

Neurónové siete pre počítačové videnie

šk.r. 2021-22

Kroky klasifikácie obrazu

RNDr. Zuzana Černeková, PhD.
Ing. Viktor Kocur, PhD.

Štruktúra dnešnej prednášky

- Klasifikácia obrazu
- Klasické prístupy
- Trénovacia, validačná a testovacia množina
- Metóda najbližších susedov - KNN
- Hyperparametre
- Lineárny klasifikátor

Klasifikácia obrazu

- Kľúčová úloha v počítačovom videní

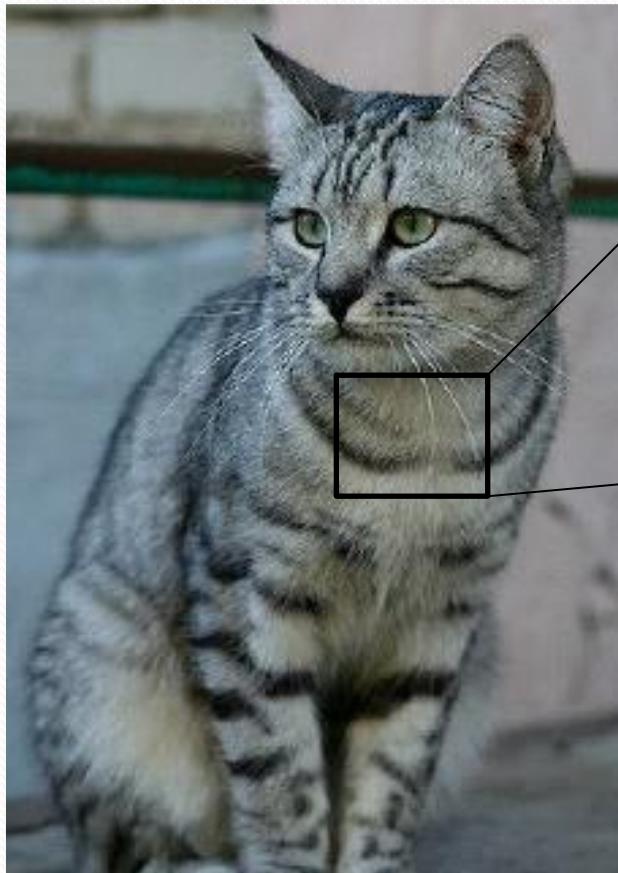


predpoklad: je daná množina diskrétnych označení tried {pes, mačka, lietadlo, auto, ...}



mačka

Problém: sémantická priepast



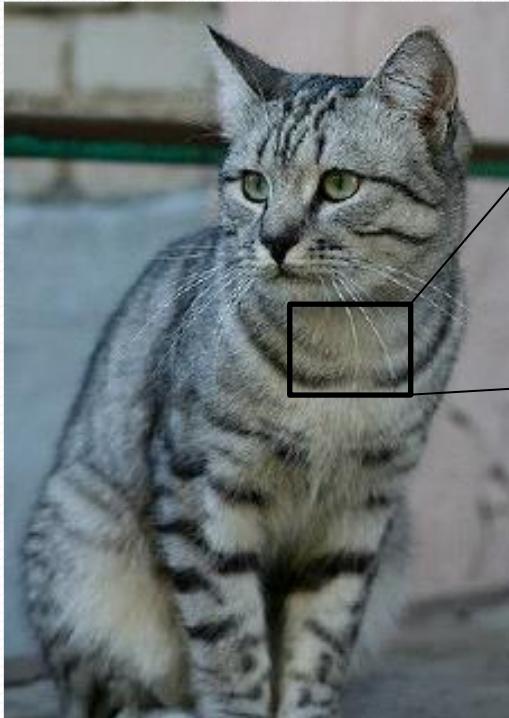
[1105 112 108 111 104 99 106 99 96 103 112 119 104 97 93 87]
[91 98 102 106 104 79 98 103 99 105 123 136 110 105 94 85]
[76 85 98 105 128 105 87 96 95 99 115 112 106 103 99 85]
[99 81 81 93 120 131 127 100 95 98 102 99 96 93 181 94]
[106 91 61 64 69 91 88 85 181 107 109 98 75 84 96 95]
[114 188 85 55 55 69 64 54 64 87 112 129 98 74 84 91]
[133 137 147 183 65 81 86 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]
[125 133 148 137 119 121 187 94 65 79 88 65 54 64 72 98]
[127 125 131 147 133 127 126 131 111 06 80 75 61 64 72 84]
[115 114 109 123 150 148 131 118 113 109 108 92 74 65 72 78]
[89 93 98 97 108 147 131 118 113 114 113 189 106 95 77 88]
[60 77 86 81 77 79 182 123 117 115 117 125 125 138 115 87]
[62 65 82 80 78 71 80 101 124 128 110 181 107 114 131 119]
[63 65 75 88 89 71 62 81 128 138 135 185 81 98 110 118]
[87 03 71 87 106 95 89 45 70 130 126 187 92 94 185 112]
[118 97 82 86 117 123 116 66 41 51 95 93 89 95 182 107]
[164 146 112 80 82 128 124 104 76 48 45 66 88 101 182 109]
[157 170 157 120 93 86 114 132 112 97 69 55 70 82 99 94]
[130 128 134 161 139 108 189 118 121 134 114 07 65 53 69 86]
[120 112 96 117 150 144 120 115 104 107 102 93 87 81 72 79]
[123 187 96 86 83 112 153 149 127 103 104 75 80 107 112 99]
[122 121 102 80 82 86 94 117 145 148 153 182 58 78 92 107]
[122 164 148 183 71 56 78 83 93 103 119 139 102 61 69 84]

Čo počítač vidí

Obraz je iba veľkou mriežkou
čísiel medzi [0,255]

Napr. 800x600x3
(3 kanály RGB)

