

MAŠINSKO UČENJE: LOGISTIČKA REGRESIJA

Profesor: Veljko M. Milutinović

Asistent: Jelena Hadzi - Purić

Aleksandar Mužina (mr14235)

Maja Vujović (mr14296)

Matematički fakultet u Beogradu

ŠTA JE MASINSKO UČENJE?

- Mašinsko učenje je naučna studija i statistički model koji koriste kompjuterski sistemi da postepeno pobojsaju performanse pri nekom specifičnom zadatku.
- Ovi algoritmi prave matematički model od prikupljenih podataka kako bi mogli da naprave nekakvu predikciju ili odluku bez da su za to eksplicitno programirani.



FORMALNA DEFINICIJA

- *“Kompjuterski program uči iz iskustva E u vezi sa skupom zadataka T i merom performanse P , ako se njegova performasa pri izvršavanju skupa zadataka T , merena sa P poboljšava sa iskustvom E ”*

*- Tom M. Mitchell,
Machine Learning (1997)*



GDE SE KORISTI MAŠINSKO UČENJE?

- Filterovanje mailova
- Pronalaženje “uljeza” na internetu
- Computer vision (Kako da kompjuteri dobiju bolje shvatanje na osnovu digitalnih slika ili videa)
- Data mining (Pronalaženje paternu u velikom broju podataka...)



LOGISTIČKA PROGRESIJA

- U statistici, logistički model je široko korišćen statistički model, koji u svojoj osnovnoj formi, koristi logističku funkciju da modelira binarno zavisne varijable.
- Matematički, binarni logistički model ima zavisne varijable sa dve mogućnosti.

Npr: pao/prošao, pobedio/izgubio, živ/mrtav, zdrav/bolestan.

- Ove 2 varijable označavamo sa 0 i 1. Vrednost ovih varijabli može biti između 1 ili 0 (između 100% i 0%).
- U logističkom modelu, “logaritamske-šanse” (eng. logg-odds) za varijablu obeleženu sa “1” je linearna kombinacija 1 ili više nezavisnih varijabli (“predviđjači”).

PRIMER

Šansa da položimo ispit na osnovu toga koliko sati smo učili
20 studenata je učilo 0-6 sati za ispit. Kolika im je sansa da polože?

Ovde koristimo logističku regresiju zato što imamo situaciju pao/prošao.

Hours	0.50	0.75	1.00	1.25	1.50	1.75	1.75	2.00	2.25	2.50	2.75	3.00	3.25	3.50	4.00	4.25	4.50	4.75	5.00	5.50
Pass	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1

Ubacićemo vrednosti iz tabele u odgovarajuće formule...i dobijamo sledeću tabelu

	Coefficient	Std.Error	z-value	P-value (Wald)
Intercept	-4.0777	1.7610	-2.316	0.0206
Hours	1.5046	0.6287	2.393	0.0167

P(Wald test) nam pokazuje u kakvoj su zavisnost
sati učenja i prolaznost. Kako je $p = 0.0167$ to znači

da su ove 2 stvari veoma povezan. Koeficijente ćemo ubaciti u logističku regresiju

kako bi procenili šanse prolaska na ispitu.

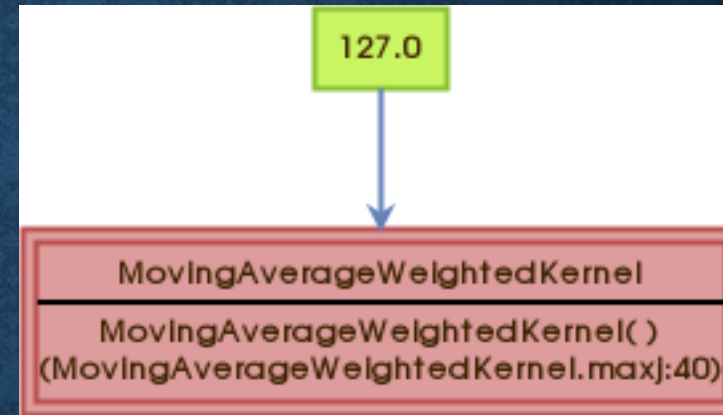
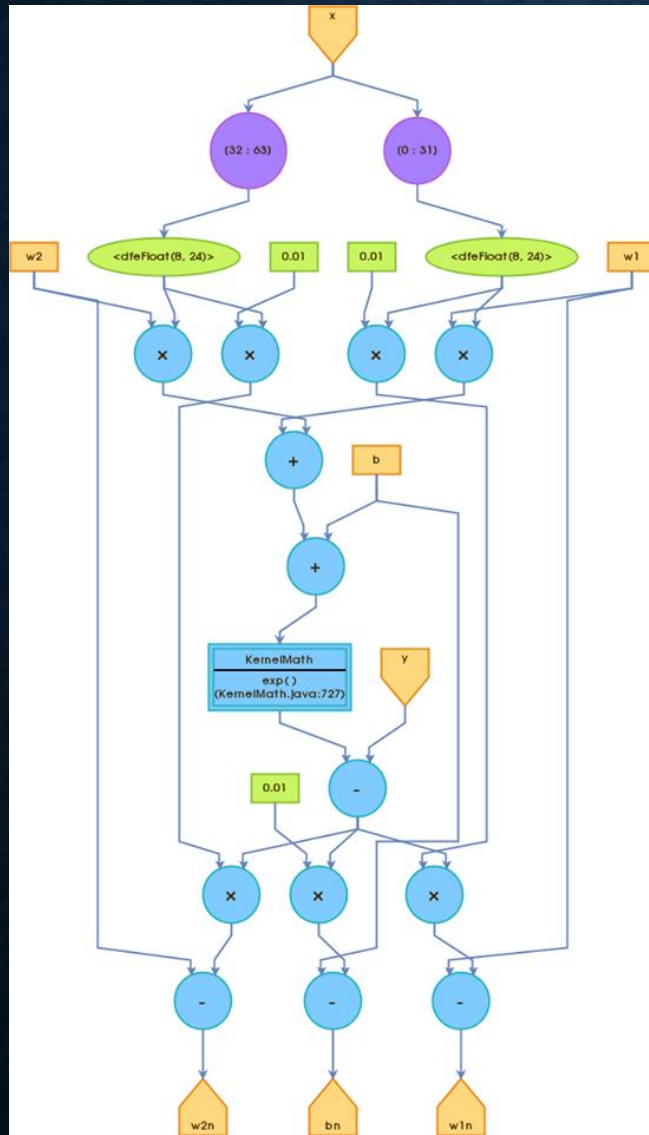
$$\text{Log-odds of passing exam} = 1.5046 \cdot \text{Hours} - 4.0777 = 1.5046 \cdot (\text{Hours} - 2.71)$$

$$\text{Odds of passing exam} = \exp(1.5046 \cdot \text{Hours} - 4.0777) = \exp(1.5046 \cdot (\text{Hours} - 2.71))$$

$$\text{Probability of passing exam} = \frac{1}{1 + \exp(-(1.5046 \cdot \text{Hours} - 4.0777))}$$

Hours of study	Passing exam		
	Log-odds	Odds	Probability
1	-2.57	$0.076 \approx 1:13.1$	0.07
2	-1.07	$0.34 \approx 1:2.91$	0.26
3	0.44	1.55	0.61
4	1.94	6.96	0.87
5	3.45	31.4	0.97

GRAFOVI



CPU KOD

CPUCode/MovingAverageWeightedCpuCode.c

Save

Undo

Redo

Settings

```
1  /**
2   * MaxFile name: MovingAverageWeighted
3   */
4  #include "Maxfiles.h"          // Includes .max files
5  #include <MaxSLiCInterface.h>  // Simple Live CPU interface
6
7  //Ulazne vrednosti, kao iz tabele
8  float X[16] = { 1, 1,
9                 1, 3,
10                5, 3,
11                8, 4,
12                1, 3,
13                9, 4,
14                10, 2,
15                1, 8
16 };
17
18 //Definisemo vektore za koeficijente
19 float Y[8] = { 0, 1, 0, 0, 1, 1, 0, 1 };
20 float W1_vec[8];
21 float W2_vec[8];
22 float B_vec[8];
23
```

CPUCode/MovingAverageWeightedCpuCode.c

Save

Undo

Redo

Settings

```
23
24 int main()
25 {
26     //Inicijalizacija
27     float W1 = 0;
28     float W2 = 0;
29     float B = 0;
30
31     printf("Running DFE\n");
32
33     for (int j = 0; j < 10; ++j) {
34         // Treniramo jedan ceo skup podataka
35         float weights[] = {W1, W2, B};
36         MovingAverageWeighted_weighted(4, weights, X, Y, W1_vec, W2_vec, B_vec);
37
38         // Slazemo rezultate na procesor
39         float W1sum = 0;
40         float W2sum = 0;
41         float Bsum = 0;
42
43         for (int i = 0; i < 8; ++i) {
44             W1sum += W1_vec[i];
45             W2sum += W2_vec[i];
46             Bsum += B_vec[i];
47         }
48
49         W1 = W1sum / 8.0;
50         W2 = W2sum / 8.0;
51         B = Bsum / 8.0;
52
53         // Proveravamo gresku na procesoru i ispisujemo prosečnu
54         float avg_err = 0;
55         for (int i = 0; i < 8; ++i) {
56             float err = pow(exp(W1*X[2*i] + W2*X[2*i + 1] + B) - Y[i], 2.0);
57             avg_err += err;
58         }
59
60         printf("AvgErr = %f\n", avg_err / 8.0);
61     }
62
63     return 0;
64 }
65
66
67
68
```

KERNERL KOD

EngineCode/src/movingaverageweighted/MovingAverageWeightedKernel.maxj

Save

Undo

Redo

Settings

```
1  /**
2   * MaxFile name: MovingAverageWeighted
3   */
4   package movingaverageweighted;
5
6   import com.maxeler.maxcompiler.v2.kernelcompiler.Kernel;
7   import com.maxeler.maxcompiler.v2.kernelcompiler.KernelParameters;
8   import com.maxeler.maxcompiler.v2.kernelcompiler.types.base.DFEVar;
9   import com.maxeler.maxcompiler.v2.kernelcompiler.types.composite.DFEVector;
10  import com.maxeler.maxcompiler.v2.kernelcompiler.types.composite.DFEVectorType;
11  import com.maxeler.maxcompiler.v2.kernelcompiler.types.base.DFEType;
12  import com.maxeler.maxcompiler.v2.kernelcompiler.stdlib.KernelMath;
13
14
15  class MovingAverageWeightedKernel extends Kernel {
16
17      MovingAverageWeightedKernel(KernelParameters parameters) {
18          super(parameters);
19
20          final DFEType type = dfeFloat(8, 24);
21          final int features = 2;
22          DFEVectorType<DFEVar> vectorType = new DFEVectorType<DFEVar>(type, features);
23
24          DFEVector<DFEVar> x = io.input("x", vectorType);
25          DFEVar y = io.input("y", type);
26          DFEVar w1 = io.scalarInput("w1", type);
27
28          DFEVar w2 = io.scalarInput("w2", type);
29          DFEVar b = io.scalarInput("b", type);
30          //DFEVar s = io.scalarInput("size", type);
31
32          DFEVar z = (w1*x[0] + w2*x[1] + b);
33          DFEVar p = KernelMath.exp(z);
34
35          DFEVar err = p - y;
36          DFEVar w1_new = w1 - 0.01 * x[0] * err;
37          DFEVar w2_new = w2 - 0.01 * x[1] * err;
38          DFEVar b_new = b - 0.01 * err;
39
40          io.output("w1n", w1_new, type);
41          io.output("w2n", w2_new, type);
42          io.output("bn", b_new, type);
43      }
```

