

Klasifikacija Slika

K-Najbližih Komšija

Uvod u Linearne Klasifikatore

Predavač: Aleksandar Kovačević

Slajdovi preuzeti sa CS 231n, Stanford

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

Klasifikacija Slika: Važan zadatak u Kompjuterskoj Viziji (*Computer Vision*)



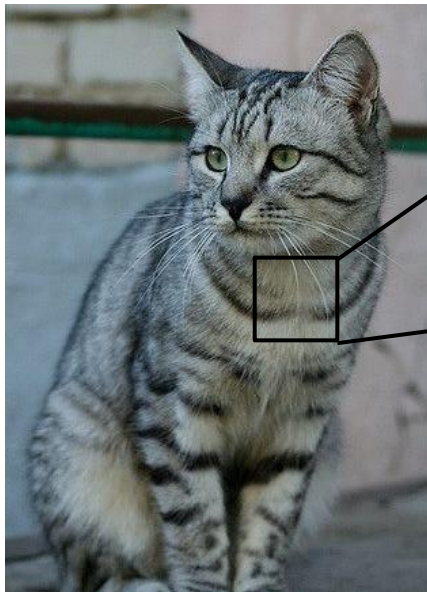
[This image](#) by [Nikita](#)'s
licensed under [CC-BY 2.0](#)

(dat je skup diskretnih klasa)
{dog, cat, truck, plane, ...}



cat

Problem: Ono što računar vidi vs. Ono što želimo da uradi



[This image](#) by [Nikita](#)'s
licensed under [CC-BY 2.0](#)

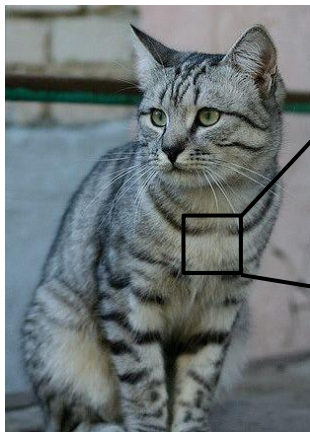
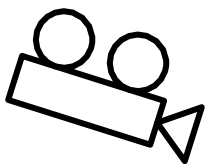
```
[[105 112 108 111 104 99 106 99 96 103 112 119 104 97 93 87]  
[ 91 98 102 106 104 79 98 103 99 105 123 136 110 105 94 85]  
[ 76 85 90 105 128 105 87 96 95 99 115 112 106 103 99 85]  
[ 99 81 81 93 120 131 127 100 95 98 102 99 96 93 101 94]  
[106 91 61 64 69 91 88 85 101 107 109 98 75 84 96 95]  
[114 108 85 55 55 69 64 54 64 87 112 129 98 74 84 91]  
[133 137 147 103 65 81 80 65 52 54 74 84 102 93 85 82]  
[128 137 144 140 109 95 86 70 62 65 63 63 60 73 86 101]  
[125 133 148 137 119 121 117 94 65 79 80 65 54 64 72 98]  
[127 125 131 147 133 127 126 131 111 96 89 75 61 64 72 84]  
[115 114 109 123 150 148 131 110 113 109 100 92 74 65 72 78]  
[ 89 93 90 97 108 147 131 118 113 114 113 109 106 95 77 80]  
[ 63 77 86 81 77 79 102 123 117 115 117 125 125 130 115 87]  
[ 62 65 82 89 78 71 80 101 124 126 119 101 107 114 131 119]  
[ 63 65 75 88 89 71 62 81 120 138 135 105 81 98 110 118]  
[ 87 65 71 87 106 95 69 45 76 130 126 107 92 94 105 112]  
[118 97 82 86 117 123 116 66 41 51 95 93 89 95 102 107]  
[164 146 112 80 82 120 124 104 76 48 45 66 88 101 102 109]  
[157 170 157 120 93 86 114 132 112 97 69 55 70 82 99 94]  
[130 128 134 161 139 100 109 118 121 134 114 87 65 53 69 86]  
[128 112 96 117 150 144 120 115 104 107 102 93 87 81 72 79]  
[123 107 96 86 83 112 153 149 122 109 104 75 80 107 112 99]  
[122 121 102 80 82 86 94 117 145 148 153 102 58 78 92 107]  
[122 164 148 103 71 56 78 83 93 103 119 139 102 61 69 84]]
```

Ono što računari vidi

Slika je matrica brojeva u
rasponu [0, 255]:

npr. 800 x 600 x 3
(3 kanala - RGB)

Izazovi: Pomeranje tačke iz koje gledamo



[105	112	108	111	104	99	106	99	96	103	112	119	104	97	93	87]
[91	98	102	106	104	79	98	103	99	105	123	136	110	105	94	85]
[76	85	90	105	128	105	87	96	95	89	115	112	106	103	99	85]
[99	81	81	93	120	131	127	100	95	98	102	99	96	93	101	94]
[106	91	61	64	69	91	88	85	101	107	109	98	75	84	96	95]
[114	100	85	55	55	69	64	54	64	87	112	129	98	74	84	91]
[133	137	147	103	65	81	80	65	52	54	74	84	102	93	85	82]
[128	137	144	140	109	95	86	78	62	65	63	63	68	73	86	101]
[125	133	140	137	119	121	117	94	65	79	88	65	54	64	72	90]
[127	125	131	147	133	127	126	131	111	96	89	75	61	64	72	84]
[115	114	109	123	150	148	131	118	113	109	100	92	74	65	72	78]
[89	93	90	97	100	147	131	118	113	114	113	109	106	95	77	80]
[63	77	86	81	77	79	102	123	117	115	117	125	125	130	115	87]
[62	65	82	80	78	71	80	101	124	126	119	101	107	114	131	119]
[63	65	75	80	89	71	62	81	128	130	135	105	61	98	110	110]
[87	65	71	87	106	95	69	65	76	130	126	107	92	94	105	112]
[118	97	82	86	117	123	116	66	41	51	95	93	89	95	102	107]
[164	146	112	80	82	120	124	104	76	48	45	66	88	101	102	109]
[157	170	157	120	63	86	114	132	112	97	60	55	78	82	99	94]
[130	128	134	161	139	100	109	118	121	134	114	87	65	53	69	86]
[128	112	96	117	150	144	120	115	104	107	102	93	87	81	72	79]
[123	107	96	86	83	112	153	149	122	109	104	75	80	107	112	99]
[122	121	102	80	82	86	94	117	145	148	153	102	58	78	92	107]
[122	164	148	103	71	56	78	83	93	103	119	139	102	61	69	84]

Svi pikseli se promene
kad pomerimo
kameru!

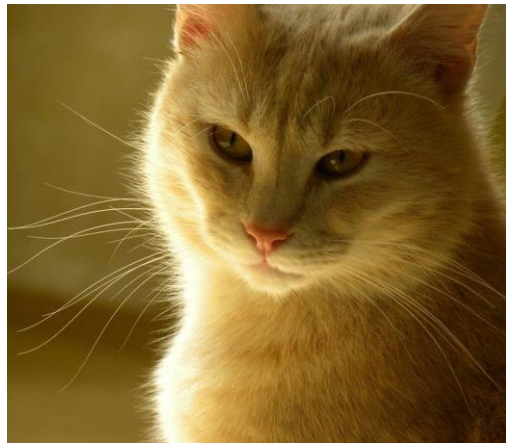
Izazovi: Osvetljenost



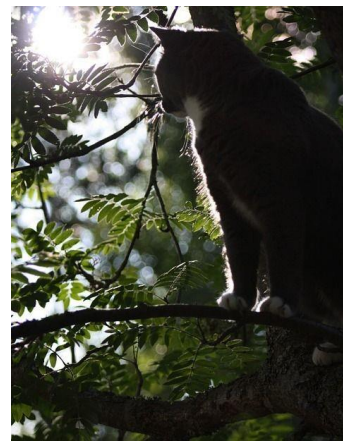
[This image](#) is [CC0 1.0](#) public domain



[This image](#) is [CC0 1.0](#) public domain



[This image](#) is [CC0 1.0](#) public domain



[This image](#) is [CC0 1.0](#) public domain

Izazovi: Deformacija



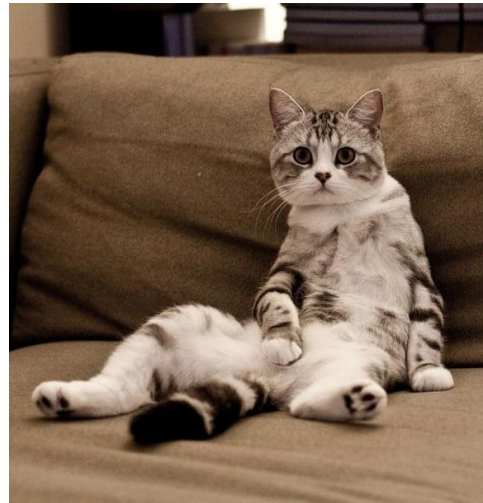
This image by [Umberto Salvagnin](#) is licensed under [CC-BY 2.0](#)