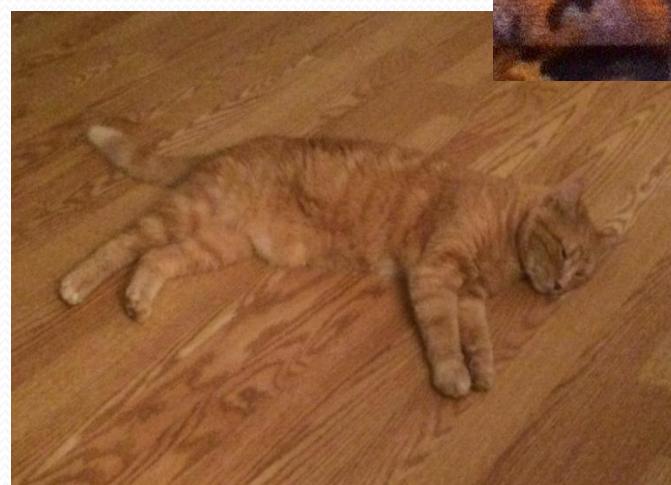
Výzvy: zmena miesta snímania



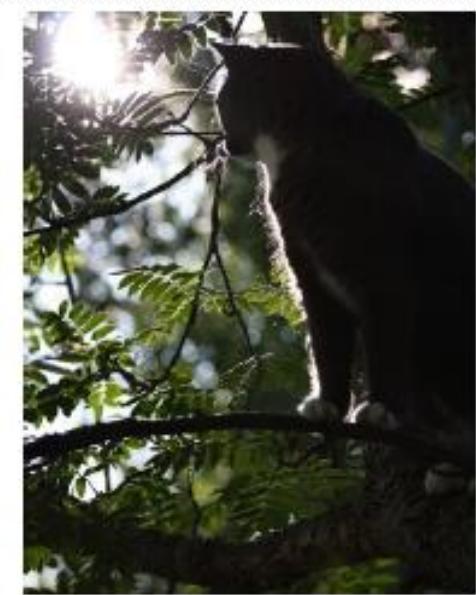
11185	112	188	111	164	99	185	99	85	183	112	119	184	97	93	87		
1	91	98	182	186	104	79	98	123	88	185	123	128	118	105	94	85	
1	36	85	98	185	128	105	87	98	85	98	115	112	186	103	99	85	
1	99	81	81	93	128	121	127	189	85	98	182	99	86	93	181	94	
1	86	91	61	64	69	91	98	25	181	187	188	66	35	84	96	96	
1	114	188	85	55	56	69	64	51	64	87	112	126	68	74	81	61	
1	123	127	117	182	65	81	88	66	53	61	34	84	182	93	85	82	
1	128	137	144	148	189	95	86	79	62	65	63	63	60	73	86	101	
1	129	135	148	137	119	121	117	94	85	75	88	85	54	84	72	98	
1	127	125	131	147	137	127	129	131	111	98	89	75	61	84	72	84	
1	135	114	189	123	154	147	131	118	113	189	188	52	74	85	72	78	
1	89	93	98	97	104	147	131	118	113	118	113	189	186	95	77	99	
1	83	77	85	61	77	79	102	123	117	115	117	125	118	115	87	1	
1	62	68	82	89	70	71	68	181	124	128	133	181	187	114	131	119	
1	63	68	75	88	89	71	62	81	128	138	135	185	81	98	118	118	
1	60	65	71	87	106	95	69	45	78	138	126	163	82	94	105	112	
1	116	87	62	86	117	123	116	66	41	51	85	83	88	95	182	107	
1	164	146	112	98	92	126	124	184	75	48	45	66	66	181	182	109	
1	153	178	157	126	63	93	95	114	122	112	87	68	56	76	92	90	94
1	148	128	144	161	138	166	189	119	121	124	124	83	66	63	68	98	
1	128	112	96	117	154	144	129	115	184	187	182	93	87	81	72	79	
1	123	187	96	88	83	112	153	149	122	184	184	76	88	187	112	99	
1	122	121	183	88	82	98	94	117	145	188	182	88	78	82	107	1	
1	122	164	148	183	71	98	78	83	88	189	112	186	182	81	68	84	

Všetky pixle sa menia,
ked' sa hýbe kamera

Výzvy: výrazné pozadie

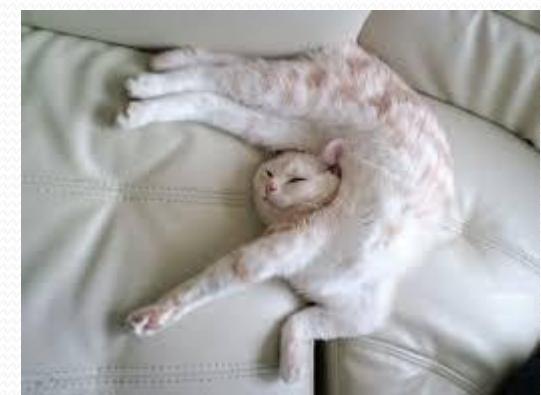


Výzvy: osvetlenie



Neurónové sieti

Výzvy: deformácie



Výzvy: zakrytie



Výzvy: vnútrotriedna variácia



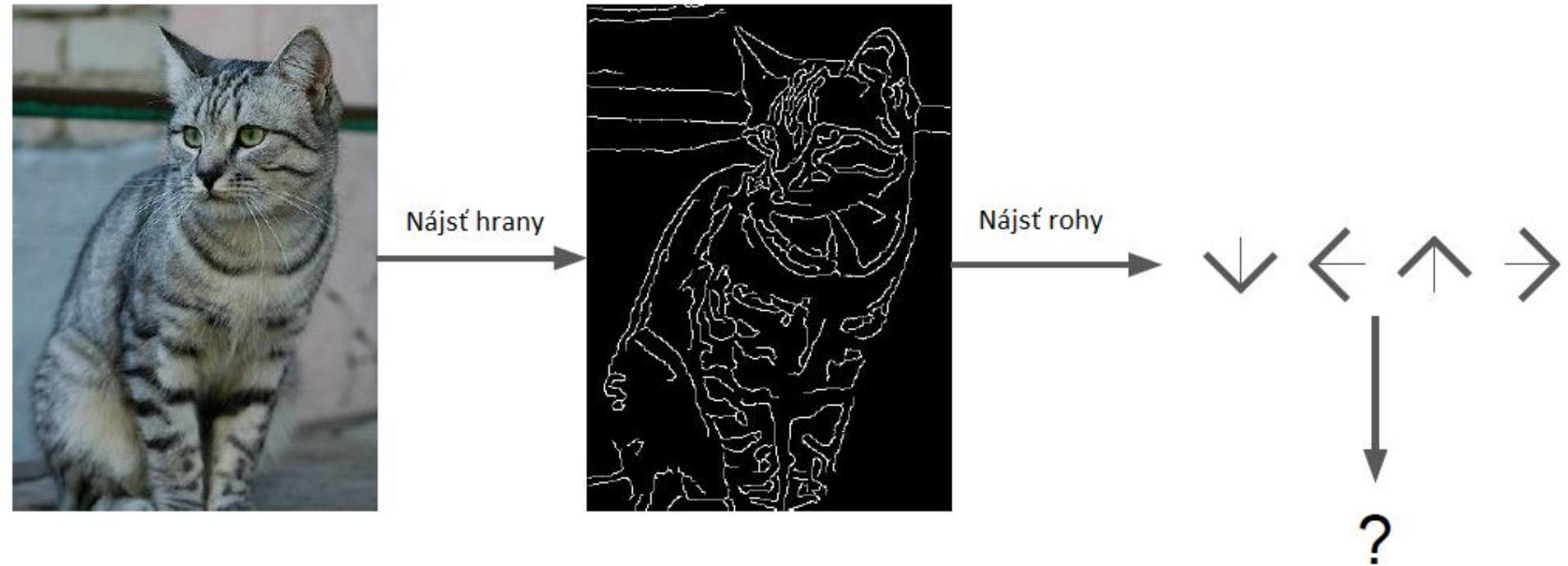
Klasifikátor

- Chceme, aby robil klasifikáciu obrazu, t.j. jeho vstupom je obraz a výstupom označenie triedy

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

- Na rozdiel napr. od triedenia zoznamu čísiel, **nie je žiadny zrejmý postup**, ako naprogramovať algoritmus pre rozpoznávanie mačky, alebo iných tried obrazov

Klasický (príznakový) prístup



Prístup strojového učenia (ML)

Dátovo orientovaný prístup

- Získajte databázu obrazov a ich tried (labels)
- Využite ML na trénovanie klasifikátora
- Vyhodnoťte klasifikátor na nových obrazoch

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

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

Ukážka trénovacej množiny

airplane



automobile



bird



cat



deer



1. klasifikátor: najbližší sused

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



Zapamätaj si všetky obrazy a ich triedy

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



Zarad' obraz do triedy, do ktorej patrí najpo-dobnejší trénovací obraz

Ukážka databázy: CIFAR10

- 10 tried obrazov
- 50.000 trénovacích
- 10.000 testovacích



Ukážka databázy: CIFAR10

- 10 tried obrazov
- 50.000 trénovacích
- 10.000 testovacích



Testovacie obrazy a ich
najbližší susedia



Metrika na určenie najbližšieho

- Metrika na porovnávanie obrazov

L1 metrika:

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

Testovací obraz

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

Trénovací obraz

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

-

Absolútна hodnota rozdielov po pixloch

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

Spolu → 456

Klasifikátor najbližšieho suseda

```
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
```

Klasifikátor najbližšieho suseda

```
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
```

Zapamätaj si
trénovacie dátá

Klasifikátor najbližšieho suseda

```
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
```

Zapamätaj si
trénovacie dátá

Pre každý testovací
obraz: Nájdi najbližší
trénovací obraz
Jeho triedu prirad'
ako výstup

Aká je rýchlosť klasifikátora?

```
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
```

Trénovanie O(1)
Klasifikácia O(N)

To je ZLE
Chceme klasifikátor,
ktoré je **rýchly** pri
klasifikácii.
Ak je **pomalý** pri
trénovaní, to je OK

Ako to vyzerá v praxi?



K najbližších susedov

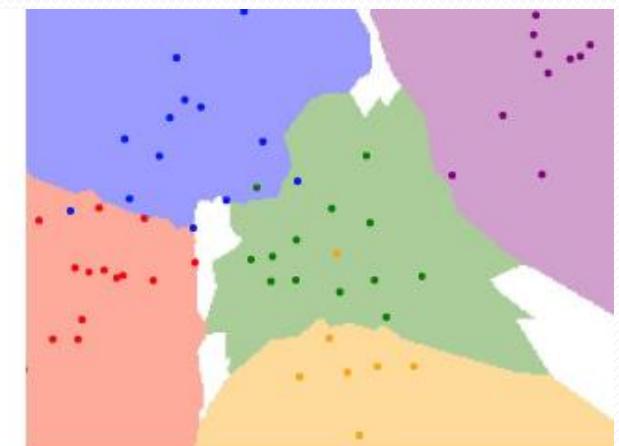
Namiesto prevzatia triedy od najbližšieho suseda, výsledok klasifikácie určí **väčšinové hlasovanie** z K najbližších bodov (biele plochy, reprezentujú remízu)



$K = 1$



$K = 3$



$K = 5$

Ako to vyzerá v praxi?



