com/gif.latex?alpha%20%3D%200.01%2C%20iterations%20%3D%2010000



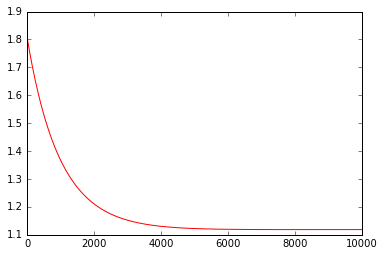
https://latex.codecogs.com/gif.latex?alpha%20%3D%200.0001%2C%20iterations%20%3D%20100



https://latex.codecogs.com/gif.latex?alpha%20%3D%200.0001%2C%20iterations%20%3D%201000



https://latex.codecogs.com/gif.latex?alpha%20%3D%200.0001%2C%20iterations%20%3D%2010000



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

Batch Gradient Descent geht über das gesamte Trainingsset um die Gewichte zu aktualisieren. Dadurch besteht die Gefahr, dass der Hauptspeicher „gesprengt“ wird. Dies führt zu einer langsameren Ausführung (durch Nutzung einer Auslagerungsdatei).

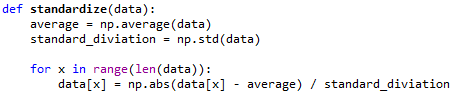
Außerdem übertritt Stochastic Gradient Descent im Gegensatz zu Batch Gradient Descent lokale Optima durch „noisy“ Gradient. Das bedeutet, dass beim Stochastic Gradient Descent ein „Rauschen“ während des Training ausgeglichen wird.

## 1.3. Vorhersage von Immobilienpreisen mit linearer Regression

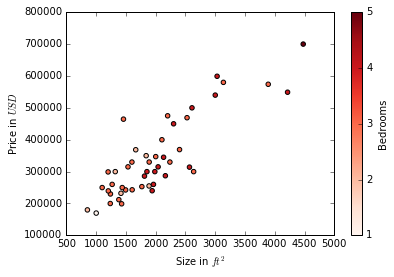
Wir nutzen folgenden Formel zur Standardisierung:



**Implementation in Python:**



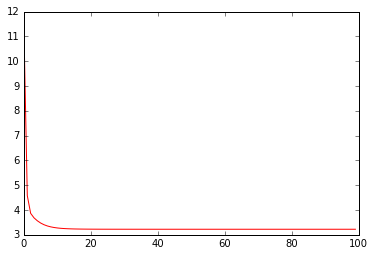
Gegeben waren die folgenden Trainingsdaten:



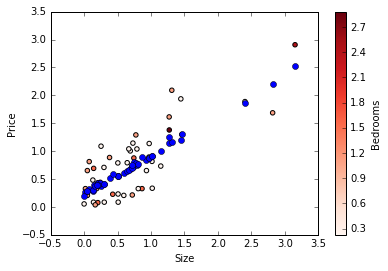
Seien nun die folgenden Parameter für das Training festgelegt:

https://latex.codecogs.com/gif.latex?alpha%20%3D%200.001%2C%20iterations%20%3D%201000

Die Fehlerkurve sieht wie folgt aus:



Das Ergebnis der Hypothese sieht wie folgt aus:



# 2. Logistische Regression

## 2.1. Training mittels SGD

### 2.1.1. Definition der Trainingsdaten

Wir definieren eine Matrix https://latex.codecogs.com/gif.latex?X welche sämtliche Trainingsdaten beinhaltet.

Dazu konkatenieren wir sämtliche Spaten der Trainingsdaten:

https://latex.codecogs.com/gif.latex?Spalten%20%3D%20%5Cleft%20%5C%7Bx_%7B1%7D%2C%20x_%7B2%7D%5Cright%20%5C%7D

https://latex.codecogs.com/gif.latex?X%20%3D%20Konkatenation%28Spalten%29

### 2.1.2. Training des Modells

Im Folgenden definieren nun eine Methode https://latex.codecogs.com/gif.latex?fit%28X%2C%20y%29 zum Training des Modells  
anhand von gegebenen Trainingsdaten.