This image by [Umberto Salvagnin](#) is licensed under [CC-BY 2.0](#)



This image by [sare bear](#) is licensed under [CC-BY 2.0](#)



This image by [Tom Thai](#) is licensed under [CC-BY 2.0](#)

Izazovi: Okluzija



[This image](#) is [CC0 1.0](#) public domain



[This image](#) is [CC0 1.0](#) public domain



[This image](#) by [jonsson](#) is licensed under [CC-BY 2.0](#)

Izazovi: Pretrpana pozadina



[This image](#) is [CC0 1.0](#) public domain



[This image](#) is [CC0 1.0](#) public domain

Izazaovi: Varijabilnost unutar iste klase



[This image](#) is [CC0 1.0](#) public domain

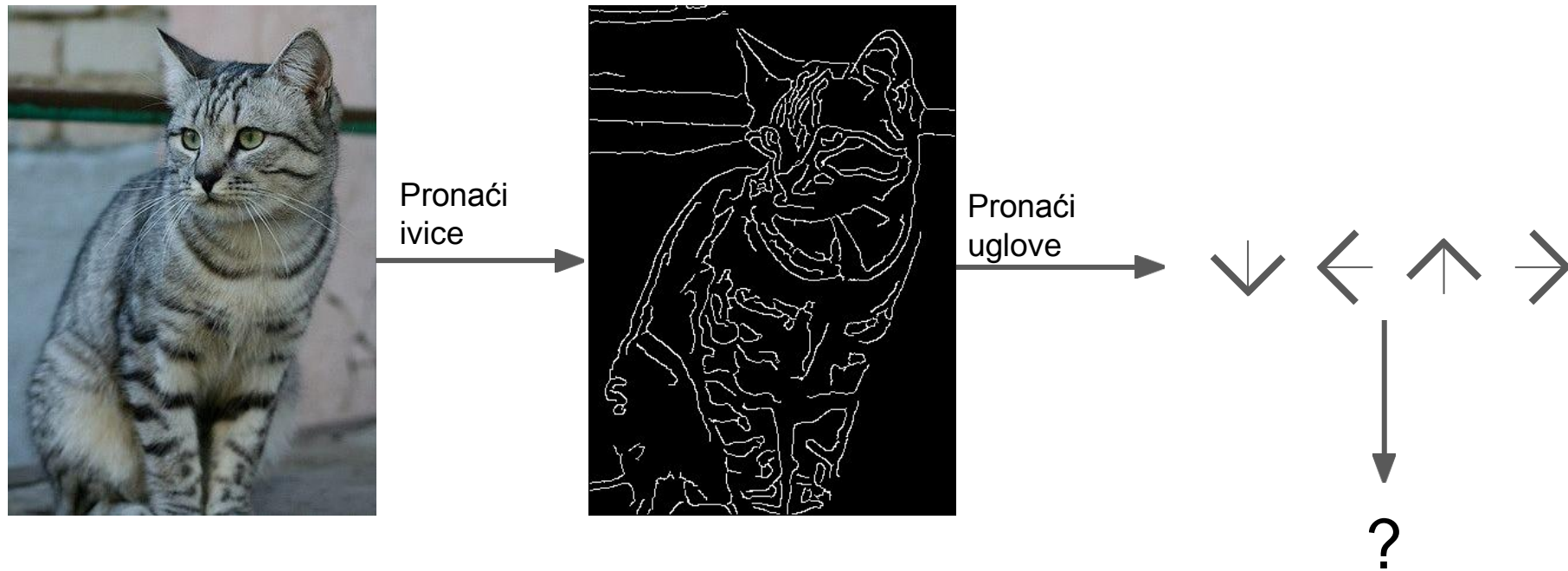
Klasifikator Slika

```
def classify_image(image):  
    # Some magic here?  
    return class_label
```

Za razliku od npr. sortiranja liste,

ne postoji jednostavan način da hard-kodujemo (*hard-code*) algoritam za prepoznavanje mačaka ili drugih klasa.

Postoji jako puno prisupa problemu klasifikacije slika



John Canny, "A Computational Approach to Edge Detection", IEEE TPAMI 1986

Upotreba Nadgledanog Učenja

1. Prikupiti skup slika i oznaka klasa
2. Upotrebiti nadgledano učenje za obučavanje klasifikatora
3. Evaluirati klasifikator na novim slikama

```
def train(images, labels):  
    # Machine learning!  
    return model
```

```
def predict(model, test_images):  
    # Use model to predict labels  
    return test_labels
```

Primer Obučavajućeg skupa

airplane



automobile



bird



cat



deer



Klasifikator K-Najbližih Komšija

K-Nearest Neighbours, KNN

```
def train(images, labels):  
    # Machine learning!  
    return model
```



Čuvamo (pamtimo)
sve slike i njihove
oznake

```
def predict(model, test_images):  
    # Use model to predict labels  
    return test_labels
```



Za novu (test) sliku:
Rezultat predikcije je
oznaka klase najbližije
slike koju imamo u
obučavajućem skupu

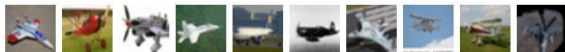
Primer – skup podataka **CIFAR10**

10 klasa

50,000 slika u obučavajućem skupu

10,000 slika u test skupu

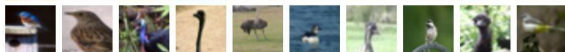
airplane



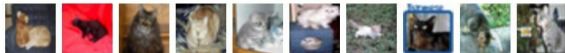
automobile



bird



cat



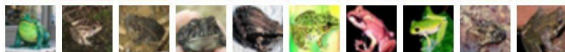
deer



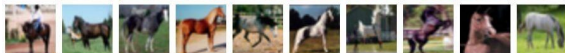
dog



frog



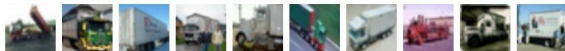
horse



ship



truck



Alex Krizhevsky, "Learning Multiple Layers of Features from Tiny Images", Technical Report, 2009.

Primer – skup podataka **CIFAR10**

10 klasa

50,000 slika u obučavajućem skupu

10,000 slika u test skupu

airplane



automobile



bird



cat



deer



dog



frog



horse



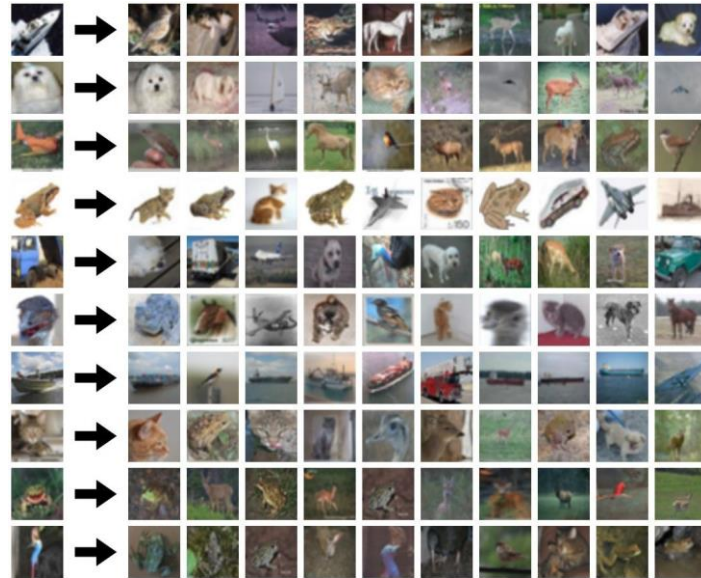
ship



truck



Test slike, Primena KNN



Alex Krizhevsky, "Learning Multiple Layers of Features from Tiny Images", Technical Report, 2009.