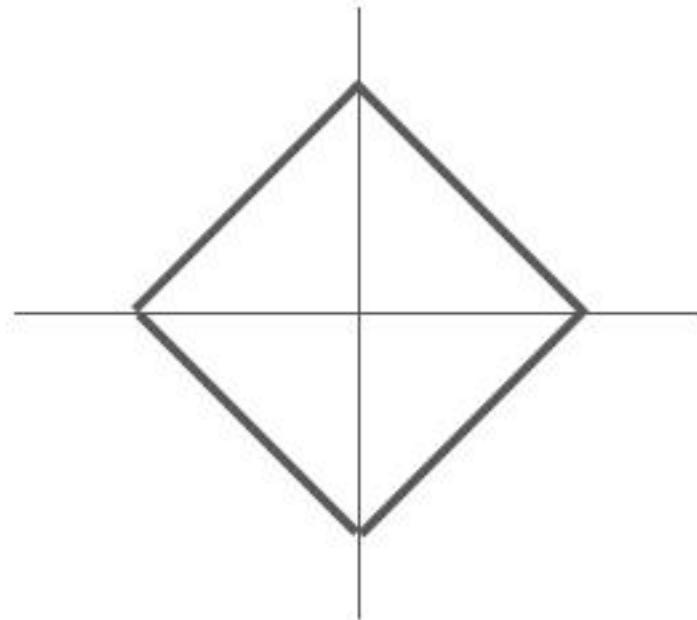
Ako to vyzerá v praxi? II



Metrika pre KNN

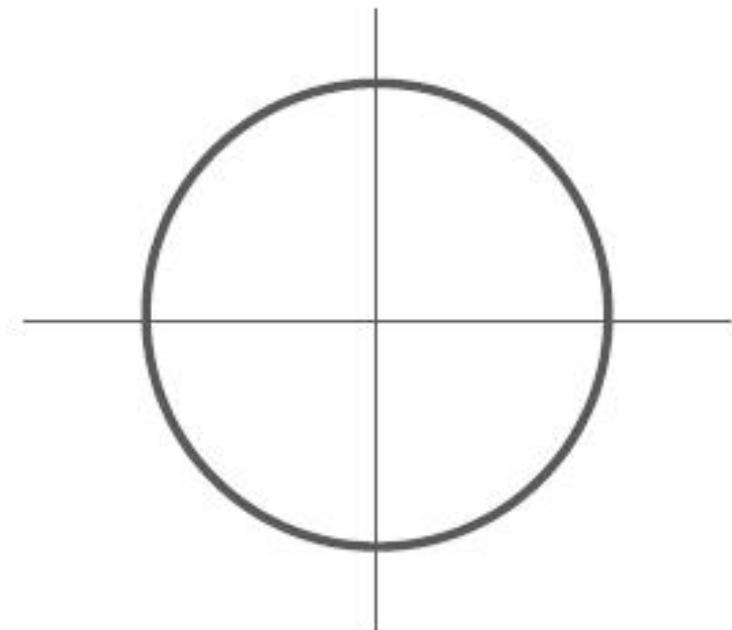
L1 (Manhattanská) metrika

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



L2 (Euklidovská) metrika

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



Metrika pre KNN II

L1 (Manhattanská) metrika

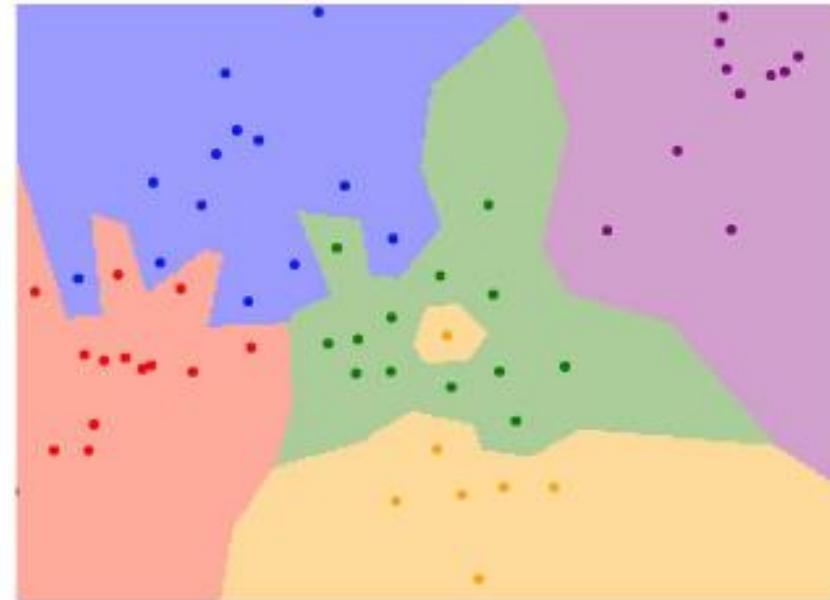
$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



$K = 1$

L2 (Euklidovská) metrika

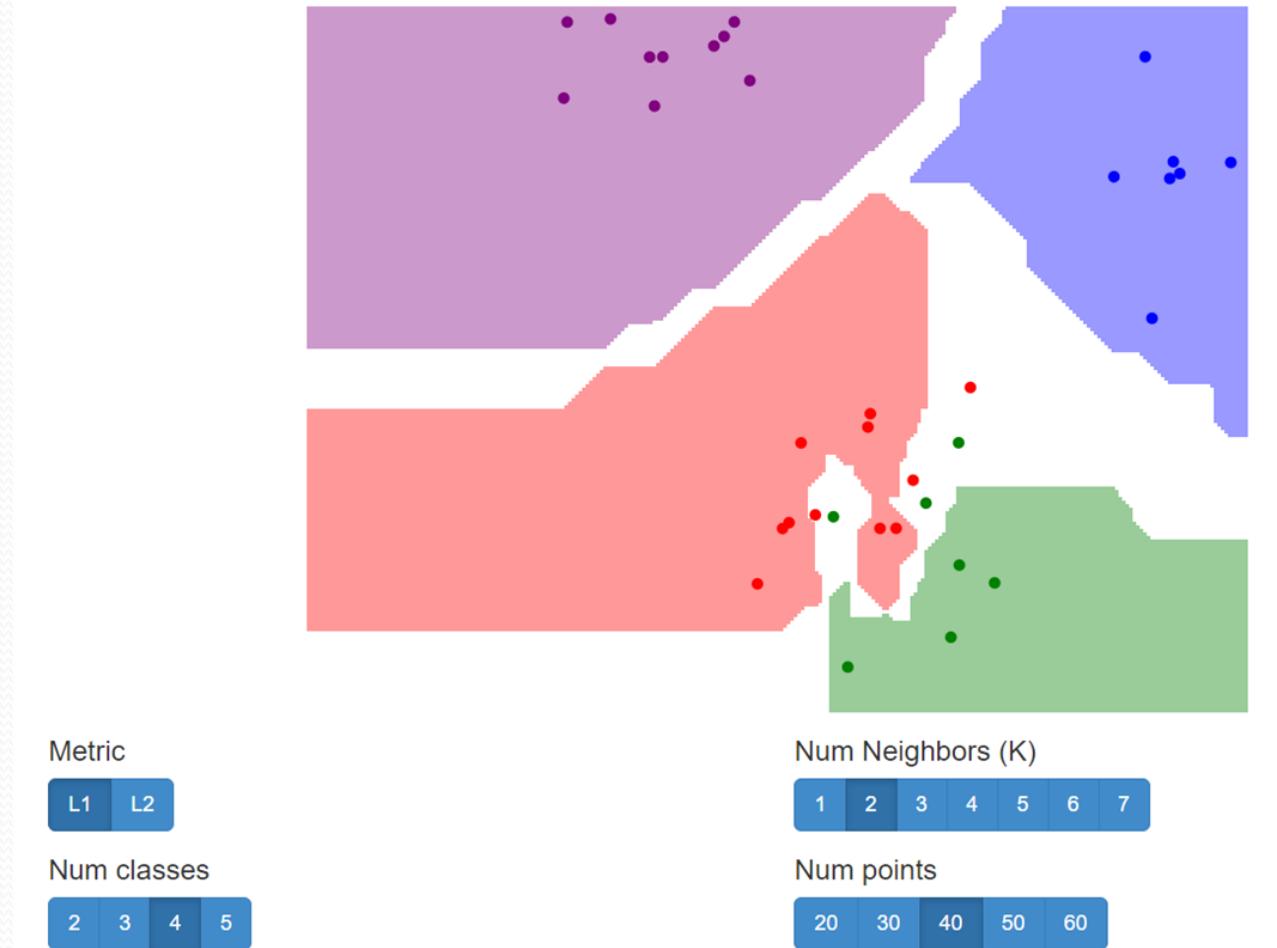
$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$