Zuerst fügen wir der Matrix https://latex.codecogs.com/gif.latex?X den Bias https://latex.codecogs.com/gif.latex?x_%7B0%7D hinzu.  
Die entstehende Matrix bezeichnen wir als https://latex.codecogs.com/gif.latex?X_%7BBias%7D .

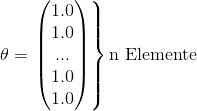
https://latex.codecogs.com/gif.latex?x_%7B0%7D%20%3D%20Bias%28X%29

https://latex.codecogs.com/gif.latex?X_%7BBias%7D%20%3D%20Konkatenation%28%5Cleft%20%5C%7B%20x_%7B0%7D%20%5Cright%20%5C%7D%20%5Ccup%20Spalten%29

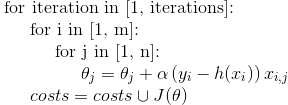
Nun bestimmen wir die Gewichte https://latex.codecogs.com/gif.latex?%5Ctheta mittels Stochastic Gradient Descent:

Zuerst initialisieren wir eine Menge für https://latex.codecogs.com/gif.latex?costs zu speichernden Werte der Kosten und  
initialisieren zusätzlich die Gewichte https://latex.codecogs.com/gif.latex?%5Ctheta mit dem Wert 1.0:

https://latex.codecogs.com/gif.latex?costs%20%3D%20%5Cleft%20%5C%7B%20%5Cright%20%5C%7D



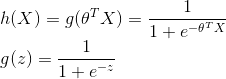
Anschließend werden nach und nach die Gewichte in einer Schleife angenähert:



Als Kostenfunktion nutzen wir die Log-Likelihood-Funktion:

C:\Users\Andreas\AppData\Local\Microsoft\Windows\INetCache\Content.Word\gif.gif

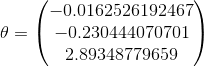
Die Hypothese definieren wir mit Hilfe der Sigmoid-Funktion https://latex.codecogs.com/gif.latex?g%28z%29 :



Seien nun die folgenden Parameter für das Training festgelegt:

https://latex.codecogs.com/gif.latex?alpha%20%3D%200.001%2C%20iterations%20%3D%201000

In diesem Beispiel entstehen dadurch die folgenden Gewichte:

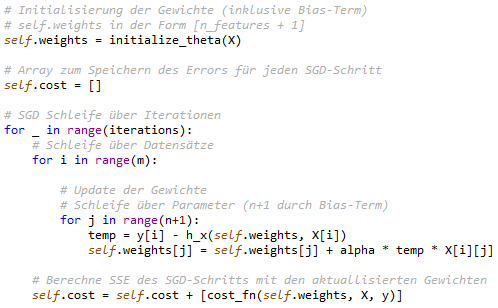


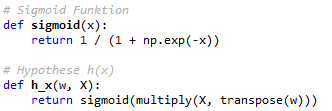
**Implementation in Python:**

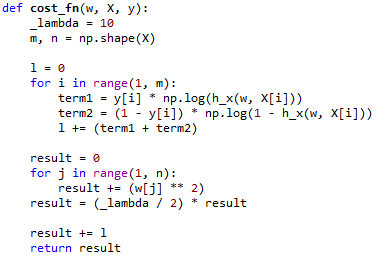
Wir definieren zuerst Hilfsfunktionen, um die Gewichte zu initialisieren:



Nun definieren wir eine Funktion zum Bestimmen der Gewichte:



Dabei ist die Hypothese wie folgt definiert:  


Dabei ist die Kostenfunktion wie folgt definiert:  


Wir speichern uns die Werte von https://latex.codecogs.com/gif.latex?%5Ctheta und nutzen diese in der Definition von https://latex.codecogs.com/gif.latex?predict%28X_%7BTest%7D%29 .

Damit ist die Definition der Methode https://latex.codecogs.com/gif.latex?fit%28X%2C%20y%29 zum Training des Modells  
anhand von gegebenen Trainingsdaten abgeschlossen.