Mera sličnost (udaljenosti) za poređenje slika

L1 mera:

$$d_1(I_1, I_2) = \sum_P |I_1^P - I_2^P|$$

test image

56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

-

training image

10	20	24	17
8	10	89	100
12	16	178	170
4	32	233	112

=

pixel-wise absolute value differences

46	12	14	1
82	13	39	33
12	10	0	30
2	32	22	108

add
→ 456

```
import numpy as np

class NearestNeighbor:
    def __init__(self):
        pass

    def train(self, X, y):
        """ X is N x D where each row is an example. Y is 1-dimension of size N """
        # the nearest neighbor classifier simply remembers all the training data
        self.Xtr = X
        self.ytr = y

    def predict(self, X):
        """ X is N x D where each row is an example we wish to predict label for """
        num_test = X.shape[0]
        # lets make sure that the output type matches the input type
        Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

        # loop over all test rows
        for i in xrange(num_test):
            # find the nearest training image to the i'th test image
            # using the L1 distance (sum of absolute value differences)
            distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
            min_index = np.argmin(distances) # get the index with smallest distance
            Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

        return Ypred
```

KNN klasifikator

```
import numpy as np
```

```
class NearestNeighbor:
```

```
    def __init__(self):
```

```
        pass
```

```
    def train(self, X, y):
```

```
        """ X is N x D where each row is an example. Y is 1-dimension of size N """
```

```
        # the nearest neighbor classifier simply remembers all the training data
```

```
        self.Xtr = X
```

```
        self.ytr = y
```

```
    def predict(self, X):
```

```
        """ X is N x D where each row is an example we wish to predict label for """
```

```
        num_test = X.shape[0]
```

```
        # lets make sure that the output type matches the input type
```

```
        Ypred = np.zeros(num_test, dtype = self.ytr.dtype)
```

```
        # loop over all test rows
```

```
        for i in xrange(num_test):
```

```
            # find the nearest training image to the i'th test image
```

```
            # using the L1 distance (sum of absolute value differences)
```

```
            distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
```

```
            min_index = np.argmin(distances) # get the index with smallest distance
```

```
            Ypred[i] = self.ytr[min_index] # predict the label of the nearest example
```

```
        return Ypred
```

Čuvamo (pamtimo)
obučavajući skup

KNN klasifikator

```
import numpy as np

class NearestNeighbor:
    def __init__(self):
        pass

    def train(self, X, y):
        """ X is N x D where each row is an example. Y is 1-dimension of size N """
        # the nearest neighbor classifier simply remembers all the training data
        self.Xtr = X
        self.ytr = y

    def predict(self, X):
        """ X is N x D where each row is an example we wish to predict label for """
        num_test = X.shape[0]
        # lets make sure that the output type matches the input type
        Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

        # loop over all test rows
        for i in xrange(num_test):
            # find the nearest training image to the i'th test image
            # using the L1 distance (sum of absolute value differences)
            distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
            min_index = np.argmin(distances) # get the index with smallest distance
            Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

        return Ypred
```

Za test sliku:

Naći nasličniju (najbližu) sliku
iz obučavajućeg skupa

Vratiti oznaku klase te slike kao
predikciju klase za test sliku


```

import numpy as np

class NearestNeighbor:
    def __init__(self):
        pass

    def train(self, X, y):
        """ X is N x D where each row is an example. Y is 1-dimension of size N """
        # the nearest neighbor classifier simply remembers all the training data
        self.Xtr = X
        self.ytr = y

    def predict(self, X):
        """ X is N x D where each row is an example we wish to predict label for """
        num_test = X.shape[0]
        # lets make sure that the output type matches the input type
        Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

        # loop over all test rows
        for i in xrange(num_test):
            # find the nearest training image to the i'th test image
            # using the L1 distance (sum of absolute value differences)
            distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
            min_index = np.argmin(distances) # get the index with smallest distance
            Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

        return Ypred

```

KNN klasifikator

Sa N slika u
obučavajućem
skupu, koliko traje
obučavanje i
primena na test
sliku?

```

import numpy as np

class NearestNeighbor:
    def __init__(self):
        pass

    def train(self, X, y):
        """ X is N x D where each row is an example. Y is 1-dimension of size N """
        # the nearest neighbor classifier simply remembers all the training data
        self.Xtr = X
        self.ytr = y

    def predict(self, X):
        """ X is N x D where each row is an example we wish to predict label for """
        num_test = X.shape[0]
        # lets make sure that the output type matches the input type
        Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

        # loop over all test rows
        for i in xrange(num_test):
            # find the nearest training image to the i'th test image
            # using the L1 distance (sum of absolute value differences)
            distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
            min_index = np.argmin(distances) # get the index with smallest distance
            Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

        return Ypred

```

KNN klasifikator

Sa N slika u obučavajućem skupu, koliko traje obučavanje i primena na test sliku?

Odgovor:

Obučavanje $O(1)$,

Primena na test sliku $O(N)$

```

import numpy as np

class NearestNeighbor:
    def __init__(self):
        pass

    def train(self, X, y):
        """ X is N x D where each row is an example. Y is 1-dimension of size N """
        # the nearest neighbor classifier simply remembers all the training data
        self.Xtr = X
        self.ytr = y

    def predict(self, X):
        """ X is N x D where each row is an example we wish to predict label for """
        num_test = X.shape[0]
        # lets make sure that the output type matches the input type
        Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

        # loop over all test rows
        for i in xrange(num_test):
            # find the nearest training image to the i'th test image
            # using the L1 distance (sum of absolute value differences)
            distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
            min_index = np.argmin(distances) # get the index with smallest distance
            Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

        return Ypred

```

KNN klasifikator

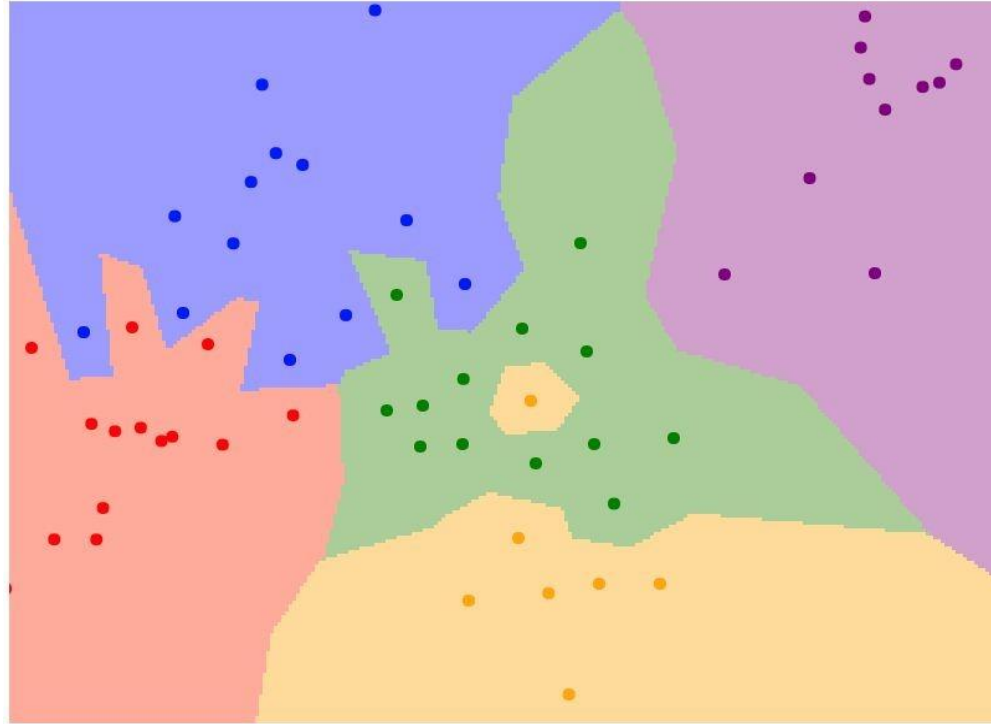
Sa N slika u
obučavajućem skupu,
koliko traje obučavanje i
primena na test sliku?

Odgovor:

Obučavanje $O(1)$,
Primena na test
sliku $O(N)$

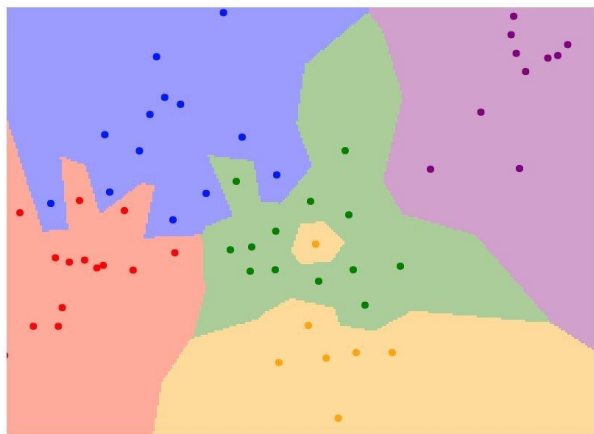
To nije ono što želimo:
hoćemo klasifikatore koji se
brzo primenjuju, dok je
sporo obučavanje prihvatljivo