$K = 1$

KNN vyskúšajte si

<http://vision.stanford.edu/teaching/cs231n-demos/knn/>



Hyperparametre

Aká je najlepšia hodnota pre K ?

Akú je najlepšie zvoliť metriku?

Toto sú **hyperparametre**: to sú také parametre algoritmu, ktoré nastavuje používateľ dopredu a nenastavujú sa počas trénovania

Hyperparametre

Aká je najlepšia hodnota pre K ?

Akú je najlepšie zvoliť metriku?

Toto sú **hyperparametre**: to sú také parametre algoritmu, ktoré nastavuje používateľ dopredu a nenastavujú sa počas trénovania

Správne hodnoty veľmi závisia od problému, ktorý riešime. Treba skúsiť všetky možné a potom vybrať hodnoty, čo najlepšie fungujú

Nastavenie hyperparametrov

Nápad č. 1 Zvoliť hyperparametre,
ktoré najlepšie fungujú na dátach

Vaša databáza

Nastavenie hyperparametrov

Nápad č. 1 Zvoliť hyperparametre,
ktoré najlepšie fungujú na dátach

ZLÝ nápad: $K = 1$ funguje
vždy na trénovacích dátach

Vaša databáza

Nastavenie hyperparametrov

Nápad č. 1 Zvoliť hyperparametre, ktoré najlepšie fungujú na dátach

ZLÝ nápad: $K = 1$ funguje vždy na trénovacích dátach

Vaša databáza

Nápad č. 2 Rozdeliť dáta: na **trénovacie** a **testovacie** a zvoliť také hyperparametre, ktoré najlepšie fungujú na testovacích

Trénovacie dáta

Testovacie

Nastavenie hyperparametrov

Nápad č. 1 Zvoliť hyperparametre, ktoré najlepšie fungujú na dátach

ZLÝ nápad: $K = 1$ funguje vždy na trénovacích dátach

Vaša databáza

Nápad č. 2 Rozdeliť dáta: na **trénovacie** a **testovacie** a zvoliť také hyperparametre, ktoré najlepšie fungujú na testovacích

ZLÝ nápad: nevieme, ako bude fungovať algoritmus na nových dátach

Trénovacie dáta

Testovacie

Nastavenie hyperparametrov

Nápad č. 1 Zvoliť hyperparametre, ktoré najlepšie fungujú na dátach

ZLÝ nápad: $K = 1$ funguje vždy na trénovacích dátach

Vaša databáza

Nápad č. 2 Rozdeliť dáta: na **trénovacie** a **testovacie** a zvoliť také hyperparametre, ktoré najlepšie fungujú na testovacích

ZLÝ nápad: nevieme, ako bude fungovať algoritmus na nových dátach

Trénovacie dáta

Testovacie

Nápad č. 3 Rozdeliť dáta: na **trénovacie**, **validačné** a **testovacie**, vybrať hyperparametre na validačných a vyhodnotiť na testovacích

Trénovacie dáta

Validačné dáta

Testovacie

Nastavenie hyperparametrov

Nápad č. 1 Zvoliť hyperparametre, ktoré najlepšie fungujú na dátach

ZLÝ nápad: $K = 1$ funguje vždy na trénovacích dátach

Vaša databáza

Nápad č. 2 Rozdeliť dáta: na **trénovacie** a **testovacie** a zvoliť také hyperparametre, ktoré najlepšie fungujú na testovacích

ZLÝ nápad: nevieme, ako bude fungovať algoritmus na nových dátach

Trénovacie dáta

Testovacie

Nápad č. 3 Rozdeliť dáta: na **trénovacie**, **validačné** a **testovacie**, vybrať hyperparametre na validačných a vyhodnotiť na testovacích

LEPŠIE!!!

Trénovacie dáta

Validačné dáta

Testovacie

Setting Hyperparameters

train

Idea #4: Cross-Validation: Split data into **folds**,
try each fold as validation and average the results

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

Useful for small datasets, but not used too frequently in deep learning

KNN sa na obrazoch nepoužíva

Je veľmi pomalý pri klasifikácii

Metriky nie sú na pixloch dostatočne informatívne

Originál



So zakrytými časťami



Posunutý



Tónovaný



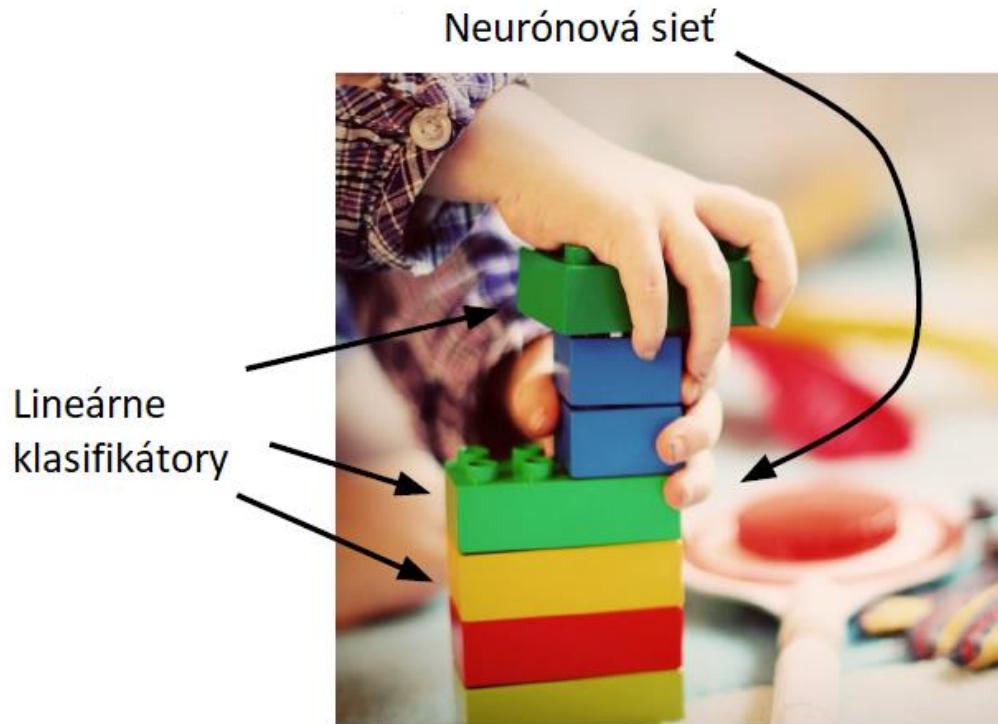
(všetky tri obrazy majú rovnakú L2 vzdialenosť k originálu)

KNN summarizácia

- Pri **klasifikácii obrazu** začíname s trénovacou množinou obrazov a ich zaradením do tried a klasifikujeme neznáme obrazy v testovacej
- KNN klasifikuje obraz na základe zaradenia najbližších susedov z trénovacej množiny
- Metrika a parameter K sú hyperparametre
- Na výber hyper parametrov používame validačné dát a skutočne **testujeme na konci iba raz**

