Code ist alles top 10/10 für beide Aufgaben. Aber es sieht so aus, als kommst du schnell ins Overfitting. Wenn die Gewichtsmatrix stark nach einer Zahl aussieht, wie hier nach der 1, konzentriert sich das Model nur auf diese Klasse. Eigentlich im binären Fall nicht schlimm, da es automatisch die andere Zahl ist, wenn es nicht die 1 ist, aber nicht ganz so schön und das spiegelt sich auch etwas in der Performance wieder. Mit der Regularisierung erkennt man dann mit einem ziemlich hohen Lambda Wert bei dir eine gute Matrix (1e+05), wo man die 4 überlagert mit der 1 sieht.

Nimmst du allerdings eine größere Schrittweite, kommst du mit geringerer Regularisierung schneller zu einem besseren Ergebnis (99.8%), folglich wirst du momentan in einem lokalen Minimum landen. Die Norm sind adaptive Schrittweiten, so könntest du zB probieren, in den ersten 50 Iterationen mit einer großen Schrittweite zu arbeiten, und danach die Schrittweite alle weiteren 50 Iterationen um 1/10 zu reduzieren. Dadurch vermeidest du es, Anfangs in einem lokalen Minimum zu hängen, und später nicht aus dem Optimum wieder rauszuspringen wegen zu großer Schrittweite. Aber dazu werde ich im kommenden Tutorium mehr erzählen, wenn wir über Loss Contours sprechen.

Addiere ein kleines Epsilon(zB 1e-07) zum log wegen log(0), kommt nur vor mit größeren Schrittweiten.

Ein paar kleine Code Vereinfacherungen:

- Statt .transpose() kann man einfach ".T" benutzen

- Die Klassifizierung mit <0.5 kann man als Einzeiler mit np.around() schreiben (rundet zur nähesten Zahl)

- Anstelle von np.matmul oder np.dot kann man auch "@", was der Matrixmulti Operator ist (seit Python 3.5)

Wenn du in Python mehr mit Machine Learning machen willst, wirst du früher oder später eine der beiden großen Libraries PyTorch oder Tensorflow verwenden müssen, die beide ihre eigenen Matrizen anstelle von numpy arrays verwenden. Im Forschungsbereich wird eher PyTorch verwendet, in der Industrie mehr Tensorflow, also falls du Interesse hast, kannst du dir die gerne anschauen, zB https://pytorch.org/tutorials/beginner/basics/intro.html -> Tensors.