KNN u 2d

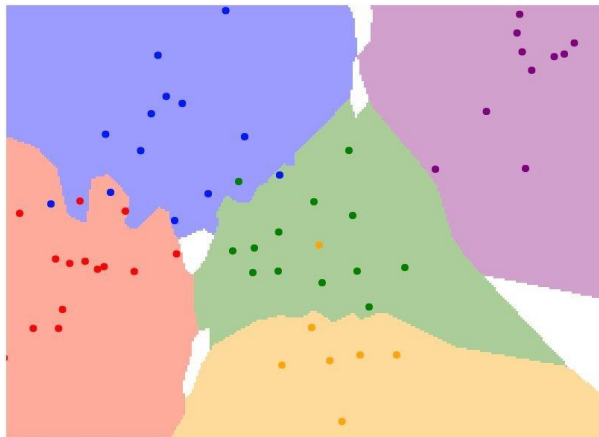


KNN u 2d

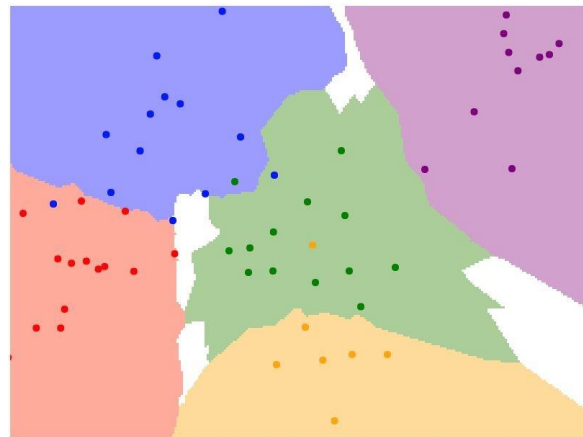
Umesto da gledamo klasu samo jedne najbliže slike,
uradimo **većinsko glasanje** prvih K najbližih slika



$K = 1$



$K = 3$



$K = 5$

Kako to izgleda u praksi?



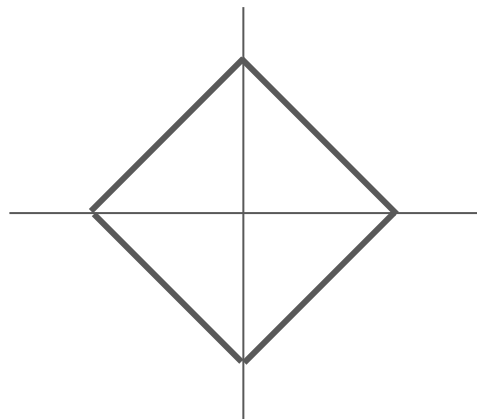
Kako to izgleda u praksi?



K-Najbližih Komšija: Mera Sličnosti

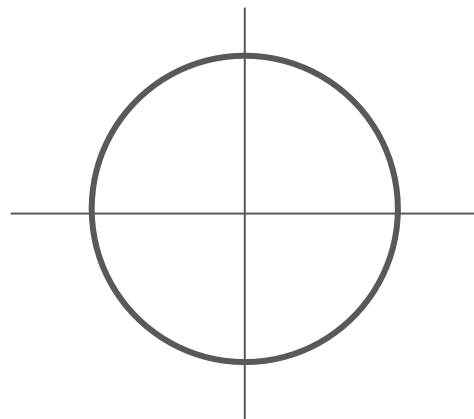
L1 (Manhattan) rastojanje

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



L2 (Euklidsko) rastojanje

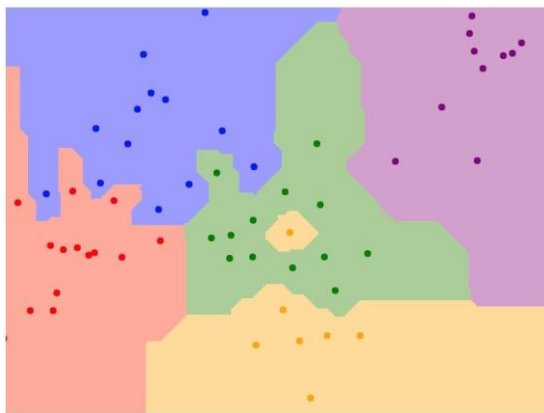
$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$



K-Najbližih Komšija: Mera Sličnosti

L1 (Menhetin) rastojanje

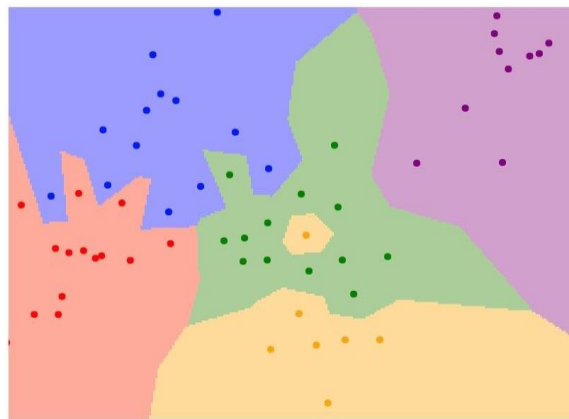
$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



K = 1

L2 (Euklidsko) rastojanje

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$



K = 1

Hiper-parametri (*Hyperparameters*)

Koja je najbolja vrednost za k ?

Koja je najbolja **mera sličnosti**?

Ovo su **hiper-parametri**: odluke koje moramo da donesemo o modelu pre procesa učenja.

Postoje metode koje nam mogu pomoći da donesemo odluku, neke od njih ćemo raditi tokom krusa

Hiper-parametri (*Hyperparameters*)

Koja je najbolja vrednost za k ?

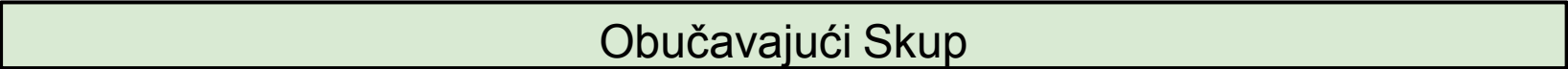
Koja je najbolja **mera sličnosti**?

Ovo su **hiper-parametri**: odluke koje moramo da donesemo o modelu pre procesa učenja.

Hiper-parametri jako zavise od problema. Nema univerzalnih vrednosti!
Ako je moguće treba probati sve izbore da bi našli najbolji.

Kako određujemo hiper-parametre?

Način #1: Biramo vrednosti koje daju najbolje rezultate na obučavajućem skupu

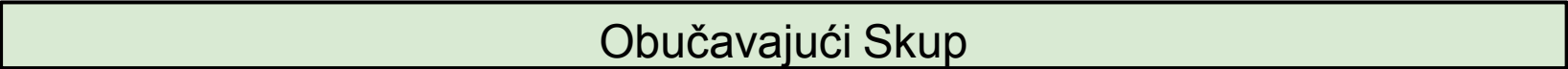


Obučavajući Skup

Kako određujemo hiper-parametre?

Način #1: Biramo vrednosti koje daju najbolje rezultate na obučavajućem skupu

Loš način: Za $K = 1$ uvek imamo savršene rezultate na ob. skupu

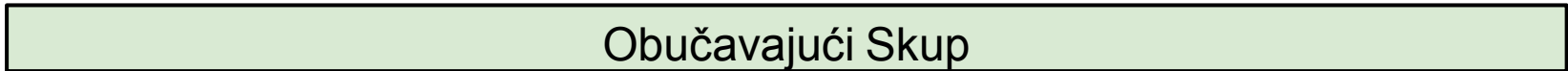


Obučavajući Skup

Kako određujemo hiper-parametre?

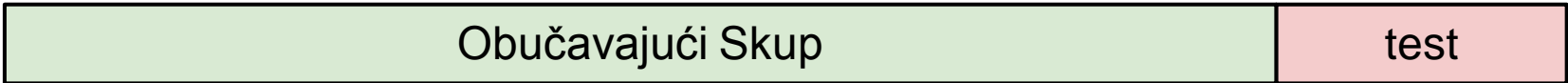
Način #1: Bирамо вредности које дају најбоље резултате на обучавајућем скупу

Loš način: Za $K = 1$ uvek imamo savršene rezultate na ob. skupu



Obučavajući Skup

Način #2: Delimo ob. skup na **obučavajući** i **test**, бирамо вредности које дају најбоље резултате на test skupu



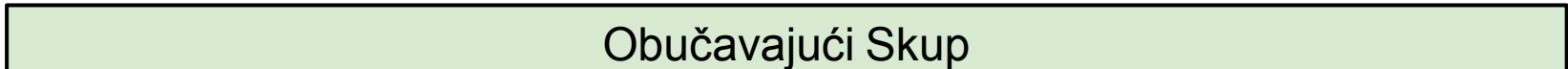
Obučavajući Skup

test

Kako određujemo hiper-parametre?