### 2.1.3. Aufstellen der Hypothese

Weiterhin definieren wir eine Methode https://latex.codecogs.com/gif.latex?predict%28X_%7BTest%7D%29 zur numerischen Vorhersage der Zielvariable.

Es gilt für https://latex.codecogs.com/gif.latex?X_%7BTest%7D :

https://latex.codecogs.com/gif.latex?X_%7BTest%7D%20%3D%20Konkatenation%28%5Cleft%20%5C%7B%7B%7Bx_%7BTest%7D%7D_%7B1%7D%7D%20%5Cright%20%5C%7D%29

https://latex.codecogs.com/gif.latex?Spalten_%7BTest%7D%20%3D%20%5Cleft%20%5C%7B%20%7B%7Bx_%7BTest%7D%7D_%7B1%7D%7D%20%5Cright%20%5C%7D

Zuerst fügen wir der Matrix https://latex.codecogs.com/gif.latex?X_%7BTest%7D den Bias https://latex.codecogs.com/gif.latex?%7Bx_%7B0%7D%7D_%7BTest%7D hinzu.  
Die entstehende Matrix bezeichnen wir als https://latex.codecogs.com/gif.latex?%7BX_%7BTest%7D%7D_%7BBias%7D .

https://latex.codecogs.com/gif.latex?%7B%7Bx_%7BTest%7D%7D_%7B0%7D%7D%20%3D%20Bias%28X_%7BTest%7D%29

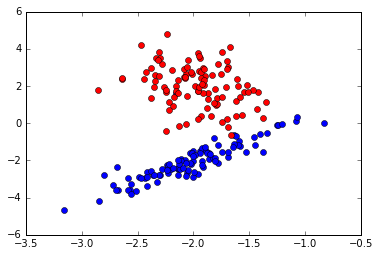
https://latex.codecogs.com/gif.latex?%7BX_%7BTest%7D%7D_%7BBias%7D%20%3D%20Konkatenation%28%5Cleft%20%5C%7B%20%7BX_%7BTest%7D%7D_%7BBias%7D%20%5Cright%20%5C%7D%20%5Ccup%20Spalten_%7BTest%7D%29

Das Ergebnis der Hypothese https://latex.codecogs.com/gif.latex?%7Bh%28%7BX_%7BTest%7D%7D_%7BBias%7D%29%7D geben wir als Ergebnis zurück.

Damit ist die Definition der Methode https://latex.codecogs.com/gif.latex?predict%28X_%7BTest%7D%29 zur numerischen Vorhersage der Zielvariable abgeschlossen.

### 2.1.4. Ergebnis

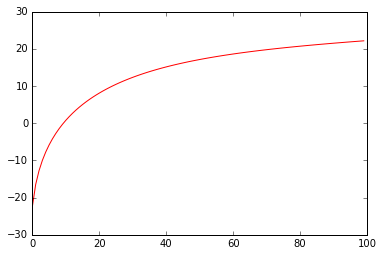
Gegeben waren die folgenden Trainingsdaten:



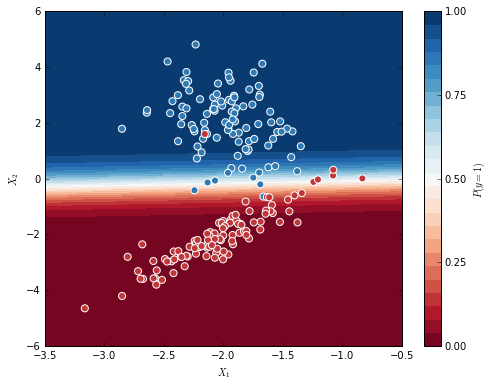
Seien nun die folgenden Parameter für das Training festgelegt:

https://latex.codecogs.com/gif.latex?alpha%20%3D%200.001%2C%20iterations%20%3D%201000

Die Fehlerkurve sieht wie folgt aus:



Das Ergebnis der Hypothese sieht wie folgt aus:



### 2.1.5. Beantwortung der Fragen

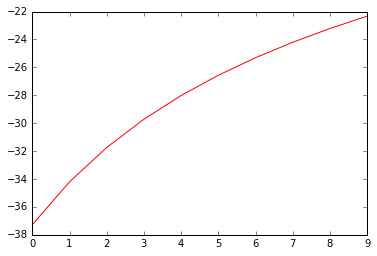
**1) Überprüfen Sie, welche Auswirkung die Veränderung des Wertes iterations und alpha in der Funktion fit() aus die Qualität der Decision-Boundary haben und dokumentieren Sie ihre Ergebnisse im Protokoll.**

An den folgenden Testdaten lässt sich folgendes ableiten:

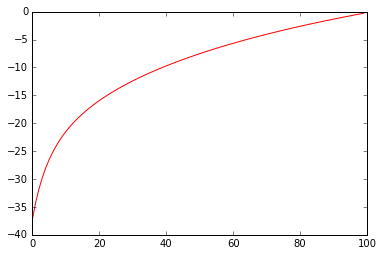
* Je kleiner der Wert für alpha, desto genauer, jedoch auch langsamer die Annährung
* Je größer der Wert für iterations, desto genauer die Annährung, jedoch wird  
  irgendwann Konvergenz erreicht

Auswirkungen auf den Graphen des CostWertes:

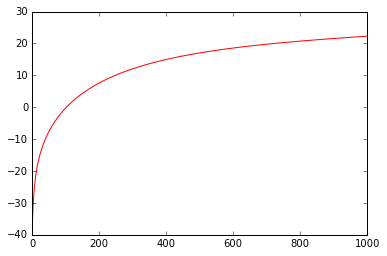
https://latex.codecogs.com/gif.latex?alpha%20%3D%200.001%2C%20iterations%20%3D%2010



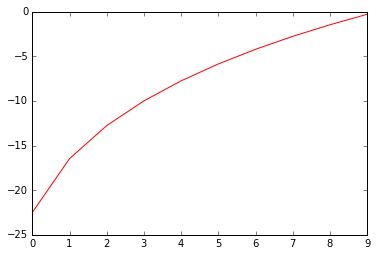
https://latex.codecogs.com/gif.latex?alpha%20%3D%200.001%2C%20iterations%20%3D%20100



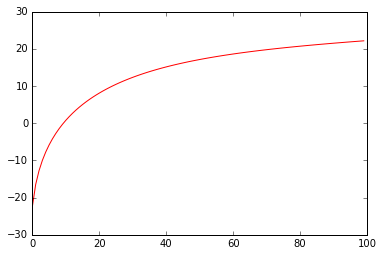
https://latex.codecogs.com/gif.latex?alpha%20%3D%200.001%2C%20iterations%20%3D%201000



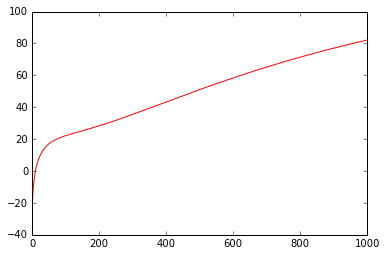
https://latex.codecogs.com/gif.latex?alpha%20%3D%200.01%2C%20iterations%20%3D%2010



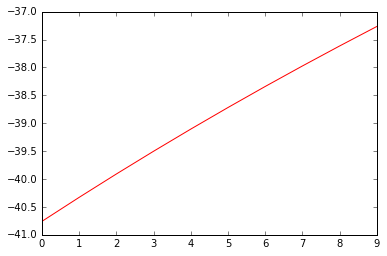
https://latex.codecogs.com/gif.latex?alpha%20%3D%200.01%2C%20iterations%20%3D%20100