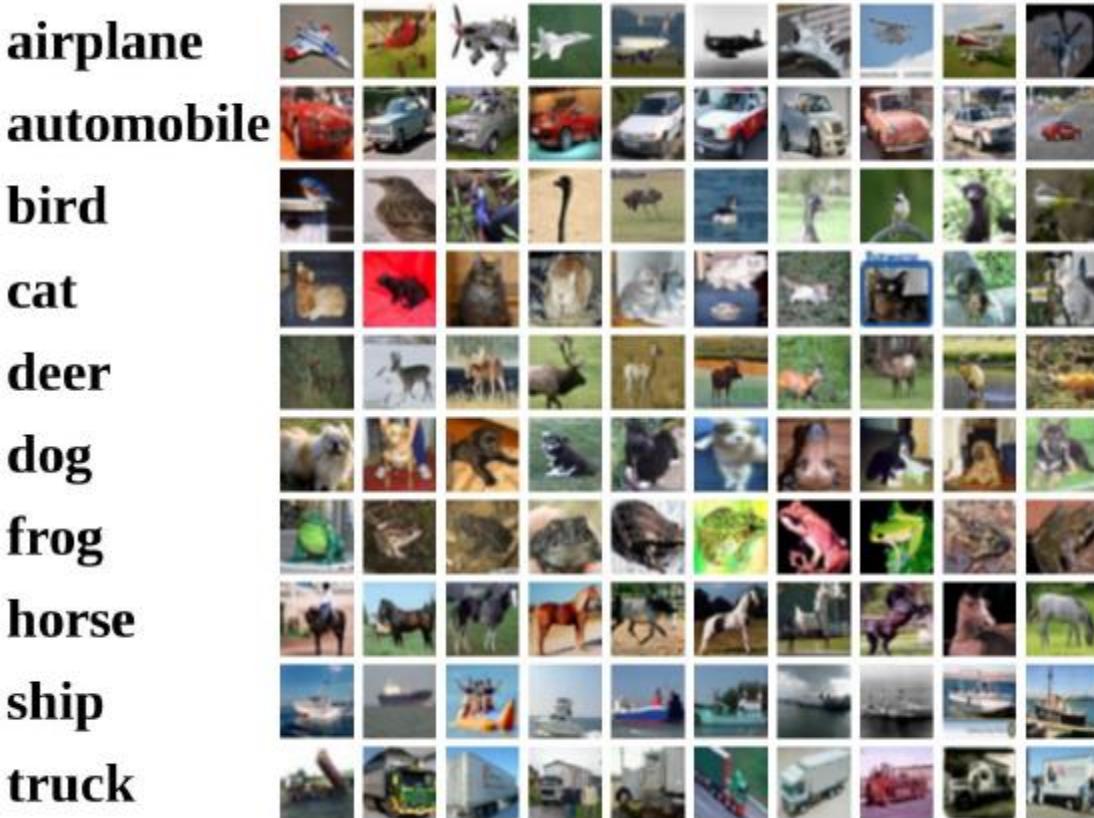
Lineárna klasifikácia

- Je dôležitá preto, že neurónová siet' sa skladá z lineárnych klasifikátorov, ktoré spájame ako lego kocky



- Lineárny klasifikátor bude fungovať vtedy, keď triedy obrazov sú lineárne separovateľné (oddeliteľné)

Znovu použijeme CIFAR 10



50.000

trénovacích obrazov

Každý obraz je

32x32x3

10.000

testovacích obrazov

Parametrický prístup

Obraz x



$f(x, W)$



10 čísel
vyjadrujúcich
skóre triedy



W

parametre
alebo váhy

Pole **32x32x3** čísel
(Spolu 3072 čísel)

Lineárny je vtedy, keď

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

Obraz \mathbf{x}



\mathbf{W}
parametre
alebo váhy

Pole **32x32x3** čísel
(Spolu 3072 čísel)

Parametrický prístup

3072x1

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

Obraz \mathbf{x}



10x1

10x3072

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



\mathbf{W}

parametre
alebo váhy

10 čísel
vyjadrujúcich
skóre triedy

Pole **32x32x3** čísel
(Spolu 3072 čísel)

Parametrický prístup

3072x1

$$f(\mathbf{x}, \mathbf{W}) = \boxed{\mathbf{W} \mathbf{x}} + \boxed{\mathbf{b}} \quad 10 \times 1$$

Obraz \mathbf{x}



10x1

10x3072



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



10 čísel
vyjadrujúcich
skóre triedy

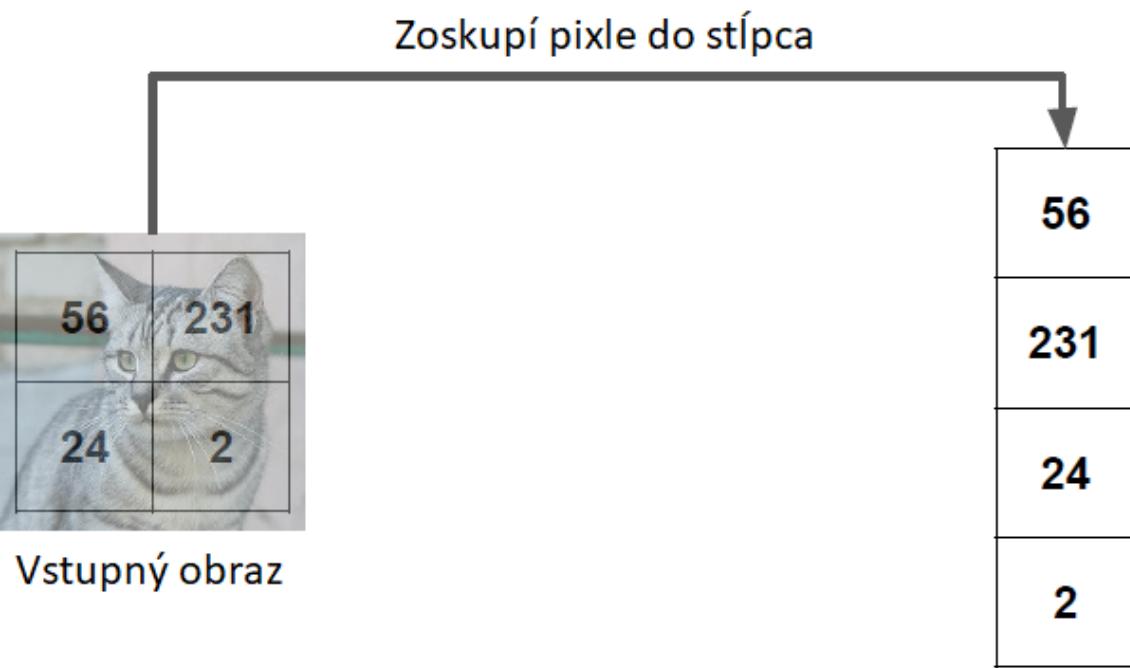
\mathbf{W}

parametre
alebo váhy

Pole **32x32x3** čísel
(Spolu 3072 čísel)