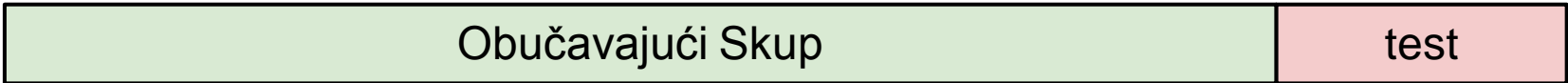
Način #1: Biramo vrednosti koje daju najbolje rezultate na obučavajućem skupu

Loš način: Za $K = 1$ uvek imamo savršene rezultate na ob. skupu



Obučavajući Skup

Način #2: Delimo ob. skup na **obučavajući** i **test**, biramo vrednosti koje daju najbolje rezultate na test skupu



Obučavajući Skup

test

Loš način: Nećamo imati nikakvu realnu predstavu o tome kako će se model ponašati na stvarno nepozatim podacima
Da li znate zašto je tako?

Kako određujemo hiper-parametre?

Način #1: Biramo vrednosti koje daju najbolje rezultate na obučavajućem skupu

Loš način: Za $K = 1$ uvek imamo savršene rezultate na ob. skupu

Obučavajući Skup

Način #2: Delimo ob. skup na **obučavajući** i **test**, biramo vrednosti koje daju najbolje rezultate na test skupu

Loš način: Nećamo imati nikakvu realnu predstavu o tome kako će se model ponašati na stvarno nepozatim podacima
Da li znate zašto je tako?

Obučavajući Skup

test

Način #3: Delimo ob. skup na **obučavajući**, **validacioni**, i **test**; biramo vrednosti koje daju najbolje rezultate na validacionom, konačnu evalaciju radimo na test skupu

Sad je sve korektno!

Obučavajući Skup

validacioni skup

test

Kako određujemo hiper-parametre?

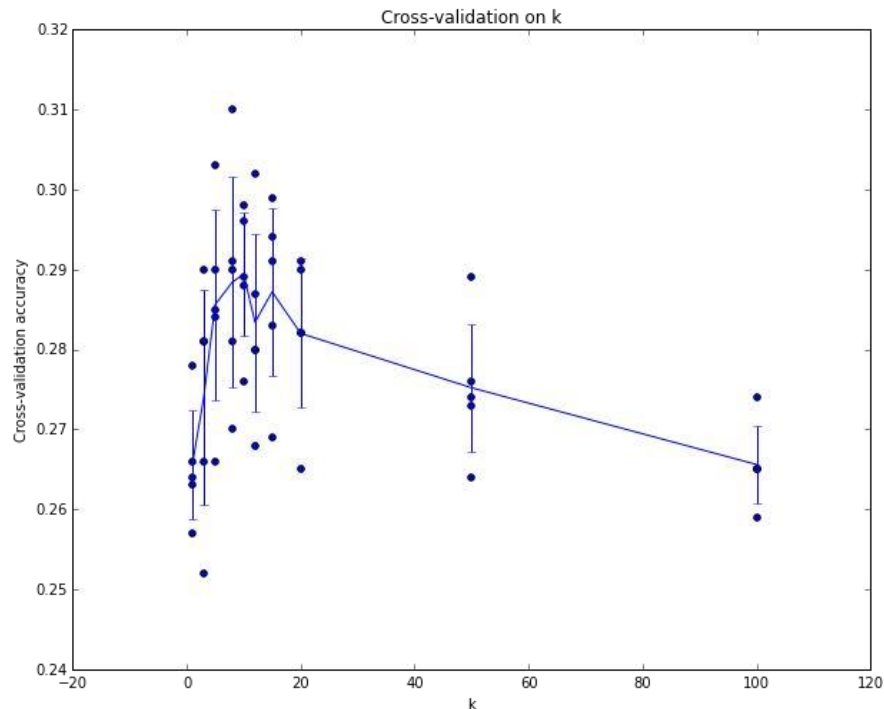
Obučavajući skup

Način #4: Unakrsna Validacija: Delimo podatke u delove (*fold*), svaki deo koristimo kao validacioni skup, pa uzimamo prosek rezultata

fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test

Korisno kad imamo manji obučavajući skup, ali retko upotrebljeno u deep learning oblasti

Određivanje hiper-parametara



Koristimo petostruku unakrsnu validaciju za određivanje k .

Svaka tačka: tačnost dobijena unakrsnom validacijom za jedno k .

Ako delove u unakrs. val. biramo na slučajan način za isto k možemo dobiti više različitih rezultata. Linija koja ide kroz ceo grafikon povezuje te srednje vrednosti. Vertikalne linije su standardne devijacije

(Izgleda da za $k \sim 7$ imamo najbolje rezultate za ovaj ob. skup)

KNN se u današnje vreme nikada ne koristi za klasifikaciju slika

- Vrlo spor kad se primenjuje na test
- Mere sličnosti koje smo pokazali nisu dovoljno dobre kada se primene na piksele

Original



Neki delovi zacrnjeni



Malo pomerena slika



Malo zatamnjena

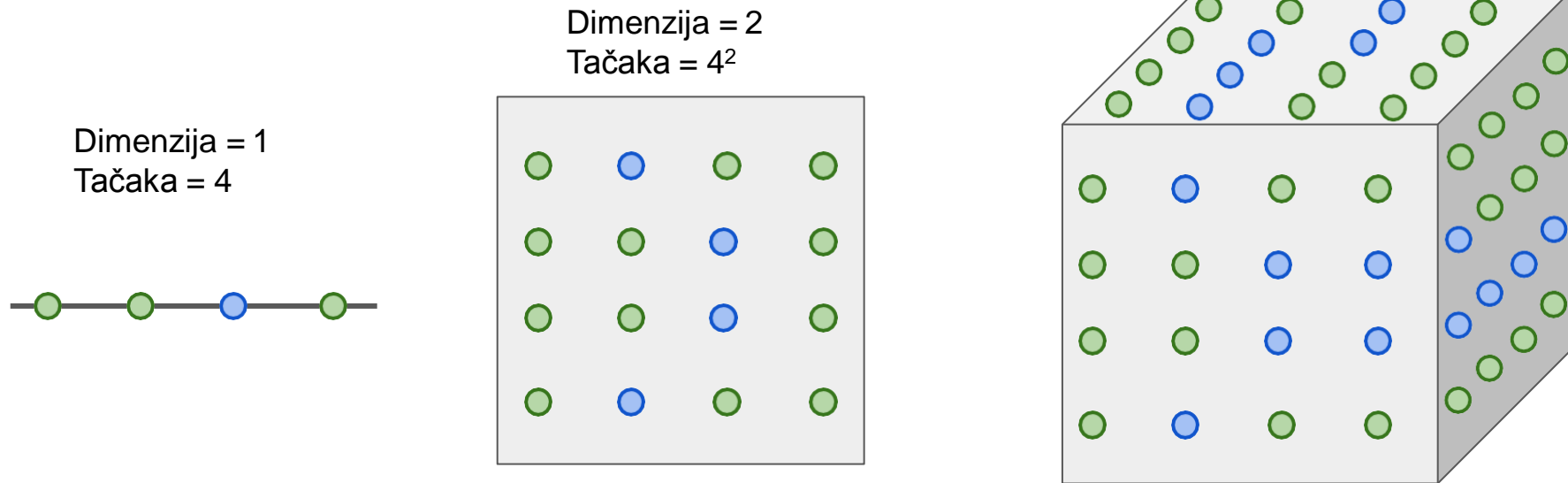


[Original image](#) is
[CC0 public domain](#)

(Sve 3 slike su na istoj L2 udaljenosti od prve slike)

KNN se u današnje vreme nikada ne koristi za klasifikaciju slika

- Problem sa visoko-dimenzionim prostorima (*Curse of dimensionality*)



KNN: Rezime

Kod **klasifikacije slika** počinjemo sa **obučavajućim skupom** slika i njima dodeljenih klasa i cilj nam je da predvidimo klase za **test skup**

Klasifikator **K-Najbližih Komšja** vrši predikciju klase na osnovu najbližih slika iz obučavajućeg skupa