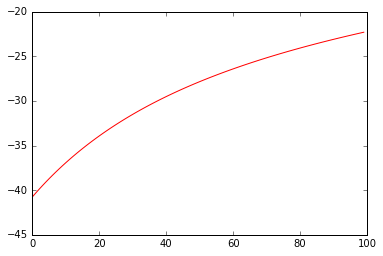
https://latex.codecogs.com/gif.latex?alpha%20%3D%200.01%2C%20iterations%20%3D%201000



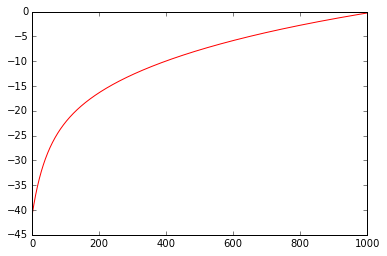
https://latex.codecogs.com/gif.latex?alpha%20%3D%200.0001%2C%20iterations%20%3D%2010



https://latex.codecogs.com/gif.latex?alpha%20%3D%200.0001%2C%20iterations%20%3D%20100



https://latex.codecogs.com/gif.latex?alpha%20%3D%200.0001%2C%20iterations%20%3D%201000



**2) Was müssten Sie tun, damit Sie mehr als zwei Klassen vorhersagen können?**

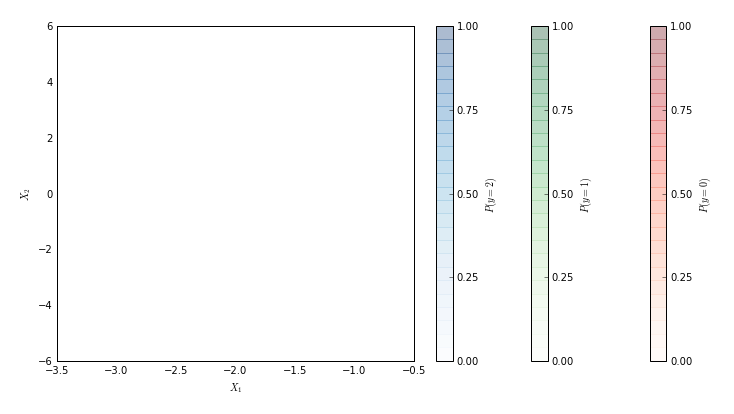
Wir definieren für jede Klasse eigene Vektor mit Zielvariablen https://latex.codecogs.com/gif.latex?y .  
Wenn für eigene gegebene Klasse das Sample der Trainingsdaten https://latex.codecogs.com/gif.latex?X_%7Bi%7D mit https://latex.codecogs.com/gif.latex?i%20%5Cin%20%5B1%2C%20m%5D zu der entsprechenden Klasse gehört, setzen wir den Wert für https://latex.codecogs.com/gif.latex?y_%7Bi%7D%20%3D%201%5C%3B%5Ctext%7B%28true%29%7D .  
Wenn nicht, setzen wir den Wert für https://latex.codecogs.com/gif.latex?y_%7Bi%7D%20%3D%200%5C%3B%5Ctext%7B%28false%29%7D .

Wir führen nun für jede Klasse ein Training durch, um den entsprechenden Wahrscheinlichkeitsgraphen zu erhalten.

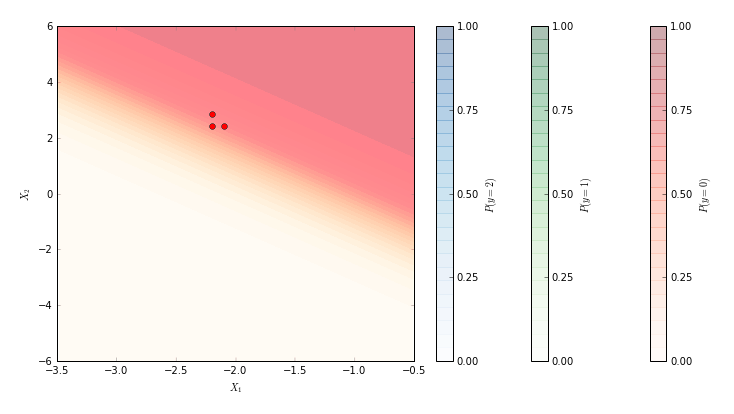
Wir zeichnen nun jeden dieser Wahrscheinlichkeitsgraphen mit einem Alpha-Wert von https://latex.codecogs.com/gif.latex?%5Cfrac%7B%5Ctext%7B1%7D%7D%7B%5Ctext%7BAnzahl%20der%20Klassen%7D%7D nach und nach übereinander.