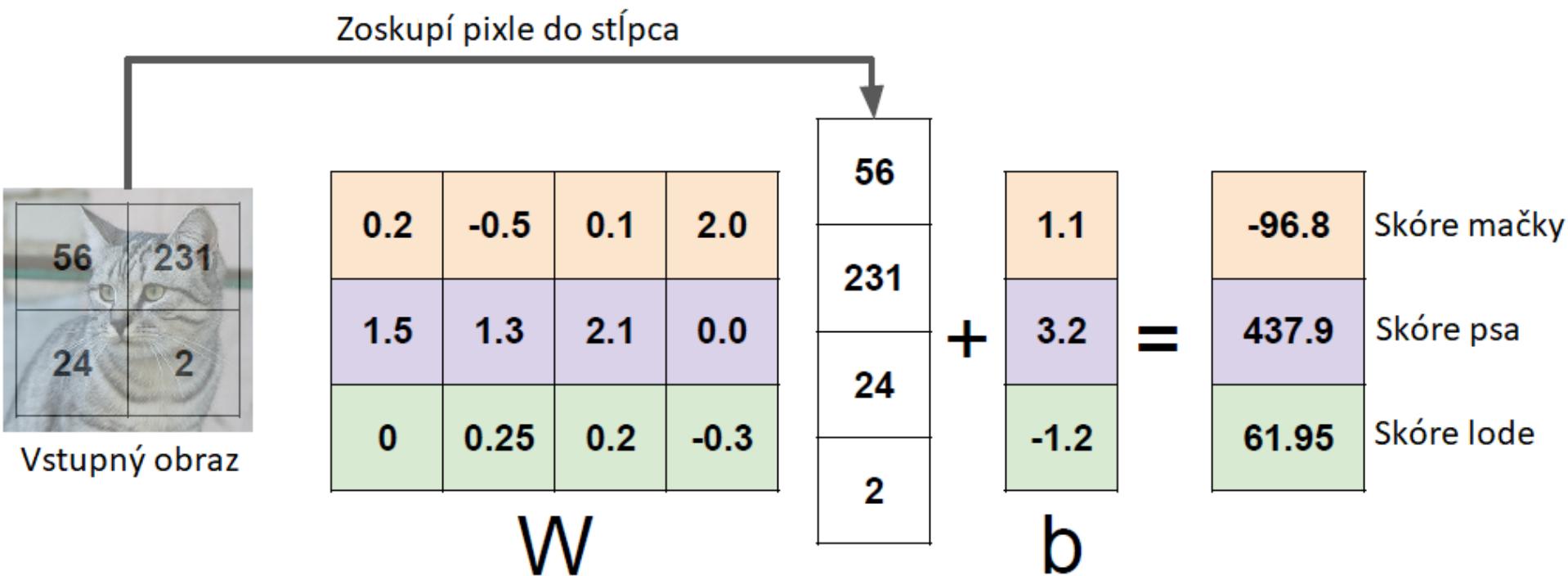
Príklad

- 4 pixle, 3 triedy (mačka, pes, lod')



Príklad

- 4 pixle, 3 triedy (mačka, pes, lod')

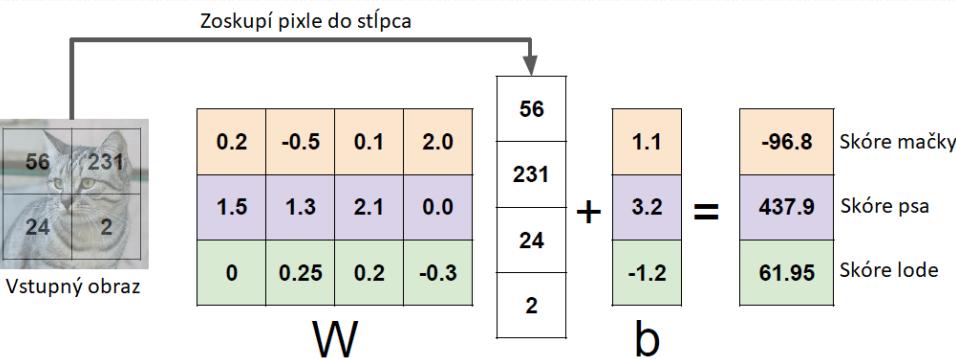


Príklad

- 4 pixle, 3 triedy (mačka, pes, lod')

Algebraický pohľad

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

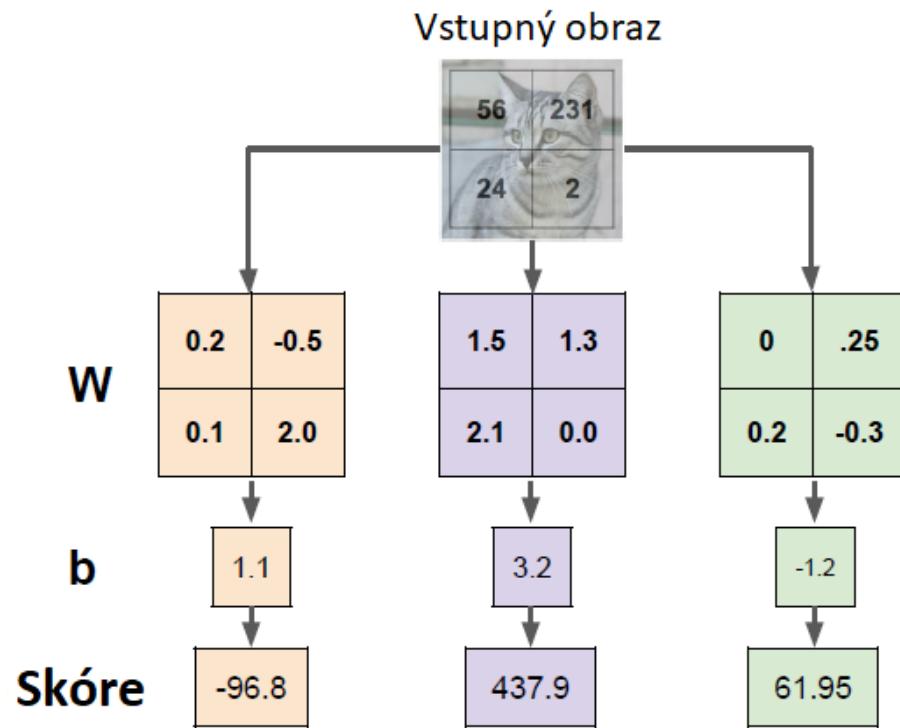
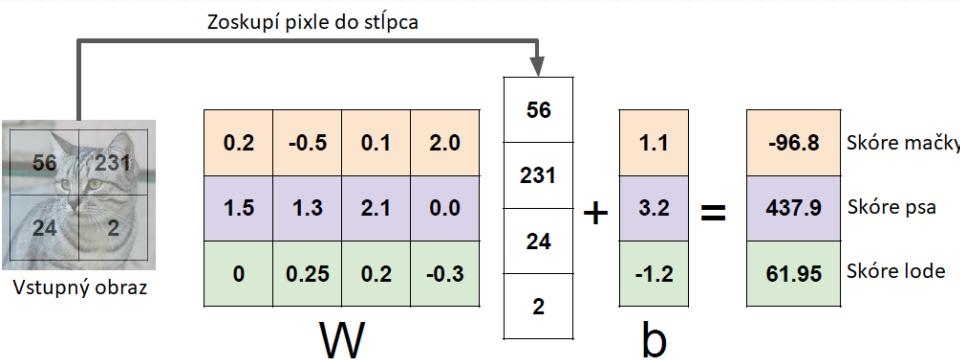


Príklad

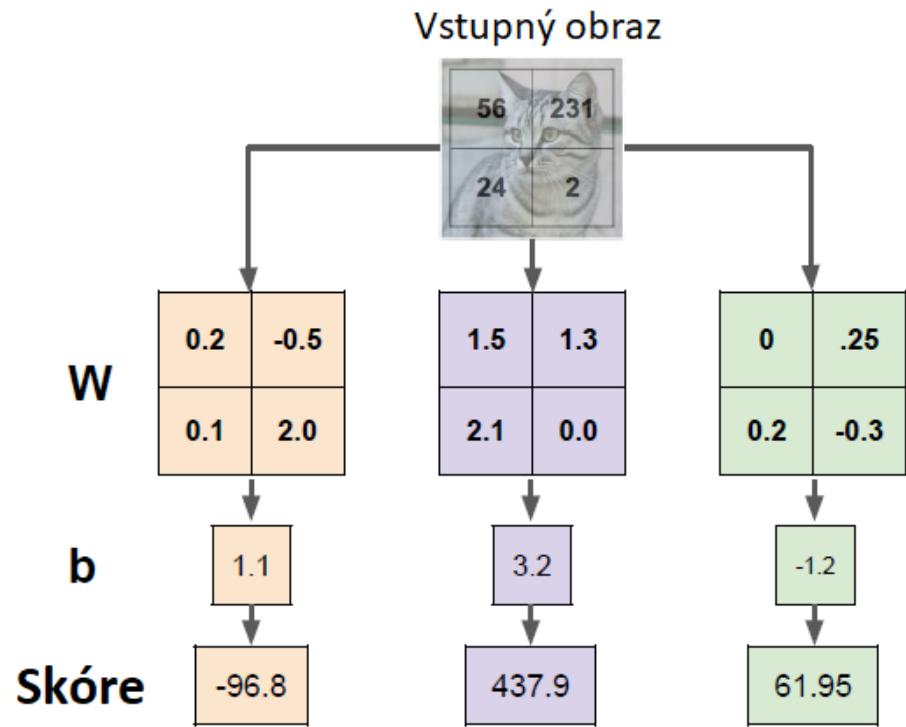
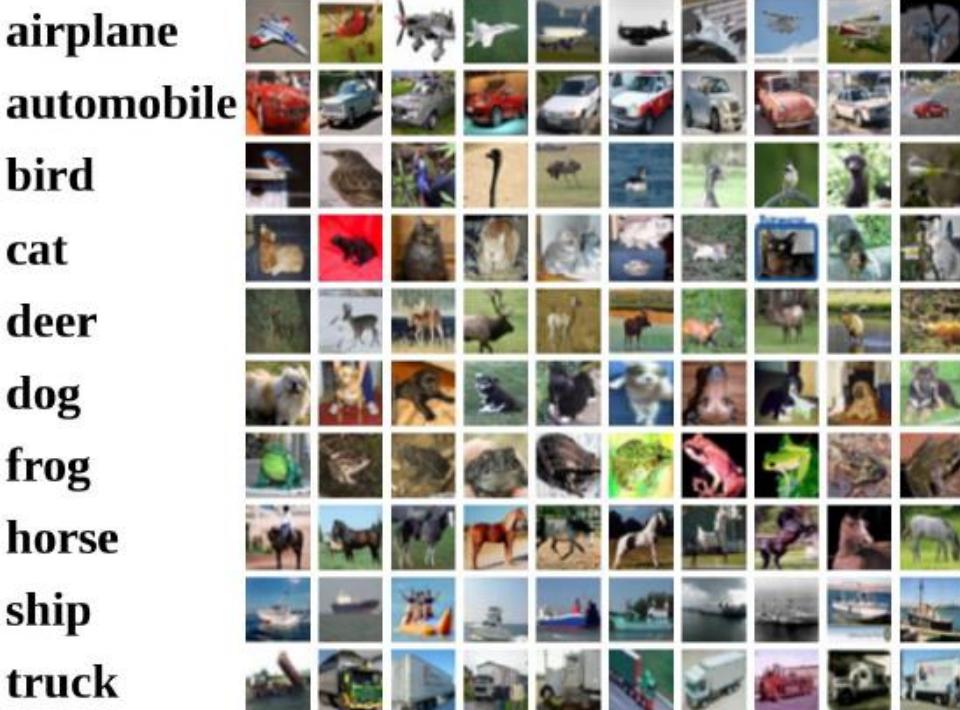
- 4 pixle, 3 triedy (mačka, pes, lod')

Algebraický pohľad

$$f(\mathbf{x}, \mathbf{W}) = \mathbf{W}\mathbf{x} + \mathbf{b}$$

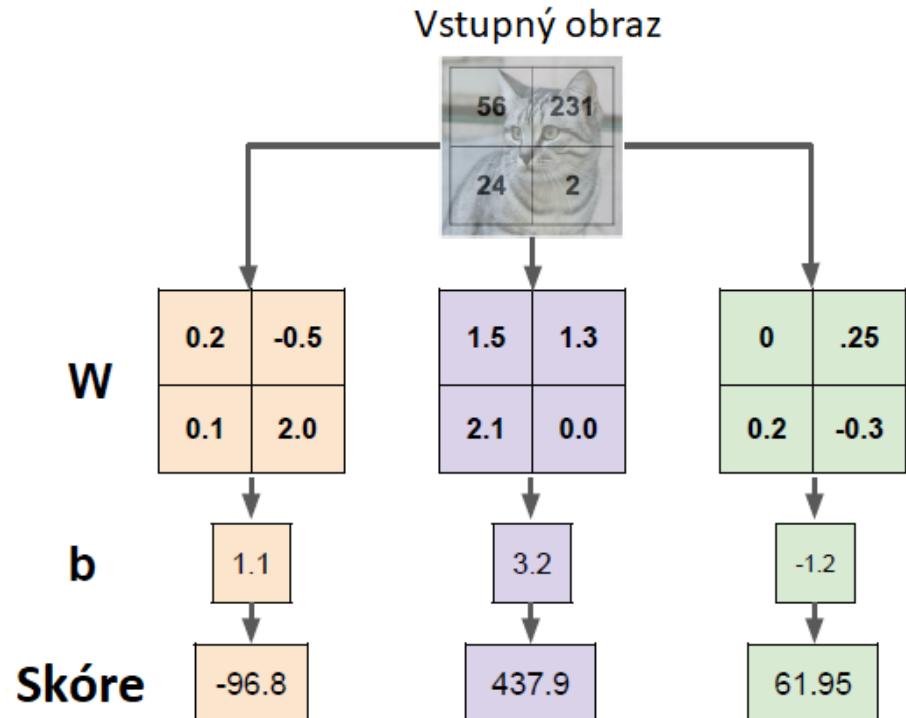
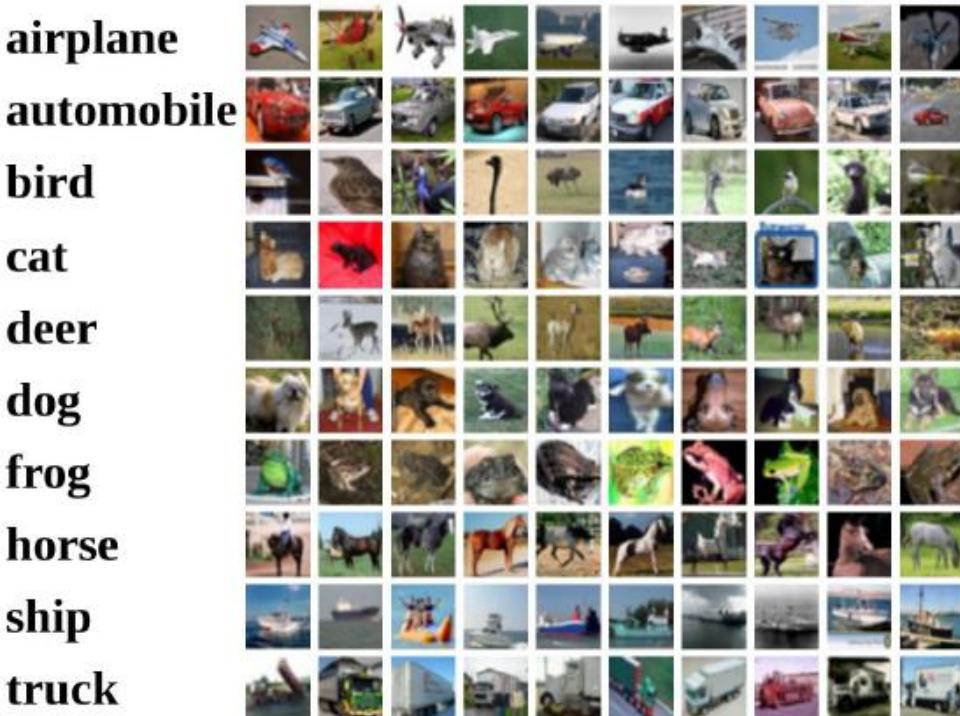


Interpretácia lineárneho klasifikátora



Interpretácia lineárneho klasifikátora

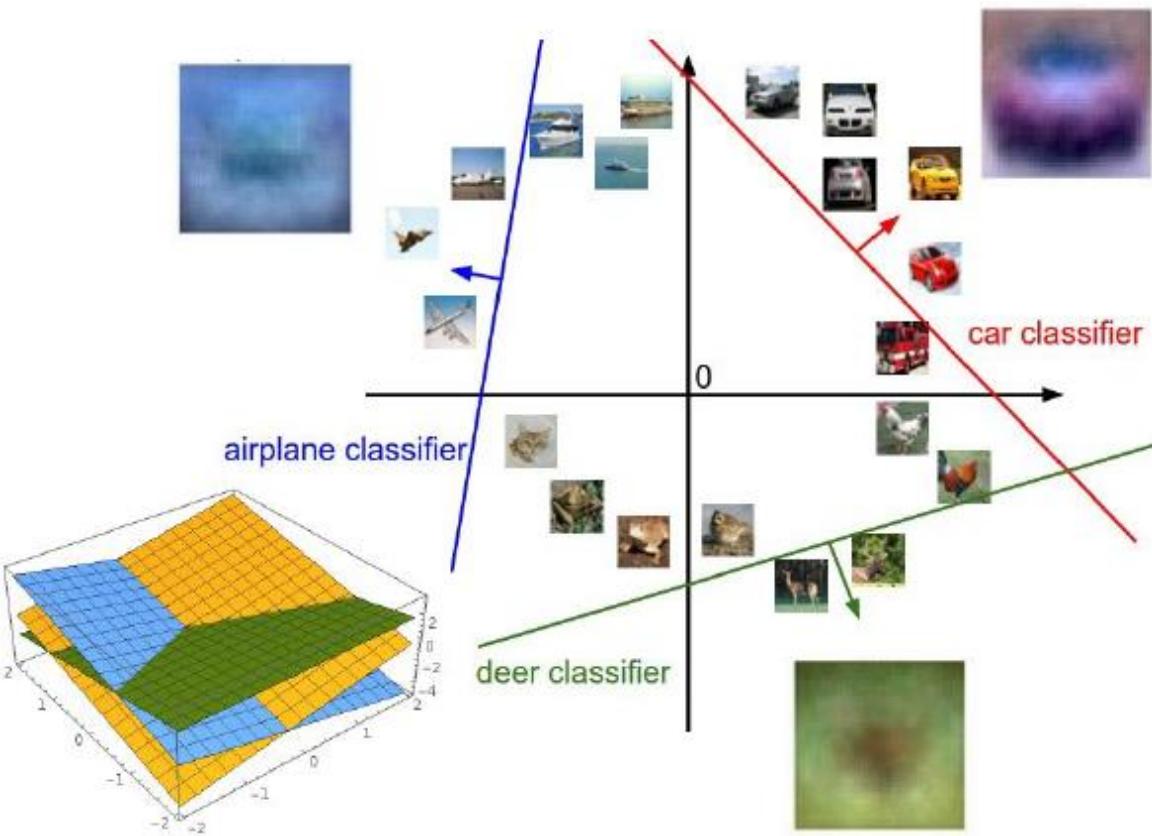
• Vizuálny pohľad



Interpretácia lineárneho klasifikátora

- Geometrický pohľad

$$f(\mathbf{x}, \mathbf{W}) = \mathbf{W}\mathbf{x} + \mathbf{b}$$



Z toho vypočítame oddelujúce nadroviny, kde pre dve triedy \mathbf{W} predstavuje otočenie a \mathbf{b} posunutie voči 0

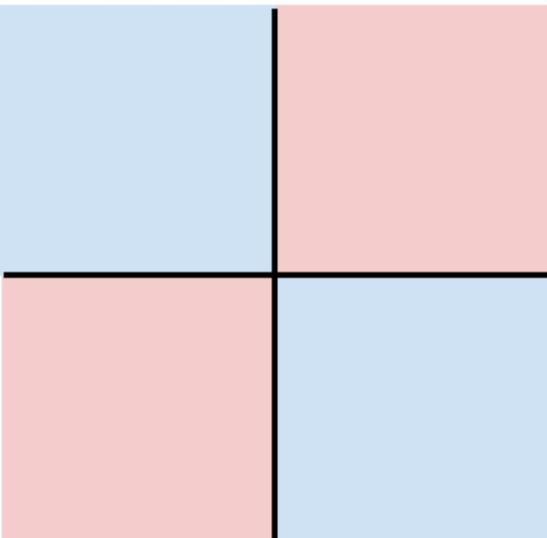
Ako klasifikovať nelineárny prípad?

Trieda 1

1. a 3. Q

Trieda 2

2. a 4. Q

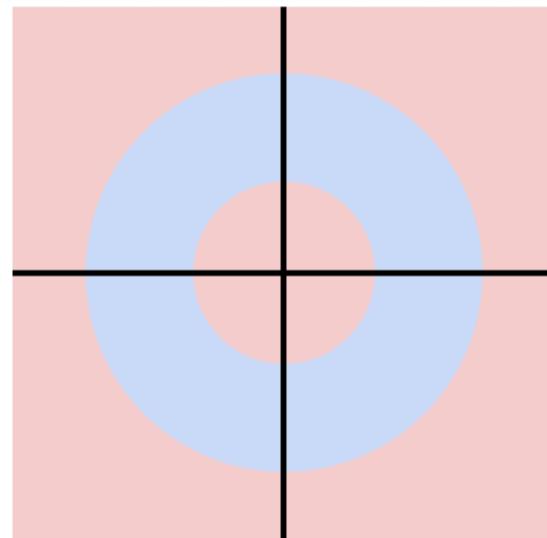


Trieda 1

$1 \leq L2\ norm \leq 2$

Trieda 2

Všetko ostatné

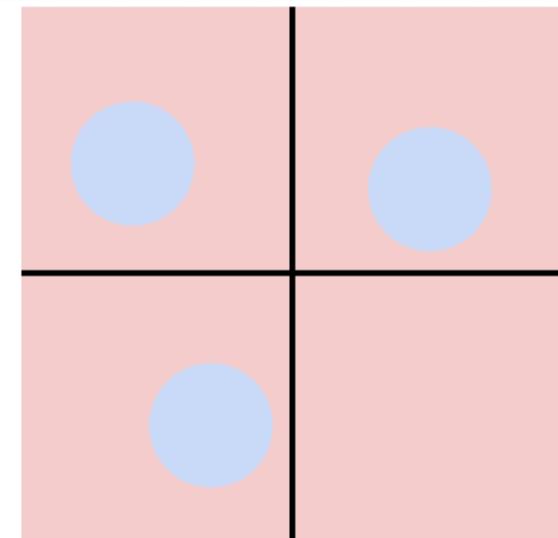


Trieda 1

3 kruhy

Trieda 2

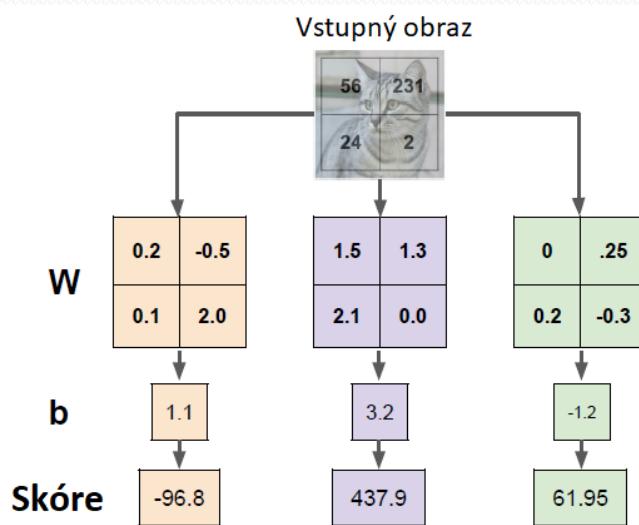
Všetko ostatné



Tri pohľady na lineárny klasifikátor

Algebraický

$$f(\mathbf{x}, \mathbf{W}) = \mathbf{W}\mathbf{x} + \mathbf{b}$$



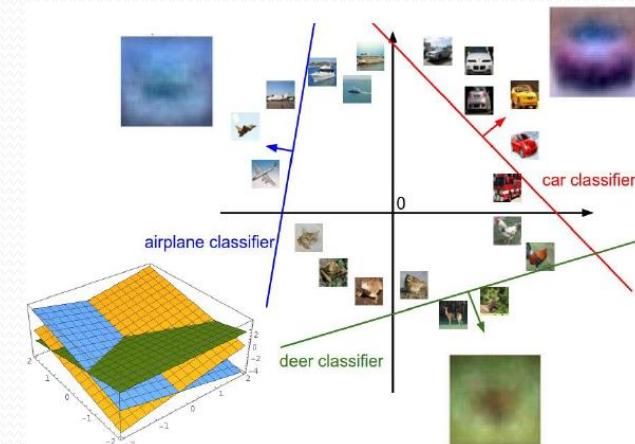
Vizuálny

Jeden vzor
pre triedu



Geometrický

Nadroviny rozde-
ľujúce priestor



Čo doteraz vieme?

Definovali sme (lineárnu) funkciu $f(\mathbf{x}, \mathbf{W}) = \mathbf{W}\mathbf{x} + \mathbf{b}$ na ohodnotenie skóre

Tu je ukážka skóre pre 3 triedy pre nejakú hodnotu \mathbf{W}

Ako vieme povedať, či toto \mathbf{W} je dobré alebo zlé?



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

Čo nás čaká najbližšie?

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

- **Stratová funkcia** (kvantifikujeme, čo znamená „dobre“ W)
- **Optimalizácia** (začneme s náhodným W a nájdeme také, čo minimalizuje stratu)
- **Konvolučné siete** (doladíme funkčnú formu pre f)