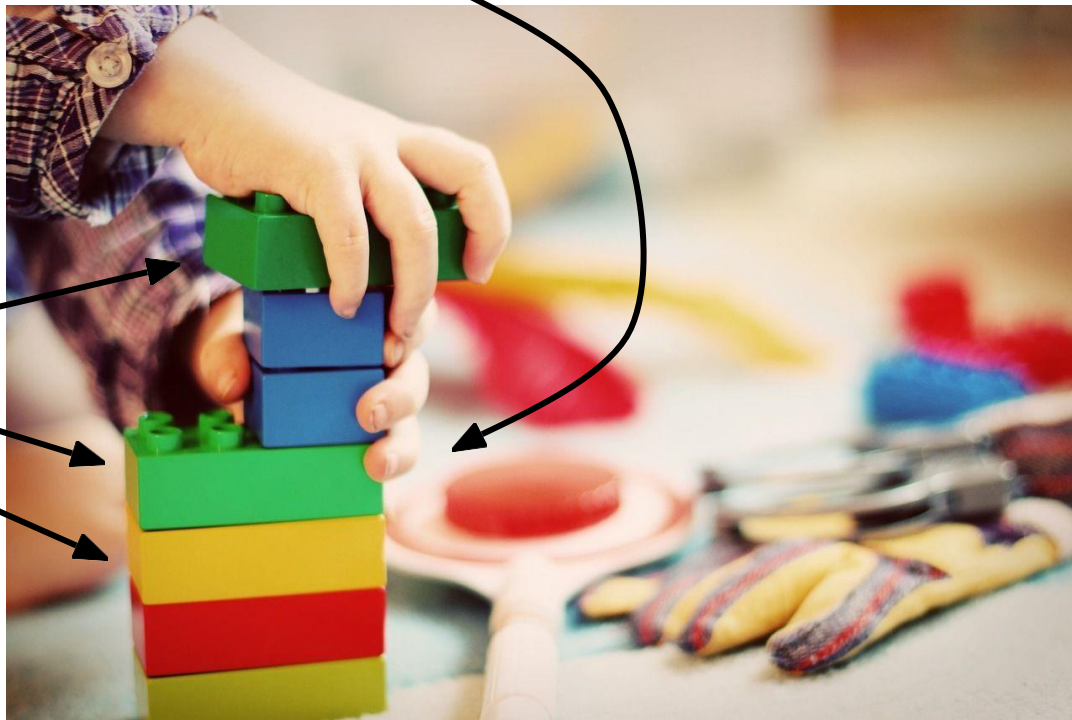
Mera udaljenosti (sličnosti) i K su **hiper-parametri**

Hiper-parametre bираmo pomoću **validacionog skupa ili unakrsne validacije**; model primenjujemo na test skup jednom i to na kraju kad smo završili sa određivanjem svih parametra! Bilo kakvo štelovanje modela prema test skupu smatra se varanjem tj. objavljivanje takvih rezultata nije etički.

Linearni Klasifikatori

Neuronska Mreža

Linearni
Klasifikatori



[This image](#) is [CC0 1.0](#) public domain

Two young girls are playing with lego toy. Boy is doing backflip on wakeboard



Man in black shirt is playing guitar.

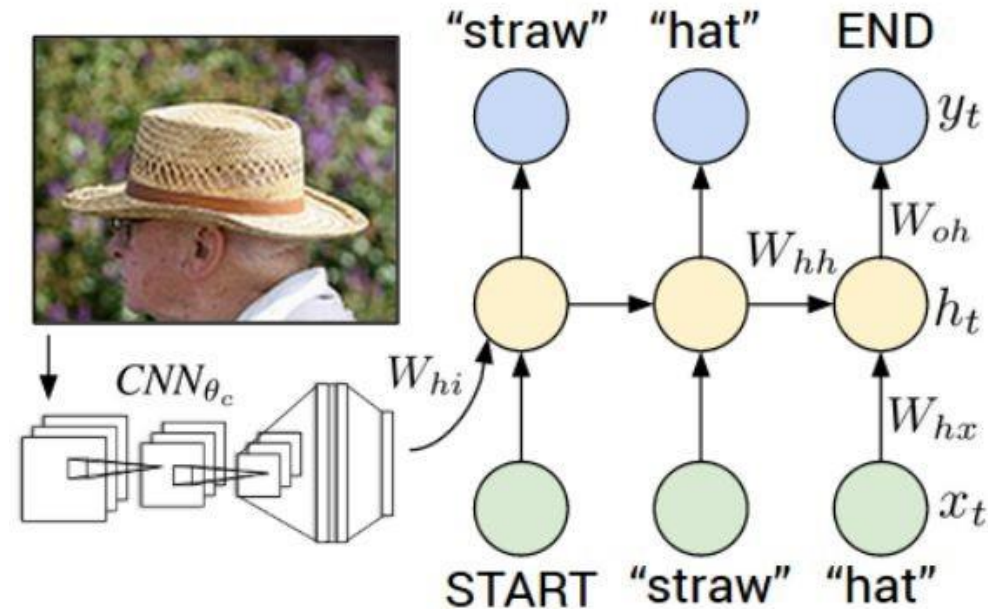
Construction worker in orange safety vest is working on road.

Karpathy and Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015
Figures copyright IEEE, 2015. Reproduced for educational purposes.

Two young girls are playing with lego toy. Boy is doing backflip on wakeboard



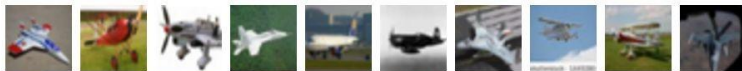
Man in black shirt is playing guitar. Construction worker in orange safety vest is working on road.



Karpathy and Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015
 Figures copyright IEEE, 2015. Reproduced for educational purposes.

CIFAR10

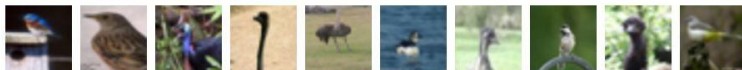
airplane



automobile



bird



cat



deer



dog



frog



horse



ship



truck



50,000 obučavajućih slika
svaka slika je dimenzionalnosti
32x32x3

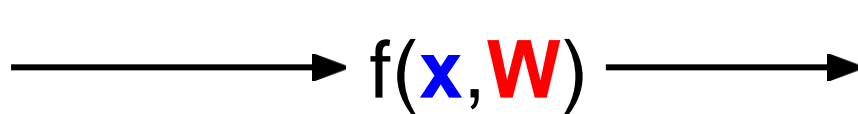
10,000 test slika

Parametrizovani Pristupi za Klasifikaciju

Slika



Niz **32x32x3** brojeva
(3072 brojeva ukupno)



10 brojeva koji
odgovaraju 10 klasa

\mathbf{W}

parametri ili
težine

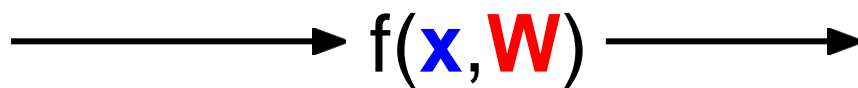
Parametrizovani Pristupi: Linearni Klasifikator

Slika



Niz **32x32x3** brojeva
(3072 brojeva ukupno)

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

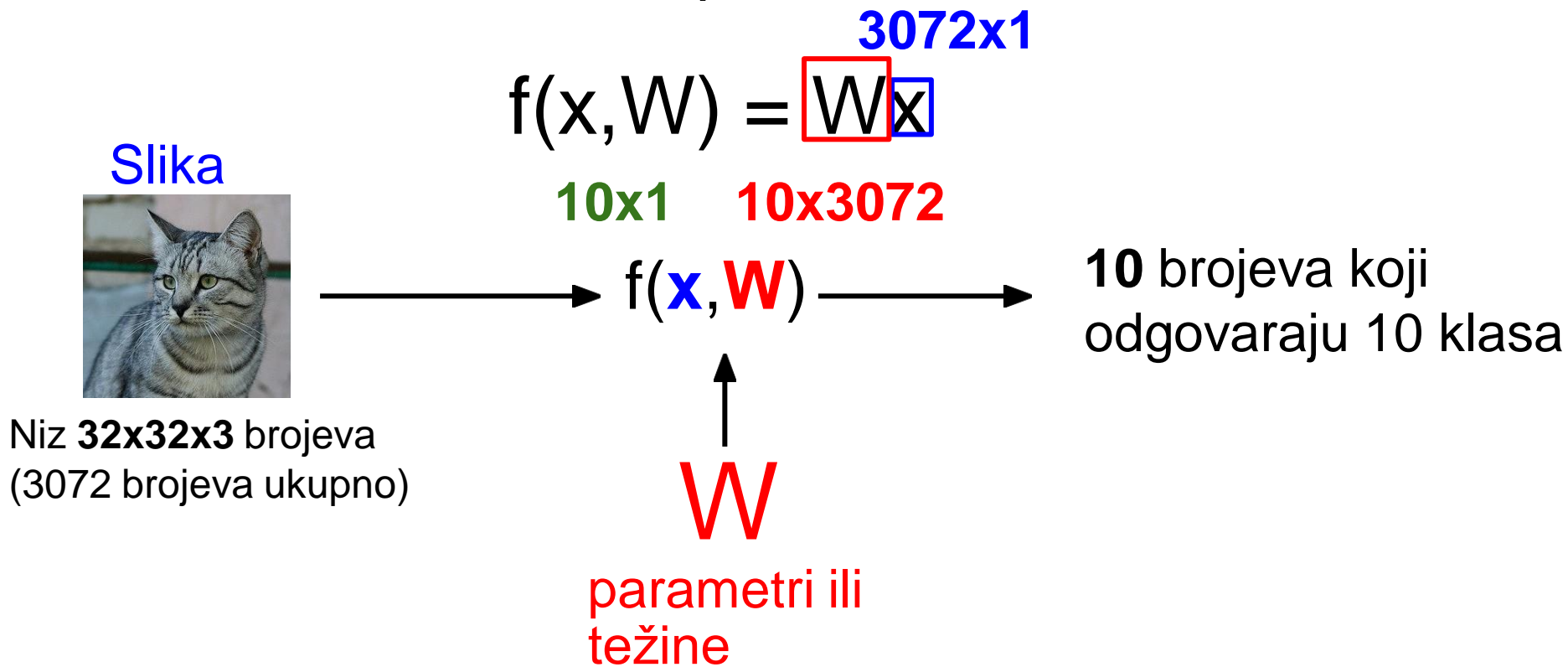


10 brojeva koji
odgovaraju 10 klasa

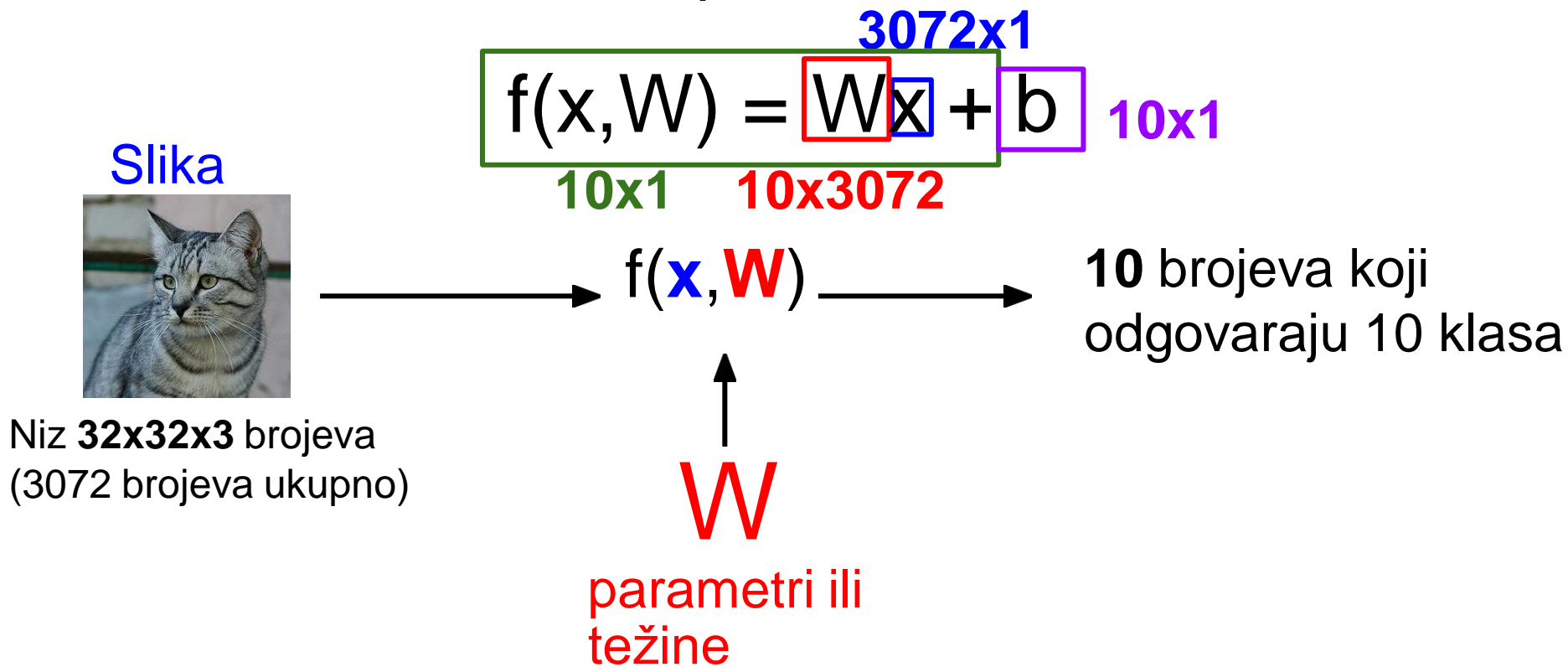
W

parametri ili
težine

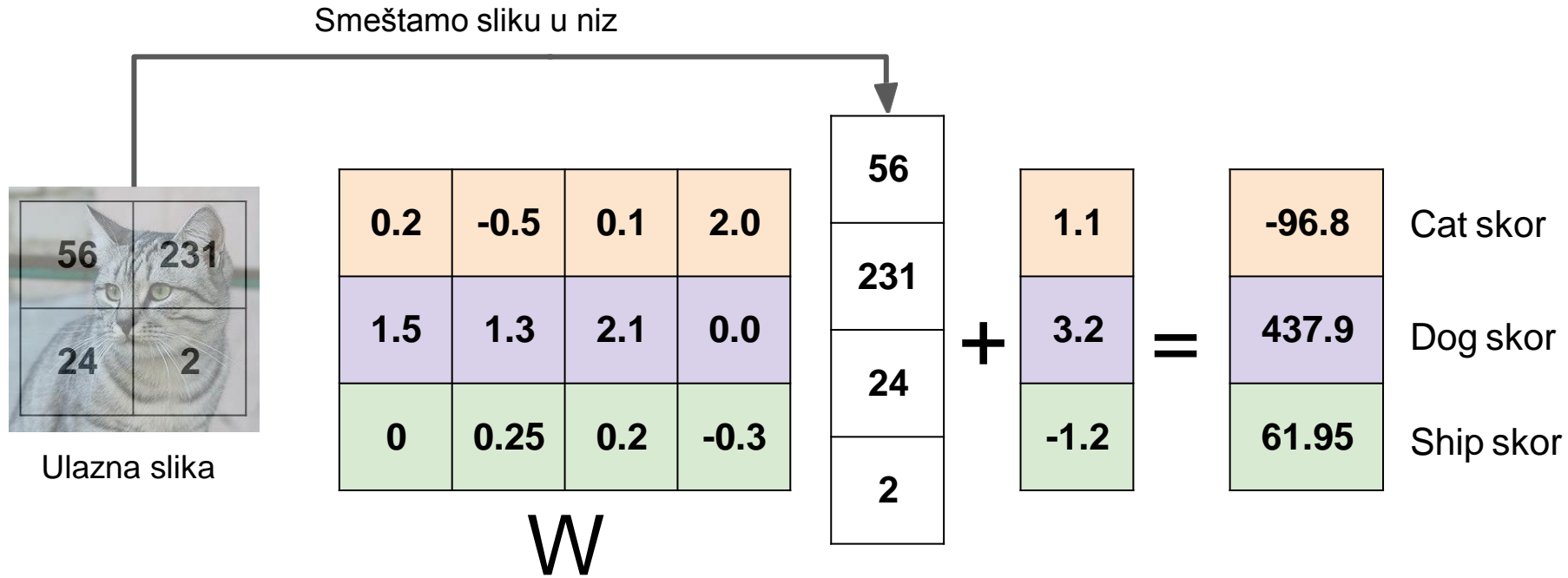
Parametrizovani Pristupi: Linearni Klasifikator



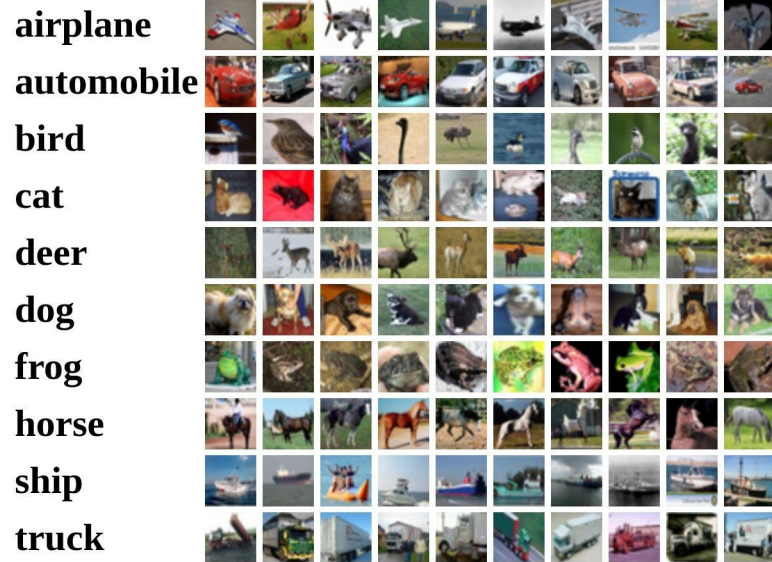
Parametrizovani Pristupi: Linearni Klasifikator



Primer sa slikom od 4 piksela i 3 klase (cat/dog/ship)



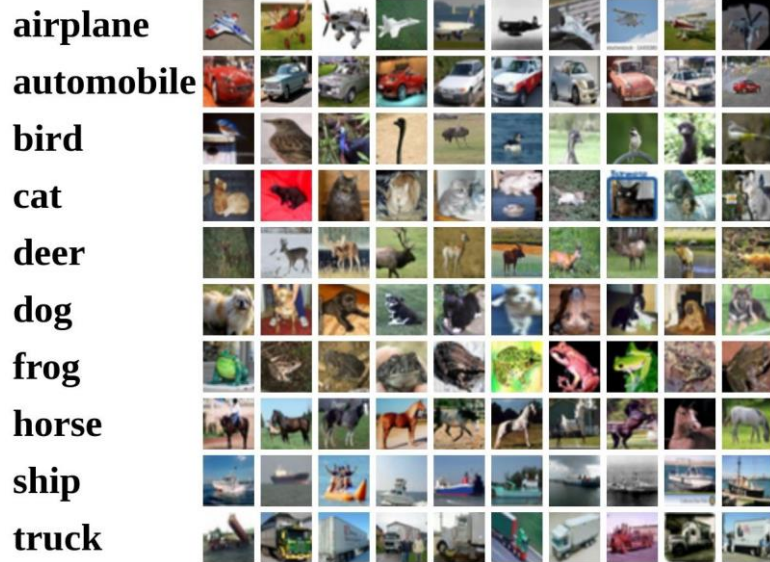
Interpretacija linearnog klasifikatora



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

Šta tačno radi?

Interpretacija linearnog klasifikatora

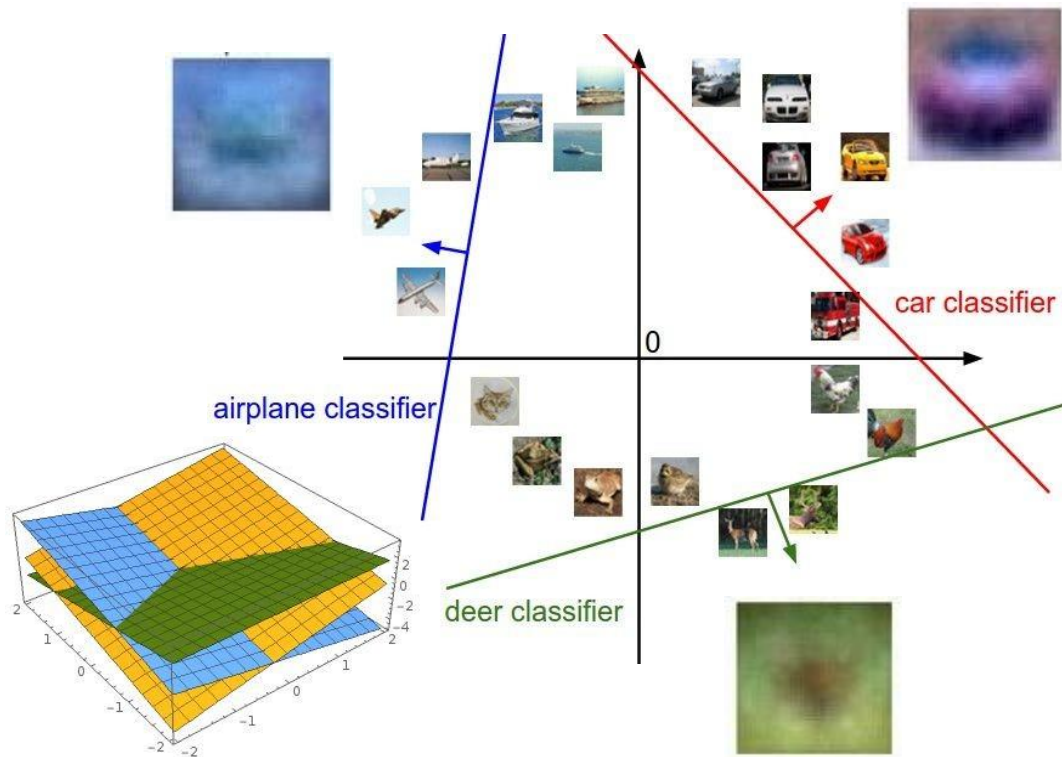


$$f(x, W) = Wx + b$$

Dole su vizualizacije
matrica težina W za sve
klase CIFAR-10:



Interpretacija linearnog klasifikatora



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



Niz **32x32x3** brojeva
(3072 brojeva ukupno)

Plot created using [WolframCloud](#)

[Cat image](#) by [Nikita](#) is licensed under [CC-BY 2.0](#)

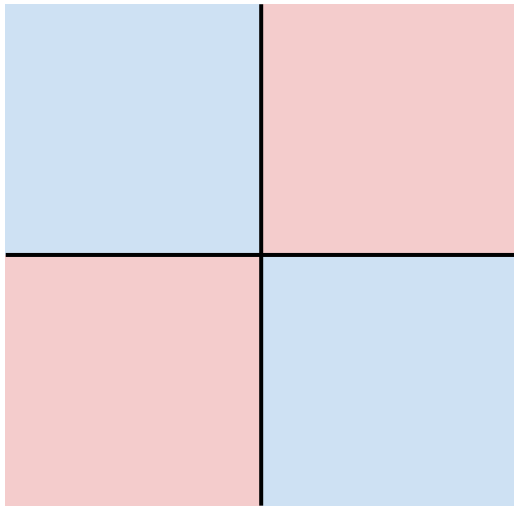
Problemi koji su teški za linearni klasifikator

Klasa 1:

broj piksela > 0 neparan

Klasa 2:

broj piksela > 0 paran

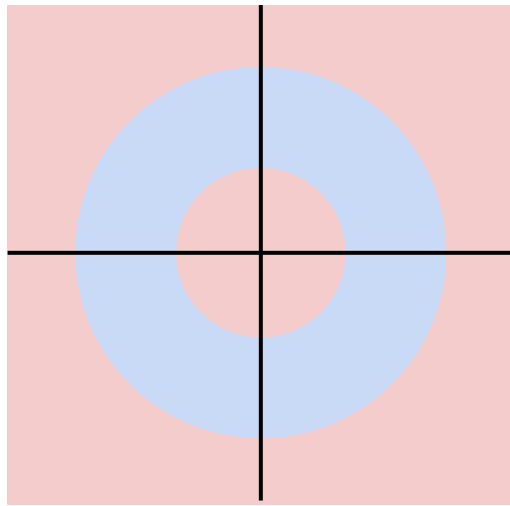


Klasa 1:

$1 \leq L2 \text{ mera} \leq 2$

Klasa 2: Sve

ostalo

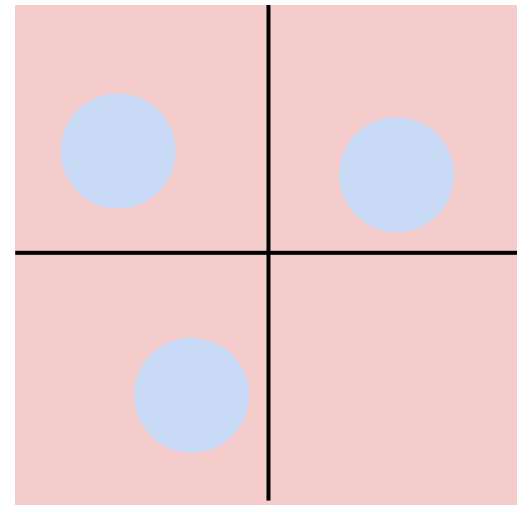


Klasa 1: 3

različita kruga

Class 2: Sve

ostalo



Do sada: Definisali smo (linearnu) skor funkciju $f(x, W) = Wx + b$

Primeri
skorova za 10
klasa za 3
slie i neko W :



Kako da znamo
da li je ovo W
dobar ili loš
klasifikator?

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

[Cat image](#) by [Nikita](#) is licensed under [CC-BY 2.0](#)
[Car image](#) is [CC0 1.0](#) public domain
[Frog image](#) is in the public domain

U nastavku kursa:

- Funkcija greške
- Optimizacija
- Neuronske i kovolutivne mreže!

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

(kvantifikujemo šta je tačno “dobro” W)

(krećemo od slučajno odabranog W i tražimo W koje minimizuje funkciju greške)

(malo ćemo menjati sam oblik funkcije f)