**Beispiel:**

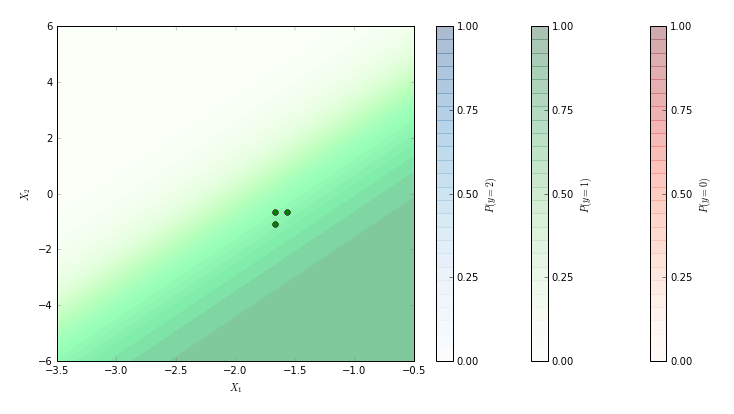
Darstellung des Wahrscheinlichkeitsgraphen keiner Klasse:



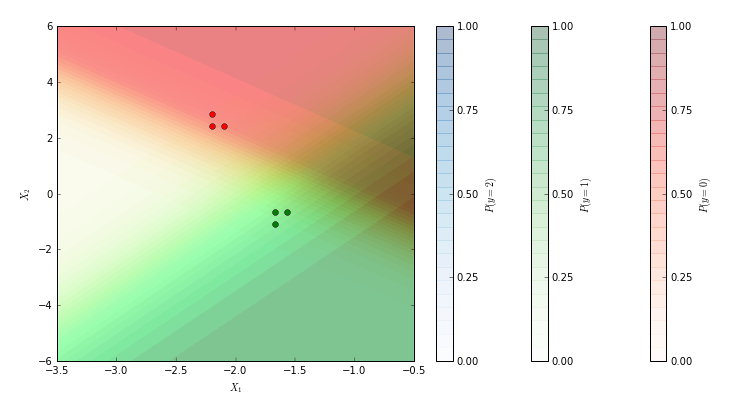
Darstellung des Wahrscheinlichkeitsgraphen der Klasse 0:



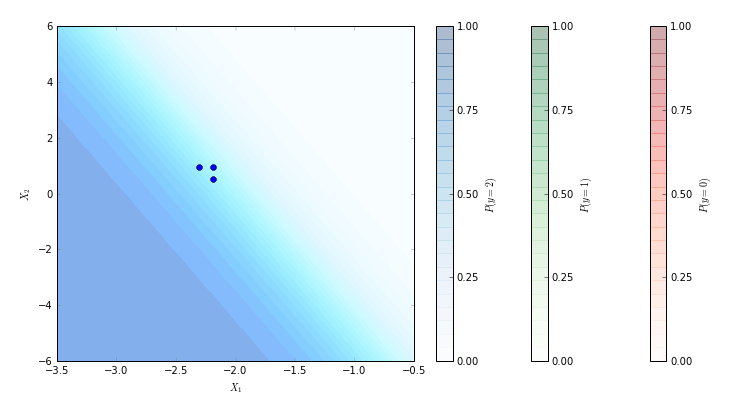
Darstellung des Wahrscheinlichkeitsgraphen der Klasse 1:



Darstellung des Wahrscheinlichkeitsgraphen der Klassen 0 und 1:



Darstellung des Wahrscheinlichkeitsgraphen der Klasse 3:



Darstellung des Wahrscheinlichkeitsgraphen der Klassen 1, 2 und 3:

