

# Funkcija Greške i Optimizacija

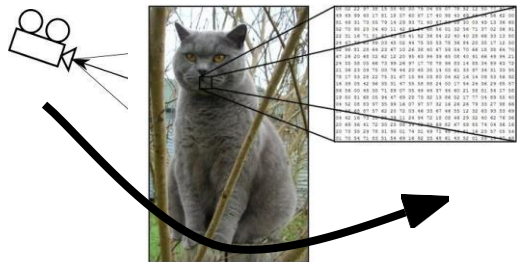
Predavač: Aleksandar Kovačević

Slajdovi preuzeti sa CS 231n, Stanford

<http://cs231n.stanford.edu/>

# Sa prošlog predavanja ... Izazovi kod klasifikacije slika

Pozicija kamere



Osvetljenje



Deformacija



Okluzija



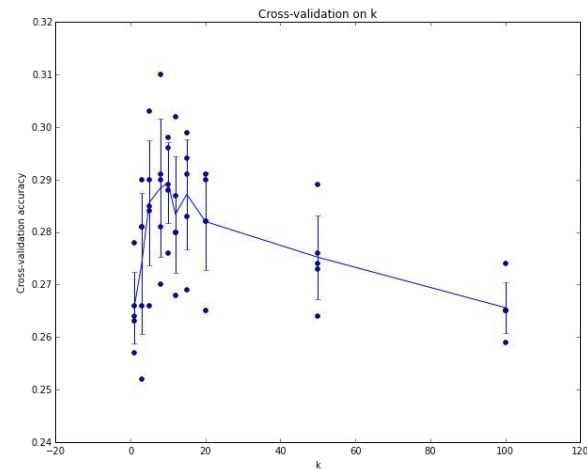
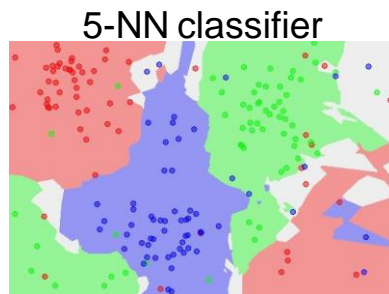
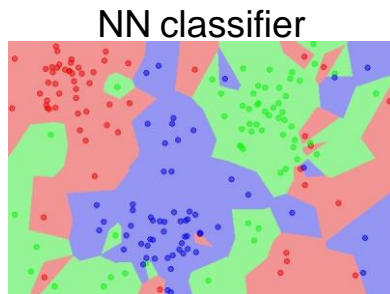
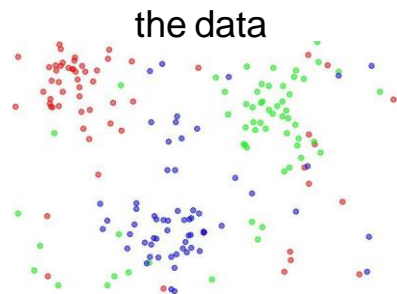
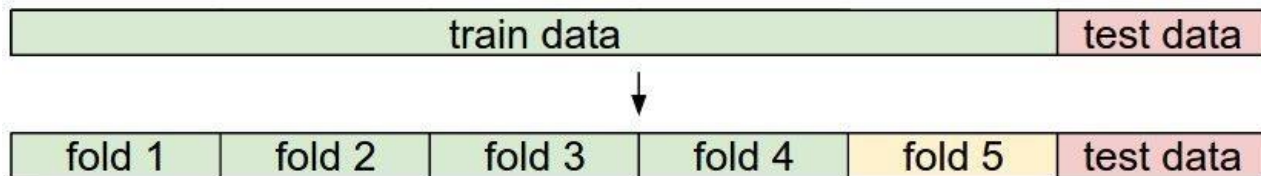
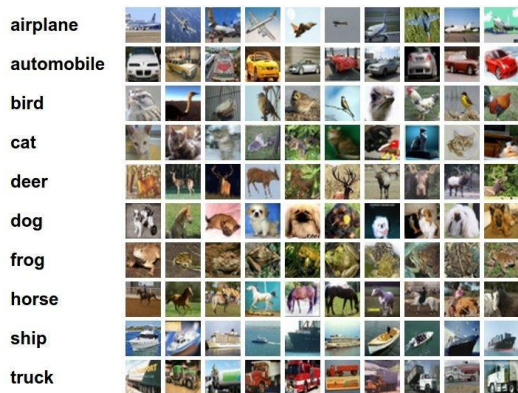
Pretrpana pozadina



Varijabilnost unutra klase



# Sa prošlog predavanja... nadgledano učenje, kNN



# Sa prošlog predavanja ... Linearni klasifikator

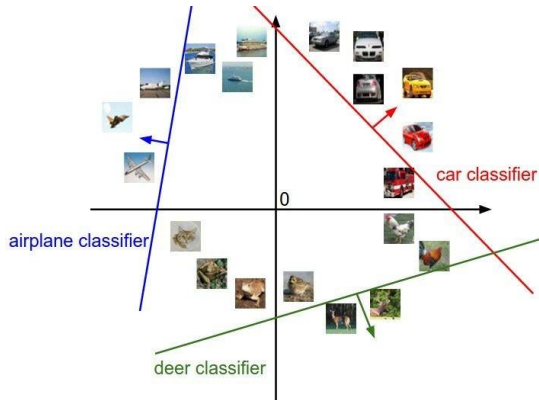
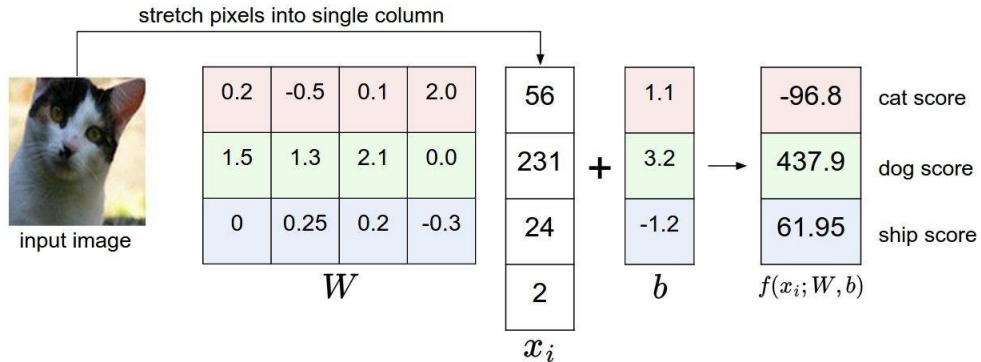


[32x32x3]  
niz od 3072 brojeva

slika      parametri

$$f(\mathbf{x}, \mathbf{W})$$

10 brojeva koji su skorovi klasa



# Sa prošlog predavanja ... Danas: Funkcija greške /Optimizacija



airplane	-3.45	-0.51	3.42
automobile	-8.87	<b>6.04</b>	4.64
bird	0.09	5.31	2.65
cat	<b>2.9</b>	-4.22	5.1
deer	4.48	-4.19	2.64
dog	8.02	3.58	5.55
frog	3.78	4.49	<b>-4.34</b>
horse	1.06	-4.37	-1.5
ship	-0.36	-2.09	-4.79
truck	-0.72	-2.93	6.14

## TODO:

1. Definisanje funkcije greške (***loss function***) koja kvantifikuje kvalitet skorova za slike iz obučavajućeg skupa
2. Nađemo način da automatski odredimo  $W$  i  $b$  tako da minimizujemo funkciju greške. (**optimizacija**)

Na primer: 3 slike u obučavajućem skupu, 3 klase.  
Za neko  $W$  skorovi  $f(x, W) = Wx$  su:



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>

Na primer: 3 slike u obučavajućem skupu, 3 klase.  
Za neko  $W$  skorovi  $f(x, W) = Wx$  su:



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>

## Fukcija greške modela SVM za više klase:

Za primer  $(x_i, y_i)$

gde je  $x_i$  slika

gde je  $y_i$  klasa (ceo broj) ,

Gde se skor računa kao:  $s = f(x_i, W)$

funkcija greške za SVM je oblika:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$



Na primer: 3 slike u obučavajućem skupu, 3 klase.  
Za neko  $W$  skorovi  $f(x, W) = Wx$  su:



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>

## Fukcija greške modela SVM za više klase:

Za primer  $(x_i, y_i)$

gde je  $x_i$  slika

gde je  $y_i$  klasa (ceo broj) ,

Gde se skor računa kao:  $s = f(x_i, W)$

funkcija greške za SVM je oblika:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$s_j - s_{y_i} + 1 \leq 0$$

$$-s_{y_i} \leq -s_j - 1$$

$$s_{y_i} \geq s_j + 1$$



Na primer: 3 slike u obučavajućem skupu, 3 klase.  
Za neko  $W$  skorovi  $f(x, W) = Wx$  su:



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>

Greške:

## Funkcija greške modela SVM za više klasa:

Za primer  $(x_i, y_i)$

gde je  $x_i$  slika

gde je  $y_i$  klasa (ceo broj) ,

Gde se skor računa kao:  $s = f(x_i, W)$

funkcija greške za SVM je oblika:

$$\begin{aligned} L_i &= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \\ &= \max(0, 5.1 - 3.2 + 1) \\ &\quad + \max(0, -1.7 - 3.2 + 1) \\ &= \max(0, 2.9) + \max(0, -3.9) \\ &= 2.9 + 0 \\ &= 2.9 \end{aligned}$$

Na primer: 3 slike u obučavajućem skupu, 3 klase.  
 Za neko  $W$  skorovi  $f(x, W) = Wx$  su:



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
<b>Greške:</b>	2.9	0	

## Fukcija greške modela SVM za više klasa:

Za primer  $(x_i, y_i)$

gde je  $x_i$  slika

gde je  $y_i$  klasa (ceo broj) ,

Gde se skor računa kao:  $s = f(x_i, W)$

funkcija greške za SVM je oblika:

$$\begin{aligned}
 L_i &= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \\
 &= \max(0, 1.3 - 4.9 + 1) \\
 &\quad + \max(0, 2.0 - 4.9 + 1) \\
 &= \max(0, -2.6) + \max(0, -1.9) \\
 &= 0 + 0 \\
 &= 0
 \end{aligned}$$

Na primer: 3 slike u obučavajućem skupu, 3 klase.  
 Za neko  $W$  skorovi  $f(x, W) = Wx$  su:



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
<b>Greške:</b>	2.9	0	10.9

## Fukcija greške modela SVM za više klasa:

Za primer  $(x_i, y_i)$

gde je  $x_i$  slika

gde je  $y_i$  klasa (ceo broj) ,

Gde se skor računa kao:  $s = f(x_i, W)$

funkcija greške za SVM je oblika:

$$\begin{aligned}
 L_i &= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \\
 &= \max(0, 2.2 - (-3.1) + 1) \\
 &\quad + \max(0, 2.5 - (-3.1) + 1) \\
 &= \max(0, 5.3) + \max(0, 5.6) \\
 &= 5.3 + 5.6 \\
 &= 10.9
 \end{aligned}$$

Na primer: 3 slike u obučavajućem skupu, 3 klase.  
 Za neko  $W$  skorovi  $f(x, W) = Wx$  su:



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
<b>Greške:</b>	2.9	0	10.9

## Fukcija greške modela SVM za više klasa:

Za primer  $(x_i, y_i)$

gde je  $x_i$  slika

gde je  $y_i$  klasa (ceo broj) ,

Gde se skor računa kao:  $s = f(x_i, W)$

funkcija greške za SVM je oblika:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Greška za sve tri slike je prosek  
grešaka:

$$L = \frac{1}{N} \sum_{i=1}^N L_i$$

$$L = (2.9 + 0 + 10.9)/3 \\ = 4.6$$

Na primer: 3 slike u obučavajućem skupu, 3 klase.  
Za neko  $W$  skorovi  $f(x, W) = Wx$  su:



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
<b>Greške:</b>	2.9	0	10.9

## Funkcija greške modela SVM za više klase:

Za primer  $(x_i, y_i)$

gde je  $x_i$  slika

gde je  $y_i$  klasa (ceo broj) ,

Gde se skor računa kao:  $s = f(x_i, W)$

funkcija greške za SVM je oblika:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Šta bi bilo ako bi  $L_i$   
računali za sve  
klase tj. i za  $j=y_i$ ?

Na primer: 3 slike u obučavajućem skupu, 3 klase.  
 Za neko  $W$  skorovi  $f(x, W) = Wx$  su:



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
<b>Greške:</b>	2.9	0	10.9

## Fukcija greške modela SVM za više klasa:

Za primer  $(x_i, y_i)$

gde je  $x_i$  slika

gde je  $y_i$  klasa (ceo broj) ,

Gde se skor računa kao:  $s = f(x_i, W)$

funkcija greške za SVM je oblika:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Šta bi bilo ako bi  
korsitili:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)^2$$

Na primer: 3 slike u obučavajućem skupu, 3 klase.  
Za neko  $W$  skorovi  $f(x, W) = Wx$  su:



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
<b>Greške:</b>	2.9	0	10.9

## Fukcija greške modela SVM za više klasa:

Za primer  $(x_i, y_i)$

gde je  $x_i$  slika

gde je  $y_i$  klasa (ceo broj) ,

Gde se skor računa kao:  $s = f(x_i, W)$

funkcija greške za SVM je oblika:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Koje su min i max  
vrednosti za funkciju  
greške?



Na primer: 3 slike u obučavajućem skupu, 3 klase.  
Za neko  $W$  skorovi  $f(x, W) = Wx$  su:



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
<hr/>			
Greške:	2.9	0	10.9

## Fukcija greške modela SVM za više klasa:

Za primer  $(x_i, y_i)$

gde je  $x_i$  slika

gde je  $y_i$  klasa (ceo broj) ,

Gde se skor računa kao:  $s = f(x_i, W)$

funkcija greške za SVM je oblika:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Obično se vrednosti  $W$   
inicijalizuju na jako male  
brojeve  $\sim 0$ .

Koje su tada vrednosti  
funkcije greške?

numpy kod za SVM primer:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

```
def L_i_vectorized(x, y, W):  
    scores = W.dot(x)  
    margins = np.maximum(0, scores - scores[y] + 1)  
    margins[y] = 0  
    loss_i = np.sum(margins)  
    return loss_i
```

$$f(x, W) = Wx$$

$$L = \frac{1}{N} \sum_{i=1}^N \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1)$$

Naša funkcija greške ima bag:

$$f(x, W) = Wx$$

$$L = \frac{1}{N} \sum_{i=1}^N \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1)$$



Naša funkcija greške ima bag:

$$f(x, W) = Wx$$

$$L = \frac{1}{N} \sum_{i=1}^N \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1)$$



Recimo da smo pronašli  $W$  tako da je  $L = 0$ .  
Da li je to  $W$  jedinstveno?

Na primer: 3 slike u obučavajućem skupu, 3 klase.  
Za neko  $W$  skorovi  $f(x, W) = Wx$  su:



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
Greške:	2.9	<b>0</b>	

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

**Ranije:**

$$\begin{aligned}
 &= \max(0, 1.3 - 4.9 + 1) \\
 &\quad + \max(0, 2.0 - 4.9 + 1) \\
 &= \max(0, -2.6) + \max(0, -1.9) \\
 &= 0 + 0 \\
 &= 0
 \end{aligned}$$

**Sa duplo većim  $W$ :**

$$\begin{aligned}
 &= \max(0, 2.6 - 9.8 + 1) \\
 &\quad + \max(0, 4.0 - 9.8 + 1) \\
 &= \max(0, -6.2) + \max(0, -4.8) \\
 &= 0 + 0 \\
 &= 0
 \end{aligned}$$

# Regularizacija težina

lambda = jačina regularizacije  
(hiper-parametar)

$$L = \frac{1}{N} \sum_{i=1}^N \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1) + \lambda R(W)$$

Tipično se koristi:

**L2 regularizacija**

**L1 regularizacija**

Elastic net (L1 + L2)

Dropout (kasnije tokom kursa)

$$R(W) = \sum_k \sum_l W_{k,l}^2$$

$$R(W) = \sum_k \sum_l |W_{k,l}|$$

$$R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$$



## L2 regularizacija: motivacija

$$x = [1, 1, 1, 1]$$

$$w_1 = [1, 0, 0, 0]$$

$$w_2 = [0.25, 0.25, 0.25, 0.25]$$

$$w_1^T x = w_2^T x = 1$$

# Softmax Klasifikator (Multinomialna Logistička Regresija)



cat	<b>3.2</b>
car	5.1
frog	-1.7

# Softmax Klasifikator (Multinomialna Logistička Regresija)



skorovi = nenormalizovane log verovatnoće klasa  
(nenormalizovane znači da se ne sabiraju na 1)

$$s = f(x_i; W)$$

cat	3.2
car	5.1
frog	-1.7

# Softmax Klasifikator (Multinomialna Logistička Regresija)



skorovi = nenormalizovane log verovatnoće klasa

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}} \quad s = f(x_i; W)$$

cat	<b>3.2</b>
car	5.1
frog	-1.7

# Softmax Klasifikator (Multinomialna Logistička Regresija)



skorovi = nenormalizovane log verovatnoće klasa

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}} \quad s = f(x_i; W)$$

cat	<b>3.2</b>
car	5.1
frog	-1.7

Softmax funkcija

# Softmax Klasifikator (Multinomialna Logistička Regresija)



skorovi = nenormalizovane log verovatnoće klasa

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}} \quad s = f(x_i; W)$$

Maksimizujemo log verovatnost (log likelihood) – obično se minimizuje negativna log verovatnost:

$$L_i = -\log P(Y = y_i|X = x_i)$$

cat	<b>3.2</b>
car	<b>5.1</b>
frog	<b>-1.7</b>

# Softmax Klasifikator (Multinomialna Logistička Regresija)



cat	<b>3.2</b>
car	<b>5.1</b>
frog	<b>-1.7</b>

skorovi = nenormalizovane log verovatnoće klasa

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}} \quad s = f(x_i; W)$$

Maksimizujemo log verovatnost (log likelihood) – obično se minimizuje negativna log verovatnost:

$$L_i = -\log P(Y = y_i|X = x_i)$$

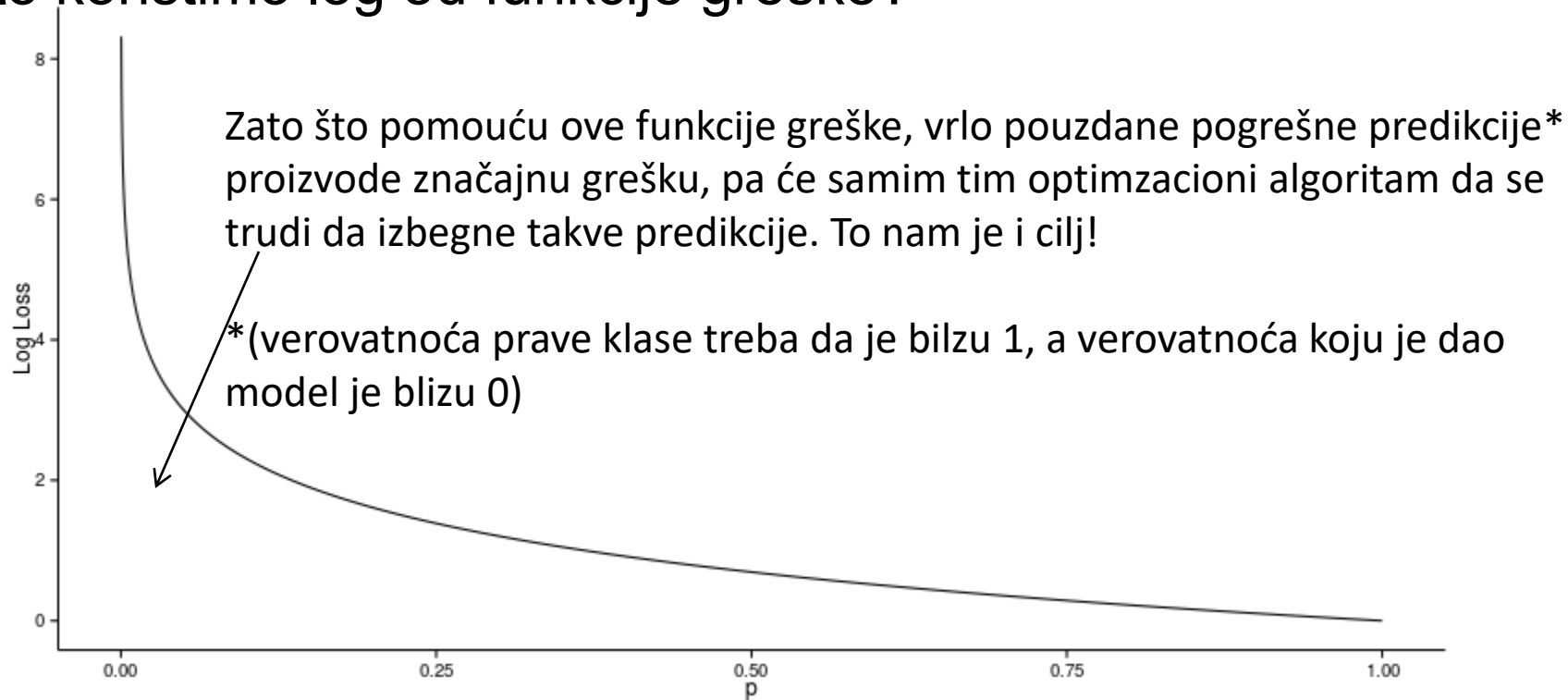
---

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$



# Softmax Klasifikator (Multinomialna Logistička Regresija)

## Zašto koristimo log od funkcije greške?



# Softmax Klasifikator (Multinomialna Logistička Regresija)



$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

cat	<b>3.2</b>
car	5.1
frog	-1.7

nenormalizovane log verovatnoće

# Softmax Klasifikator (Multinomialna Logistička Regresija)



$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

nenormalizovane verovatnoće

cat	<b>3.2</b>	exp →	<b>24.5</b>
car	5.1		164.0
frog	-1.7		0.18

nenormalizovane log verovatnoće

# Softmax Klasifikator (Multinomialna Logistička Regresija)



$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

nenormalizovane verovatnoće

cat	<b>3.2</b>	exp	<b>24.5</b>	normalizacija	<b>0.13</b>
car	5.1		164.0		0.87
frog	-1.7		0.18		0.00

nenormalizovane log verovatnoće

verovatnoće

# Softmax Klasifikator (Multinomialna Logistička Regresija)



$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

nenormalizovane verovatnoće

cat

3.2

car

5.1

frog

-1.7

exp

24.5

164.0

0.18

normalizacija

0.13

0.87

0.00

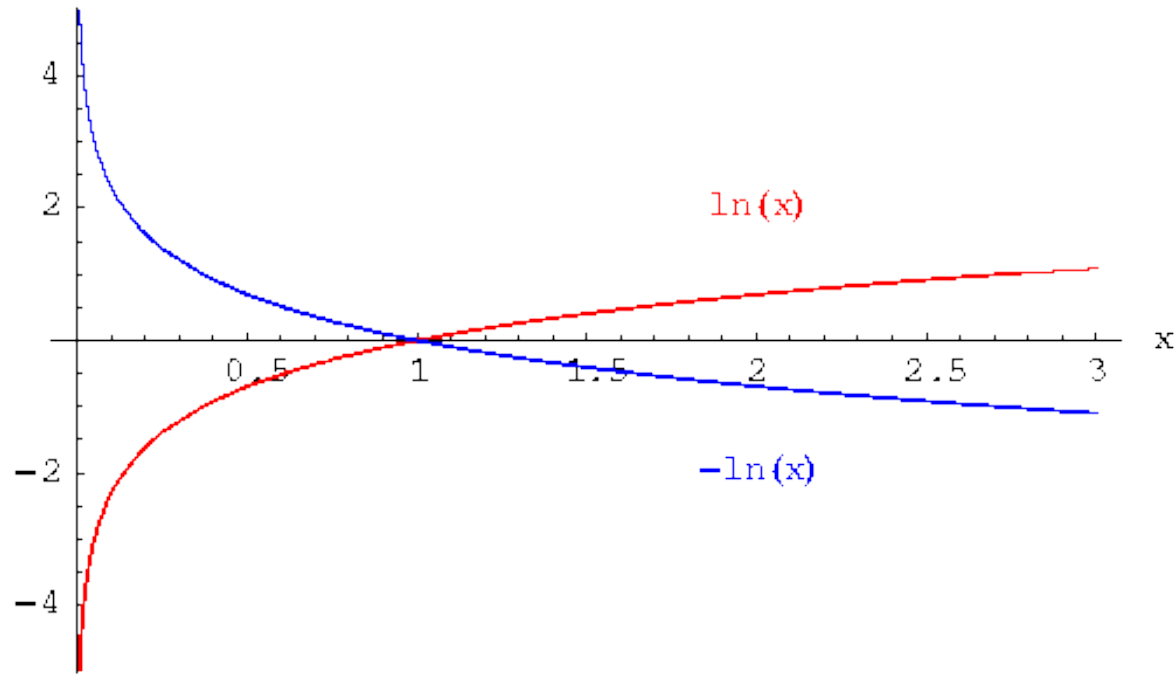
$$\rightarrow L_i = -\log(0.13) = 0.89$$

nenormalizovane log verovatnoće

verovatnoće

# Softmax Klasifikator (Multinomialna Logistička Regresija)

log vs. -log



# Softmax Klasifikator (Multinomialna Logistička Regresija)



$$L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{s_j}}\right)$$

Koje su moguće min i max vrednosti za  $L_i$ ?

nenormalizovane verovatnoće

cat

3.2

car

5.1

frog

-1.7

exp

24.5

164.0

0.18

normalizacija

0.13

0.87

0.00

$$\rightarrow L_i = -\log(0.13) = 0.89$$

nenormalizovane log verovatnoće

verovatnoće



# Softmax Klasifikator (Multinomialna Logistička Regresija)



$$L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{s_j}}\right)$$

nemormalizovane verovatnoće

Obično se vrednosti  $W$  inicijalizuju na jako male brojeve  $\sim 0$ .  
Koje su tada vrednosti funkcije greške?

cat  
car  
frog

$\sim 0$

$\sim 0$

$\sim 0$

exp

$\sim 1$

$\sim 1$

$\sim 1$

normalizacija

$\sim 1/(1+1+1)$

$\sim 1/(1+1+1)$

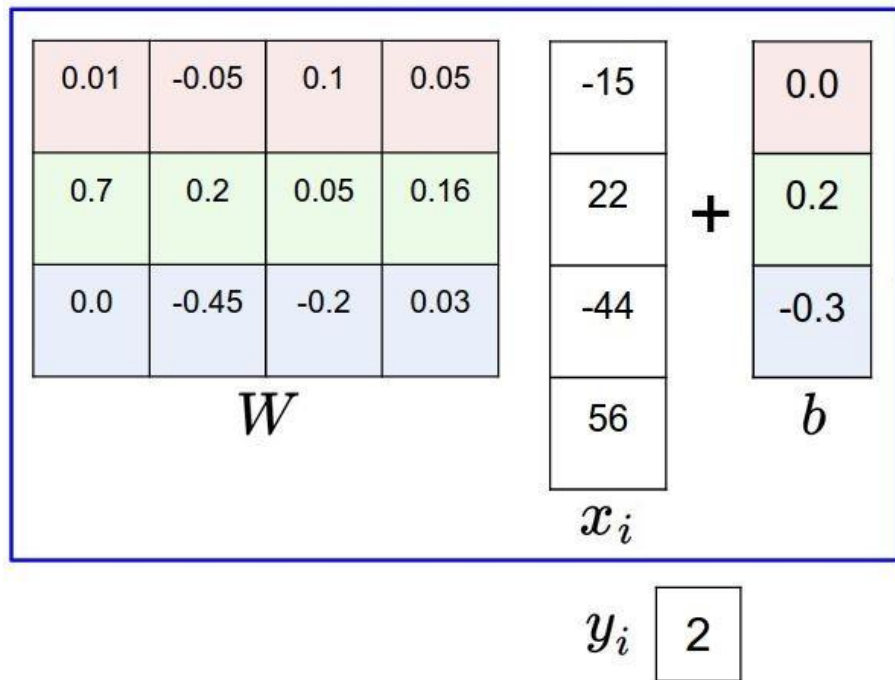
$\sim 1/(1+1+1)$

verovatnoće

$$L_i = -\log(\sim 1/3)$$

nemormalizovane log verovatnoće

matrix multiply + bias offset



hinge loss (SVM)

-2.85
0.86
0.28

$$\begin{aligned} &\max(0, -2.85 - 0.28 + 1) + \\ &\max(0, 0.86 - 0.28 + 1) \\ &= \\ &\mathbf{1.58} \end{aligned}$$

cross-entropy loss (Softmax)

-2.85
0.86
0.28

$\xrightarrow{\text{exp}}$

0.058
2.36
1.32

$\xrightarrow{\text{normalize}}$   
(to sum to one)

0.016
0.631
0.353

$$\begin{aligned} &-\log(0.353) \\ &= \\ &\mathbf{0.452} \end{aligned}$$

# Softmax vs. SVM

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

# Softmax vs. SVM

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

recimo da imamo skorove:

[10, -2, 3]

[10, 9, 9]

[10, -100, -100]

$$y_i = 0$$

Šta se dešava sa obe  
funkcije greške ako uzmemo  
jednu tačku i malo je  
pomeramo u prostoru?

# Optimizacija

# Rezime

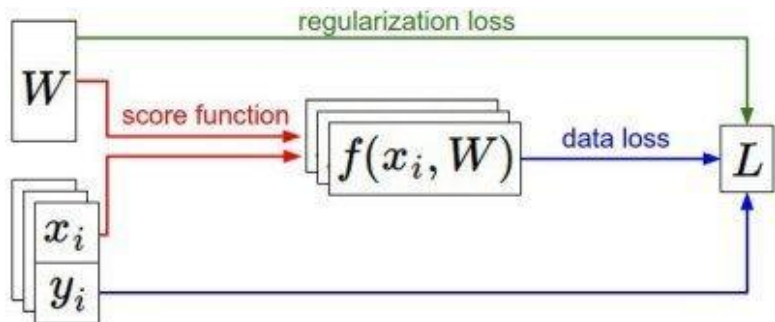
- Imamo skup parova (x,y)
- Imamo **skor funkciju**:
- Imamo **funkciju greške**:

$$s = f(x; W) \stackrel{\text{npr.}}{=} Wx$$

$$L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{sj}}\right) \quad \text{Softmax}$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \quad \text{SVM}$$

$$L = \frac{1}{N} \sum_{i=1}^N L_i + R(W) \quad \text{Greška + Regularizacija}$$



# Optimizacija - Strategija #1: Random pretraga

```
# assume X_train is the data where each column is an example (e.g. 3073 x 50,000)
# assume Y_train are the labels (e.g. 1D array of 50,000)
# assume the function L evaluates the loss function

bestloss = float("inf") # Python assigns the highest possible float value
for num in xrange(1000):
    W = np.random.randn(10, 3073) * 0.0001 # generate random parameters
    loss = L(X_train, Y_train, W) # get the loss over the entire training set
    if loss < bestloss: # keep track of the best solution
        bestloss = loss
        bestW = W
    print 'in attempt %d the loss was %f, best %f' % (num, loss, bestloss)

# prints:
# in attempt 0 the loss was 9.401632, best 9.401632
# in attempt 1 the loss was 8.959668, best 8.959668
# in attempt 2 the loss was 9.044034, best 8.959668
# in attempt 3 the loss was 9.278948, best 8.959668
# in attempt 4 the loss was 8.857370, best 8.857370
# in attempt 5 the loss was 8.943151, best 8.857370
# in attempt 6 the loss was 8.605604, best 8.605604
# ... (truncated: continues for 1000 lines)
```

## Šta dobijamo na test skupu...

```
# Assume X_test is [3073 x 10000], Y_test [10000 x 1]  
scores = Wbest.dot(Xte_cols) # 10 x 10000, the class scores for all test examples  
# find the index with max score in each column (the predicted class)  
Yte_predict = np.argmax(scores, axis = 0)  
# and calculate accuracy (fraction of predictions that are correct)  
np.mean(Yte_predict == Yte)  
# returns 0.1555
```

15.5% tačnost! nije loše!  
(State-of-the-Art, SOTA je ~95%)







## Strategija #2: **Pratimo nagib (slope)**

Kod funkcije jedne promenljive, izvod funkcije dat je sa:

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

Kad imamo više promenljivih (atributa), **gradijent** je vektor parcijalnih izvoda po svim atributima.

**W:**

[0.34,  
-1.11,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**greška 1.25347**

**gradijent dW:**

[?,  
?,  
?,  
?,  
?,  
?,  
?,  
?,  
?,  
?,...]

**W:**

[0.34,  
-1.11,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**greška 1.25347**

**W + h** (za prvi atribut):

[0.34 + **0.0001**,  
-1.11,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**greška 1.25322**

**gradijent dW:**

[?,  
?,  
?,  
?,  
?,  
?,  
?,  
?,  
?,...]

**W:**

[0.34,  
-1.11,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**greška 1.25347**

**W + h** (za prvi atribut):

[0.34 + **0.0001**,  
-1.11,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**greška 1.25322**

**gradijent dW:**

**[-2.5,**  
?,  
?,

$$(1.25322 - 1.25347)/0.0001 = -2.5$$

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

?,  
?,...]



**W:**

[0.34,  
-1.11,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**greška 1.25347**

**W + h** (drugi atribut):

[0.34,  
-1.11 + **0.0001**,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**greška 1.25353**

**gradijent dW:**

[-2.5,  
?,  
?,  
?,  
?,  
?,  
?,  
?,  
?,...]

**W:**

[0.34,  
-1.11,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**greška 1.25347**

**W + h** (drugi atribut):

[0.34,  
-1.11 + **0.0001**,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**greška 1.25353**

**gradijent dW:**

[-2.5,  
**0.6**,  
?,  
?,

$$(1.25353 - 1.25347)/0.0001 = 0.6$$

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

?,...]



**W:**

[0.34,  
-1.11,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**greška 1.25347**

**W + h** (treći atribut):

[0.34,  
-1.11,  
0.78 + **0.0001**,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**greška 1.25347**

**gradijent dW:**

[-2.5,  
0.6,  
?,  
?,  
?,  
?,  
?,  
?,  
?,...]

**W:**

[0.34,  
-1.11,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**greška 1.25347**

**W + h** (treći atribut):

[0.34,  
-1.11,  
0.78 + **0.0001**,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**greška 1.25347**

**gradijent dW:**

[-2.5,  
0.6,  
**0**,  
?,  
0

$$(1.25347 - 1.25347)/0.0001 = 0$$

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

?, ...]

Numerički izvod nije dobar pristup ovde. Greška je funkcija od  $W$ , pa možemo da koristimo analitički izvod.

$$L = \frac{1}{N} \sum_{i=1}^N L_i + \sum_k W_k^2$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$s = f(x; W) = Wx$$

treba  $\nabla_W L$   
nam

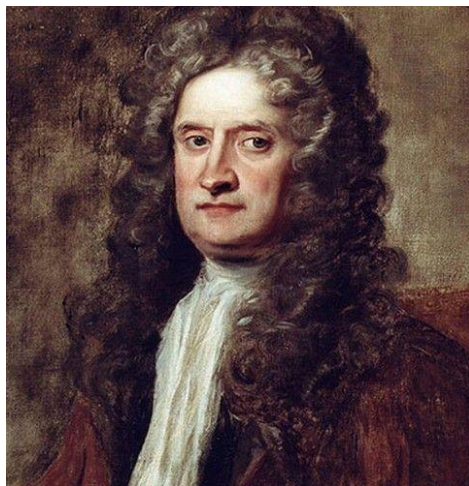
Numerički izvod nije dobar pristup ovde. Greška je funkcija od  $W$ , pa možemo da koristimo analitički izvod.

$$L = \frac{1}{N} \sum_{i=1}^N L_i + \sum_k W_k^2$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$s = f(x; W) = Wx$$

treba  $\nabla_W L$   
nam



Numerički izvod nije dobar pristup ovde. Greška je funkcija od  $W$ , pa možemo da koristimo analitički izvod.

$$L = \frac{1}{N} \sum_{i=1}^N L_i + \sum_k W_k^2$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$s = f(x; W) = Wx$$

treba nam  $\nabla_W L$

Analiza



Numerički izvod nije dobar pristup ovde. Greška je funkcija od  $W$ , pa možemo da koristimo analitički izvod.

$$L = \frac{1}{N} \sum_{i=1}^N L_i + \sum_k W_k^2$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$s = f(x; W) = Wx$$

$$\nabla_W L = \dots$$

**W:**

[0.34,  
-1.11,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**loss 1.25347**

**gradijent dW:**

dW = ...  
(neka funkcija  
od W)



[-2.5,  
0.6,  
0,  
0.2,  
0.7,  
-0.5,  
1.1,  
1.3,  
-2.1,...]

# Rezime:

- Numerički izvod: aproksimacija, spor, ali jednostavna formula
- Analitički izvod: tačan, brz, nije ga uvek lako naći, pa postoji mogućnost greške

=>

U praksi: Uvek koristimo analitički izvod, ali ga proverimo numerički. Ovo se zove provera gradijenta (***gradient check***).



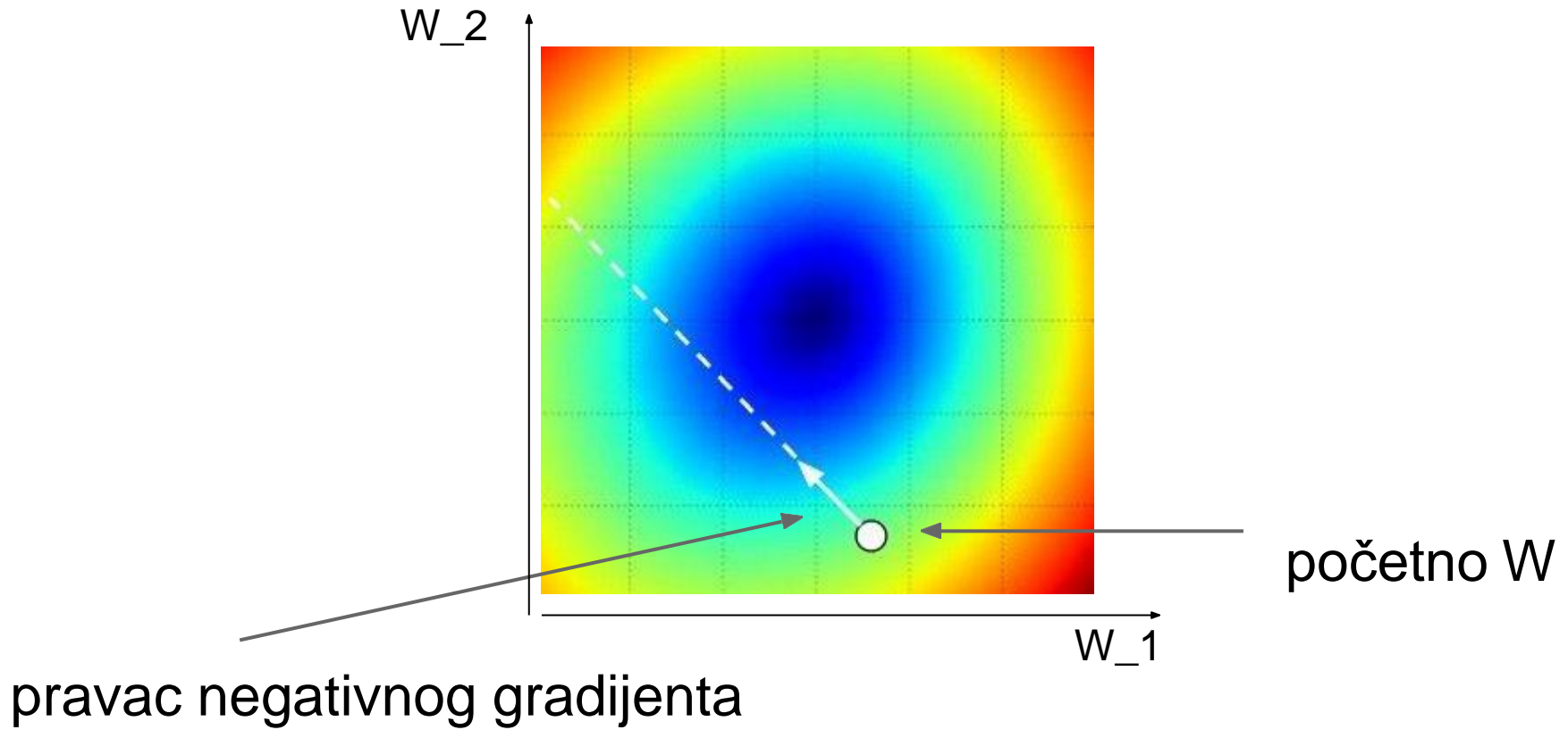
# Gradijetni Spust (*Gradient Descent*)

```
# Vanilla Gradient Descent
```

```
while True:
```

```
    weights_grad = evaluate_gradient(loss_fun, data, weights)
```

```
    weights += - step_size * weights_grad # perform parameter update
```



# Gradijetni Spust sa Mini-Podskupovima

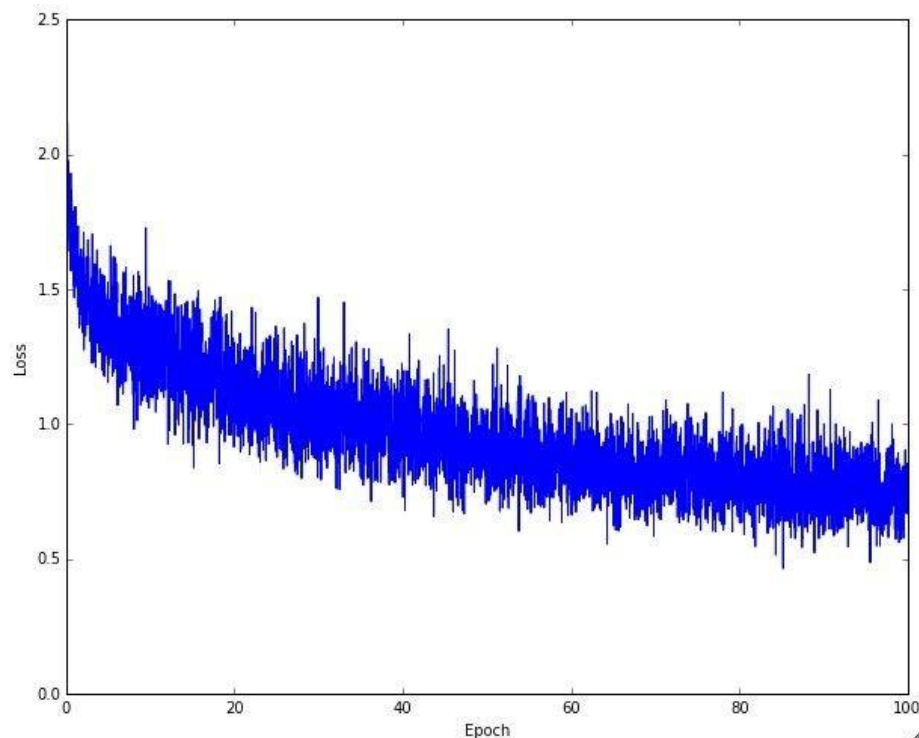
## *Mini-batch Gradient Descent*

- koristimo samo deo obučavajućeg skupa da izračunamo gradijent.

```
# Vanilla Minibatch Gradient Descent

while True:
    data_batch = sample_training_data(data, 256) # sample 256 examples
    weights_grad = evaluate_gradient(loss_fun, data_batch, weights)
    weights += - step_size * weights_grad # perform parameter update
```

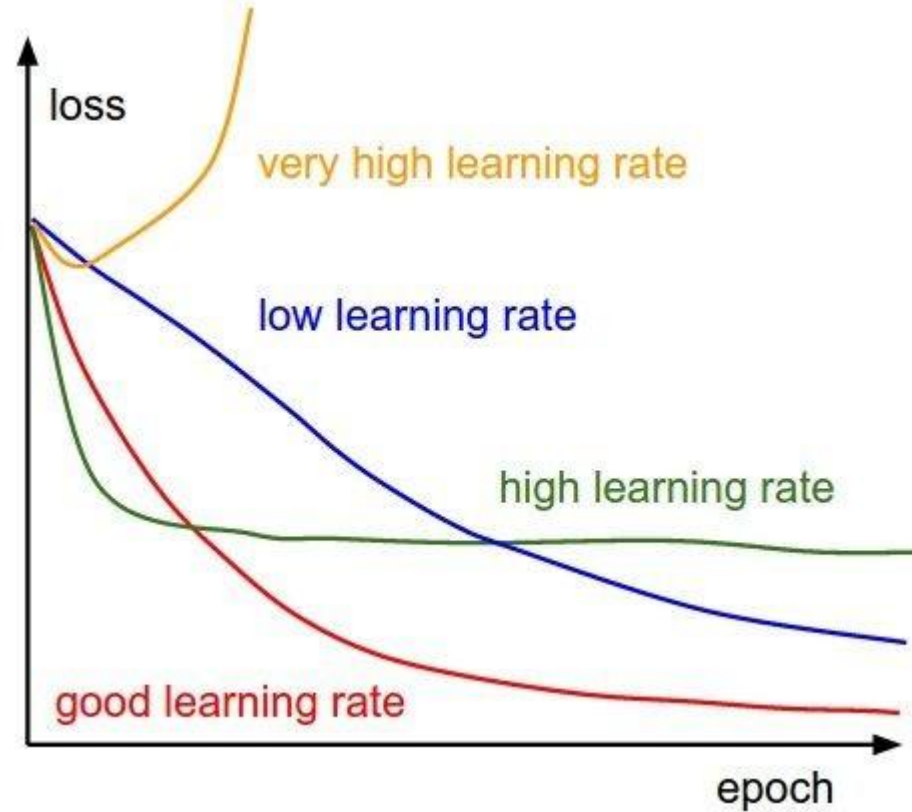
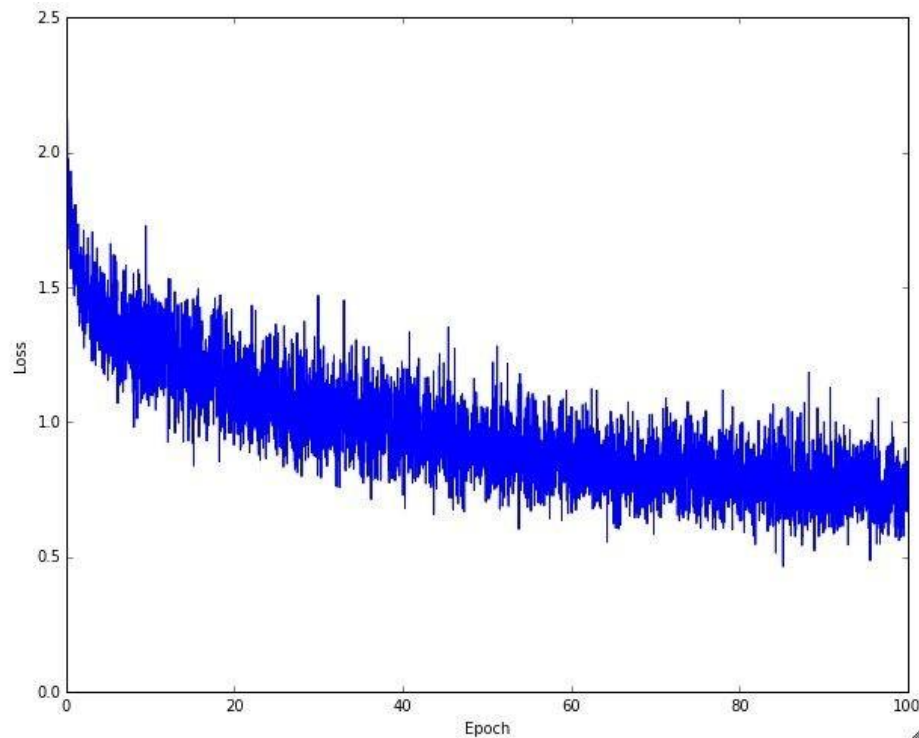
Tipične veličine podskupova su 32/64/128/256 primera



Primer optimizacije funkcije greške.

(Funkcija greške opada kroz iteracije - epohe.)

## Efekat veličine koraka (ili tempa učenja - *learning rate*)



# Gradijetni Spust sa Mini-Podskupovima

## *Mini-batch Gradient Descent*

- koristimo samo deo obučavajućeg skupa da izračunamo gradijent.

```
# Vanilla Minibatch Gradient Descent

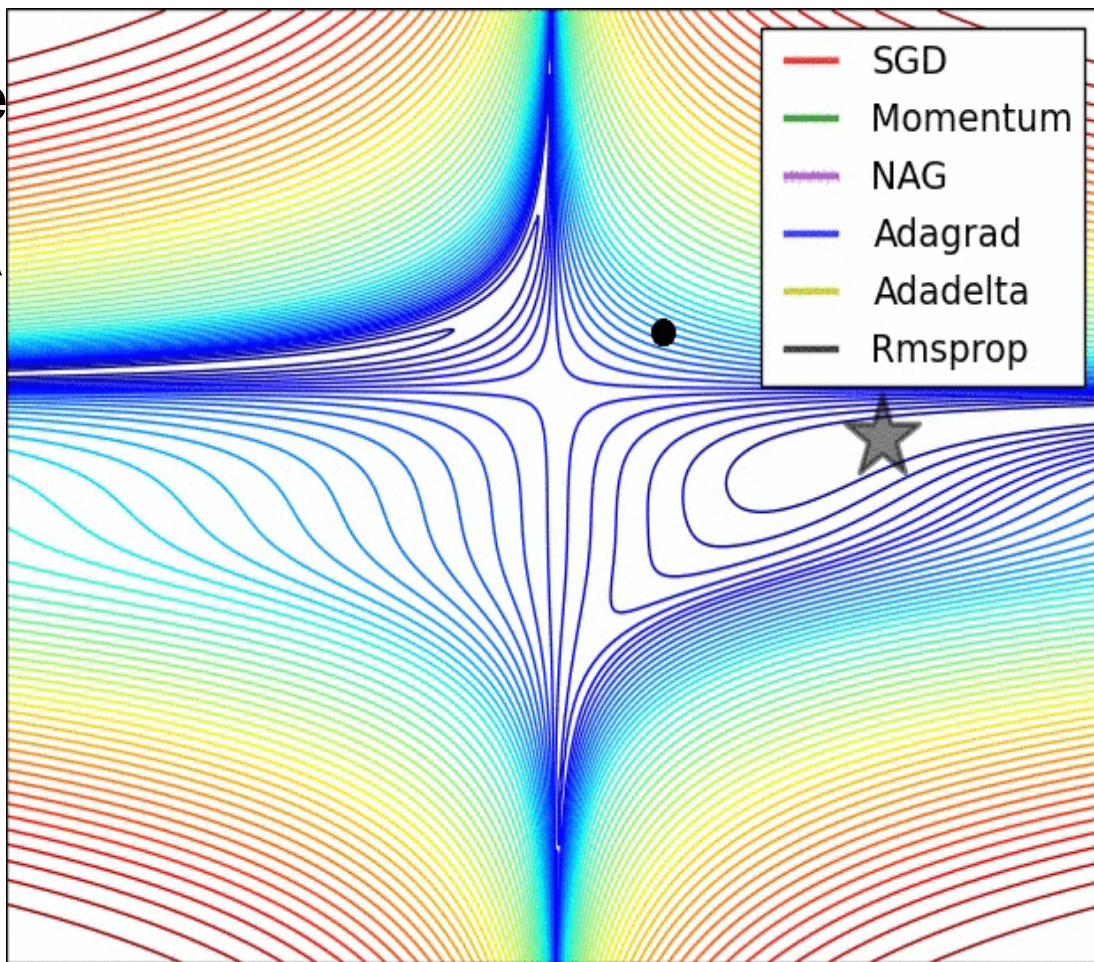
while True:
    data_batch = sample_training_data(data, 256) # sample 256 examples
    weights_grad = evaluate_gradient(loss_fun, data_batch, weights)
    weights += - step_size * weights_grad # perform parameter update
```

Tipične veličine podskupova su 32/64/128/256 primera

Kasnije tokom kursa radićemo novije metode za promenu težina (momentum, Adagrad, RMSProp, Adam, ...)



Ponašanje  
različnih  
formula za  
promene  
težina



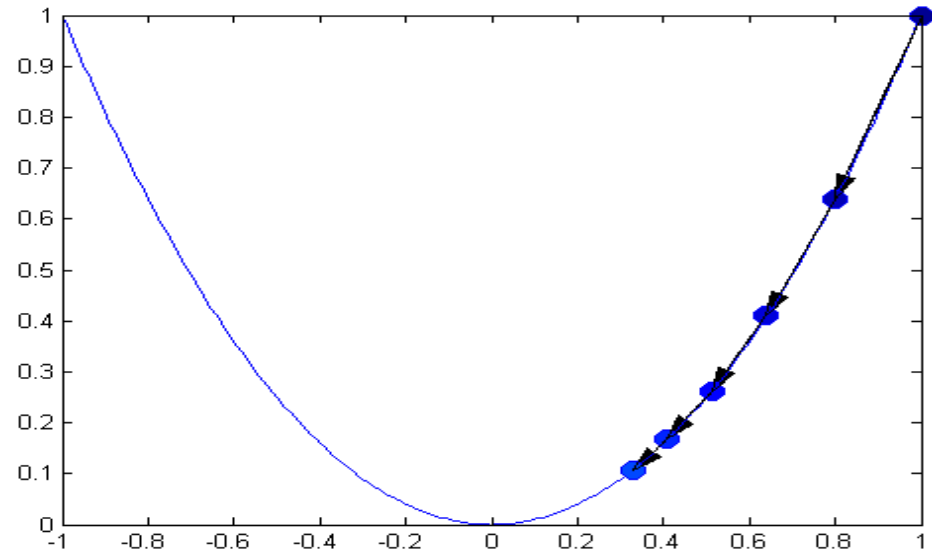
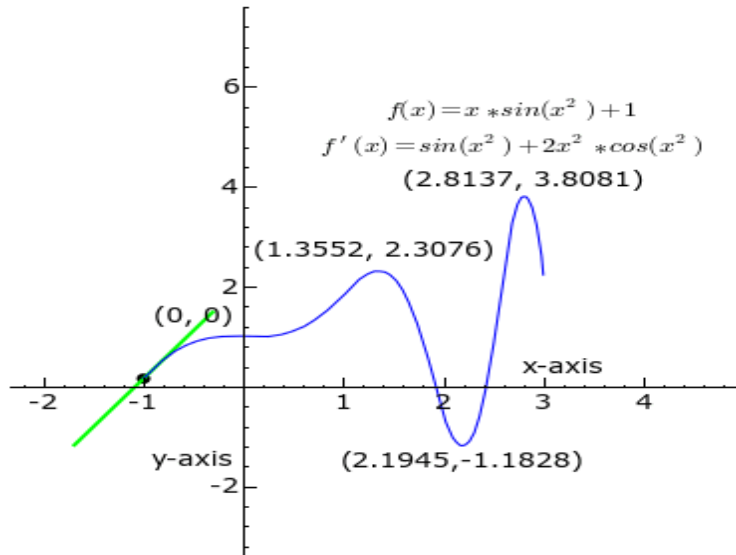
(image credits to Alec  
Radford)

# Gradijetni Spust ponavljanje



# Kako pronaći minimum funkcije?

- Izvod funkcije predstavlja nagib tangente na krivu funkcije
- Ideja: iterativno ćemo se pomerati ka minimumu
  - levo od minimuma: gradijent je negativan – pomeramo se u desno
  - desno od minimuma: gradijent je pozitivan – pomeramo se u levo



# Gradient descent

- **Optimizaciona tehnika:** data je (proizvoljna) funkcija  $J(\theta)$ . Želimo da pronademo  $\min_{\theta} J(\theta)$

Ulaz	<ul style="list-style-type: none"><li>• <math>J(\theta)</math> – funkcija koja se optimizuje</li><li>• <math>\theta_0</math> – početno rešenje</li><li>• <math>\alpha</math> – <i>learning rate</i> (veličina koraka)</li><li>• <i>maxIters</i> – maksimalan broj iteracija</li></ul>
Postupak	<pre>for t = 1, 2,..., maxIters:     for d = 1,2,...,D         • <math>\theta_d^{(t+1)} = \theta_d^{(t)} - \alpha \frac{\partial}{\partial \theta_d} J(\theta)</math></pre>
Izlaz	$\theta$ – tačka u kojoj funkcija $J(\theta)$ ima minimum

Gradijent –  
određuje smer  
pretrage

# Primena na linearnu regresiju

- Fitovanje modela: želimo da pronađemo parametre  $\theta$  za koje funkcija greške ima najmanju vrednost

$$J(\theta) = \frac{1}{2N} \sum_{i=1}^N (\theta_1 x^{(i)} + \theta_0 - y^{(i)})^2$$

- Ovo možemo uraditi primenom *gradient descent* algoritma:
  - Ponavljati do konvergencije:

$$\theta_d^{(t+1)} = \theta_d^{(t)} - \alpha \frac{\partial}{\partial \theta_d} J(\theta) \text{ za } d \in \{0,1\}$$

$$\frac{\partial J}{\partial \theta_0} = \frac{1}{N} \sum_{i=1}^N (\theta_1 x^{(i)} + \theta_0 - y^{(i)}) = \frac{1}{N} \sum_{i=1}^N (h_{\theta}(x^{(i)}) - y^{(i)}) \cdot 1$$

$$\frac{\partial J}{\partial \theta_1} = \frac{1}{N} \sum_{i=1}^N (\theta_1 x^{(i)} + \theta_0 - y^{(i)}) x^{(i)} = \frac{1}{N} \sum_{i=1}^N (h_{\theta}(x^{(i)}) - y^{(i)}) x^{(i)}$$

# Primena na linearnu regresiju

$$X = \begin{bmatrix} 1 & x^{(1)} \\ 1 & x^{(2)} \\ \dots & \dots \\ 1 & x^{(N)} \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} \quad h_{\theta}(x) = X\theta$$

$$\frac{\partial J}{\partial \theta_0} = \frac{1}{N} \sum_{i=1}^N (h_{\theta}(x^{(i)}) - y^{(i)}) \cdot 1$$

$$\frac{\partial J}{\partial \theta_1} = \frac{1}{N} \sum_{i=1}^N (h_{\theta}(x^{(i)}) - y^{(i)}) x^{(i)}$$

$$\rightarrow \frac{\partial J}{\partial \theta_d} = \frac{1}{N} \sum_{i=1}^N (h_{\theta}(x^{(i)}) - y^{(i)}) x_d^{(i)}$$

$$\theta_d^{(t+1)} = \theta_d^{(t)} - \frac{\alpha}{N} \sum_{i=1}^N (h_{\theta}(x^{(i)}) - y^{(i)}) x_d^{(i)}$$

# Gradient descent

- Ako bismo imali samo jedan primer u skupu podataka, pravilo za ažuriranje parametara je:

$$\theta_d^{(t+1)} = \theta_d^{(t)} - \alpha(h_\theta(x) - y)x_d$$

- Ovo pravilo se naziva **LMS (Least Mean Squares) update rule** ili **Widrow-Hoff learning rule**
- Magnituda promene je proporcionalna greški  $h_\theta(x) - y$ 
  - ako za dati primer naš prediktor daje vrednost veoma sličnu tačnoj vrednosti – nećemo mnogo menjati parametar
  - ako za dati primer prediktor ima veliku grešku – promena parametra će biti velika

# Batch GD vs. Stochastic GD

- *Batch GD*: u svakom koraku istovremeno ažuriramo parametre modela koristeći sve trening podatke

for  $t = 1, 2, \dots, \text{maxIters}$ :

for  $d = 1, 2, \dots, D$

- $$\theta_d^{(t+1)} = \theta_d^{(t)} - \frac{\alpha}{N} \sum_{i=1}^N (h_{\theta}(x^{(i)}) - y^{(i)}) x_d^{(i)}$$

moramo da izračunamo grešku za sve primere pa tek onda radimo promenu

- *Stochastic GD (ili Incremental GD)*: više puta prolazimo kroz skup podataka. Kad god naiđemo na trening podatak, ažuriramo gradijent na osnovu tog (jednog) trening podatka

for  $t = 1, 2, \dots, \text{maxIters}$ :

for  $d = 1, 2, \dots, D$

for  $i = 1, 2, \dots, N$

- $$\theta_d^{(t+1)} = \theta_d^{(t)} - \alpha (h_{\theta}(x^{(i)}) - y^{(i)}) x_d^{(i)}$$

nema sume ovde, promenu radimo odmah za svaki primer

# Mini-Batch Stochastic GD

- Batch GD*: u svakom koraku istovremeno ažuriramo parametre modela koristeći *sve* trening podatke

```
batch_size = 32
```


```
for  $t = 1, 2, \dots, \text{maxIters}$ :
```

```
    mini_batch = slučajno odabrana 32 primera iz skupa podataka
```

```
    for  $d = 1, 2, \dots, D$ 
```

- $$\theta_d^{(t+1)} = \theta_d^{(t)} - \frac{\alpha}{\text{batch\_size}} \sum_{i=1}^{\text{batch\_size}} (h_{\theta}(x^{(i)}) - y^{(i)}) x_d^{(i)}$$

ovde koristimo samo primere iz skupa mini\_batch



# Batch GD vs. Stochastic GD

- *Batch GD* mora da skenira ceo skup podataka da bi napravio jedan korak
  - Ovo je skupa operacija ako je broj podataka  $N$  velik
- *Stochastic GD* može da napreduje odmah, i napreduje sa svakim uočenim trening podatkom
- Često, *stochastic GD* dovede  $\theta$  „blizu“ minimuma mnogo brže od *batch GD*
- Međutim, dešava se da nikada ne „konvergira“ u minimum (parametri  $\theta$  osciluju oko minimuma)
  - U praksi, tačke „blizu“ minimuma su dovoljno dobre
- Iz ovih razloga, kada je skup podataka velik, preferiramo *stochastic GD*

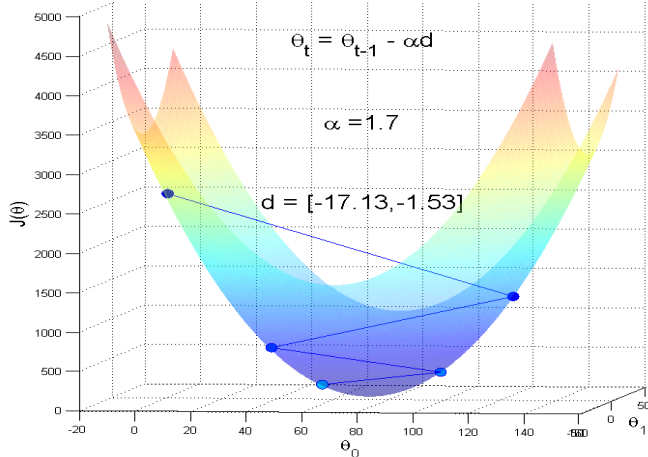
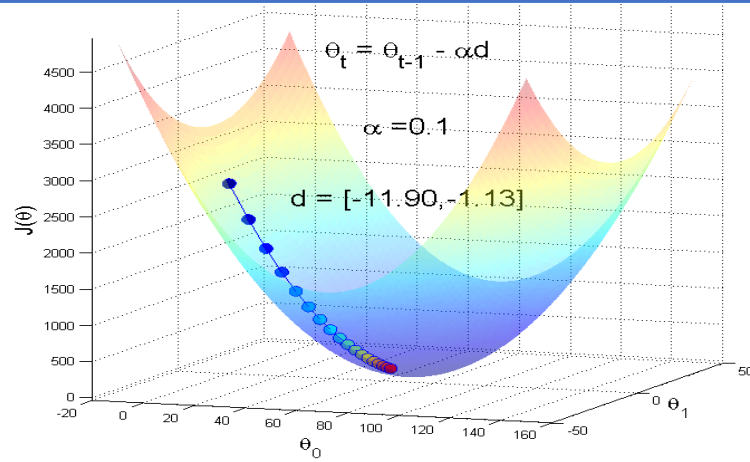


# Mini-Batch Stochastic GD

---

- Predstavlja ravnotežu koja ispravlja mane prethodna dva algoritma
- Najčešće se koristi u praksi

# Odabir $\alpha$

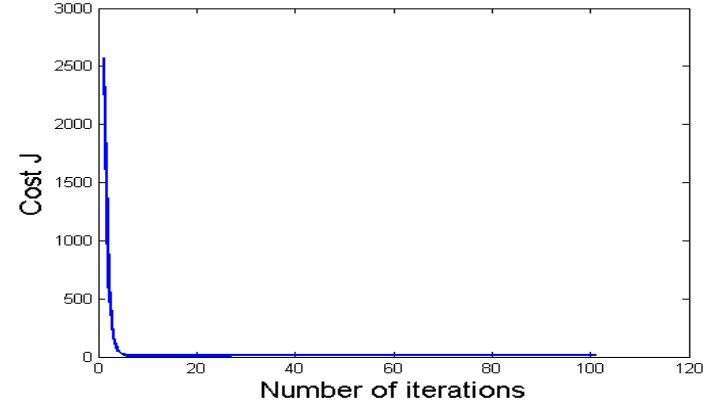
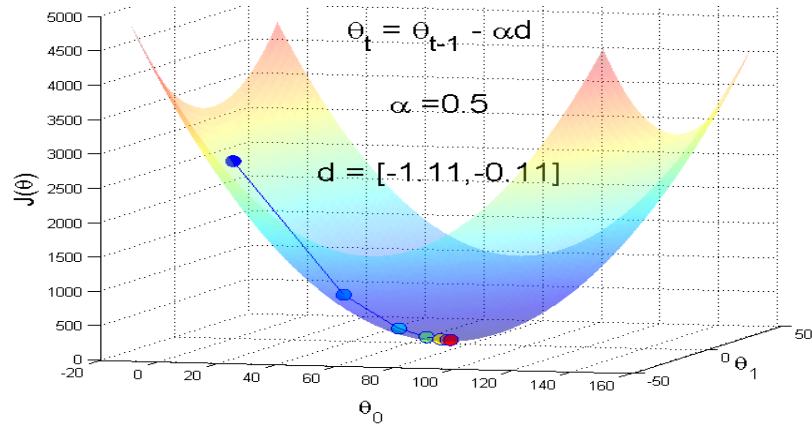
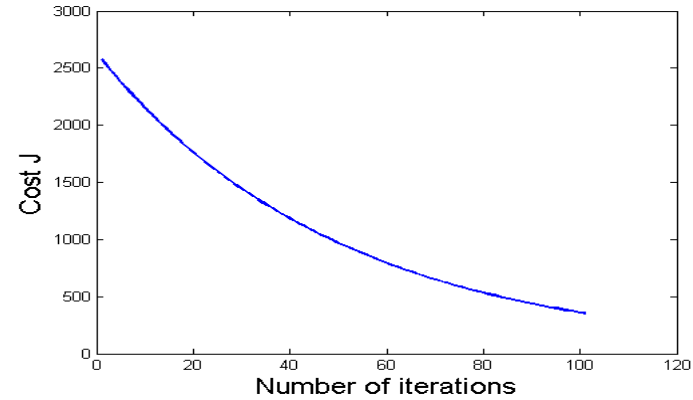
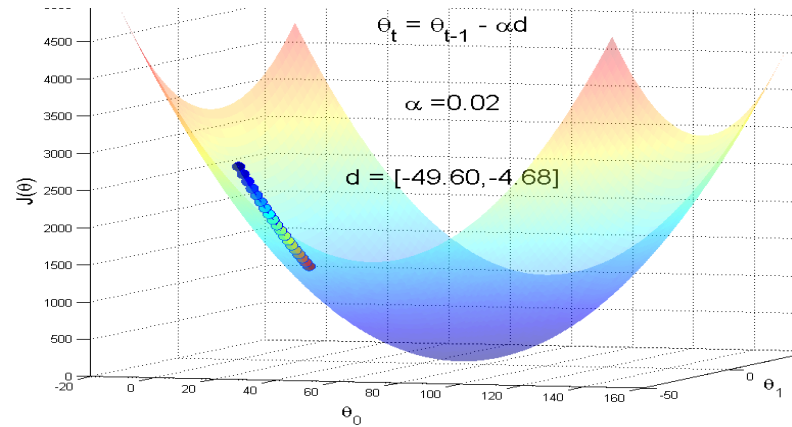


$$\theta_d^{(t+1)} = \theta_d^{(t)} - \alpha \frac{\partial}{\partial \theta_d} J(\theta)$$

- *Gradient descent* može da konvergira u minimum za fiksiranu vrednost  $\alpha$ : kako se približavamo minimumu koraci su sve manji jer je gradijent (nagib) sve manji
- Ako se nalazimo levo od minimuma vrednost gradijenta je negativna:  $\theta$  će da raste
- Ako se nalazimo desno od minimuma vrednost gradijenta je pozitivna:  $\theta$  se smanjuje

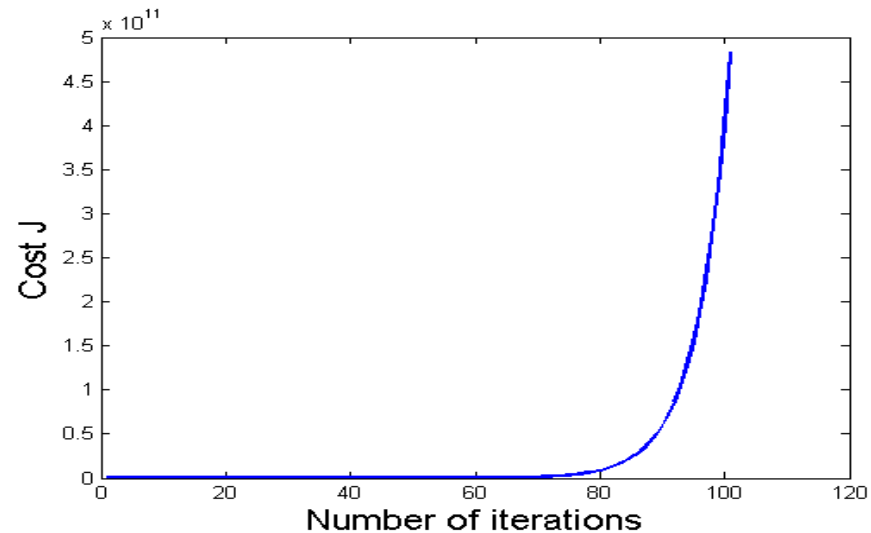
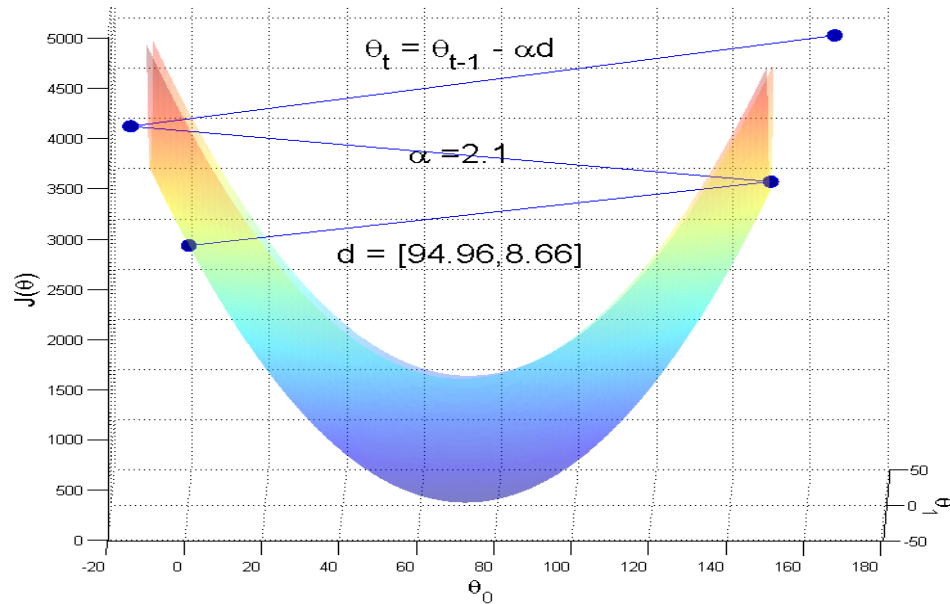
# Odabir $\alpha$

- Za male vrednosti  $\alpha$  *gradient descent* je spor



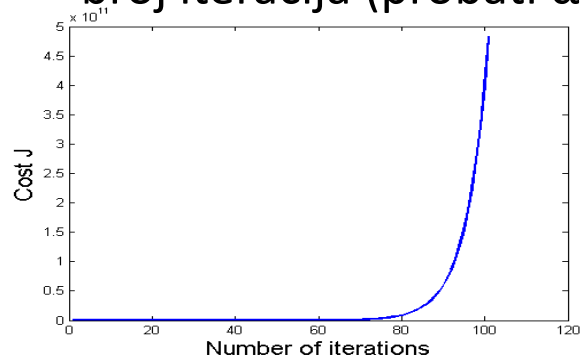
# Odabir $\alpha$

- Za preveliko  $\alpha$  *gradient descent* ne konvergira, a može čak i da divergira

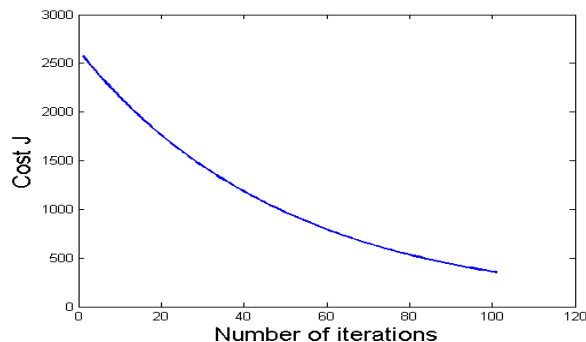


# Odabir $\alpha$ - zaključak

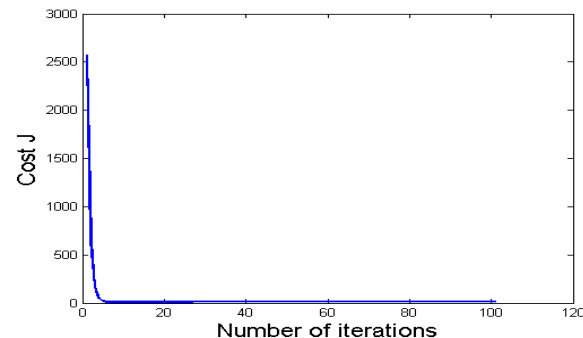
- Da bismo odredili  $\alpha$ , najbolje je posmatrati grafik funkcije greške  $J(\theta)$  u odnosu na broj iteracija (probati  $\alpha = 0.001$ ,  $\alpha = 0.01$ ,  $\alpha = 0.1$ ,  $\alpha = 1, \dots$ ):



Došlo je do divergencije.  
Smanjiti  $\alpha$



Algoritam je spor. Nije konvergirao u zadatom broju iteracija. Povećati  $\alpha$



Dobra vrednost  $\alpha$ . Algoritam je brzo konvergirao.

- Pored fiksne vrednosti  $\alpha$  (koje radi kada je funkcija “*strongly convex*”), čest izbor je i “*stepsize schedule*” – smanjivanje koraka  $\alpha$  sa brojem iteracija:

- $\eta_t = \alpha/t$  ili  $\eta_t = \alpha/\sqrt{t}$

