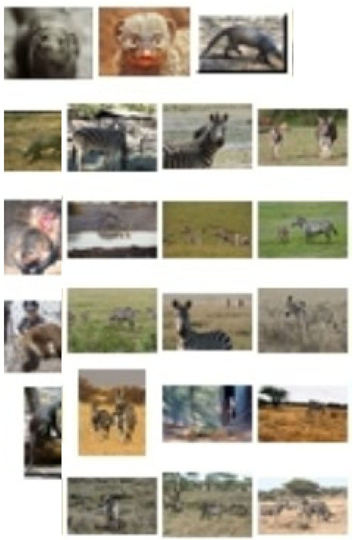
Linear classifier

ISTD 50.035

Computer Vision

Acknowledgement: Some images are from various sources: UCF, Stanford cs231n, etc.

Data driven approach



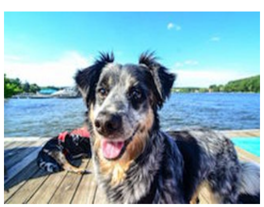
Training/Learning (usually offline)

Training set: images with known class information

Learn a model using some algorithm

A model for this specific classification problem and classifier

Testing/Evaluation (usually online)



Testing set (with label during evaluation, without label in an application)

Classifier algorithm

Predicted class information for this new image (compare with ground-truth during evaluation)

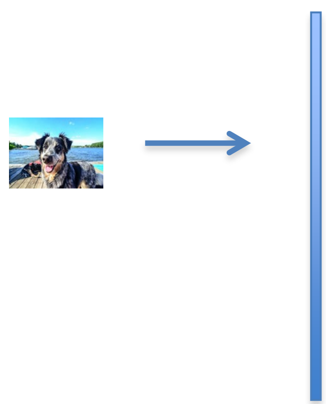
For practically use, testing time should be small

Linear Classifier

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

Score function

- Given a test image x, produce the confidence score for each class using linear transformation (total: K classes)
- Higher confidence score for a class -> more likely to be the ground-truth class
- Test image x: flatten to a Dx1 column vector, D is the image resolution times number of channel

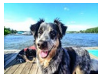


Input x: Dx1
Weight W: KxD
Bias b: Kx1
Score s:Kx1

Linear Classifier

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

- **Testing**: W, b are fixed, x is the input
- **Training**: Given N training samples (x_i, y_i), y_i takes value in [1,...,K], learn W and b
- The ground truth class is y_i

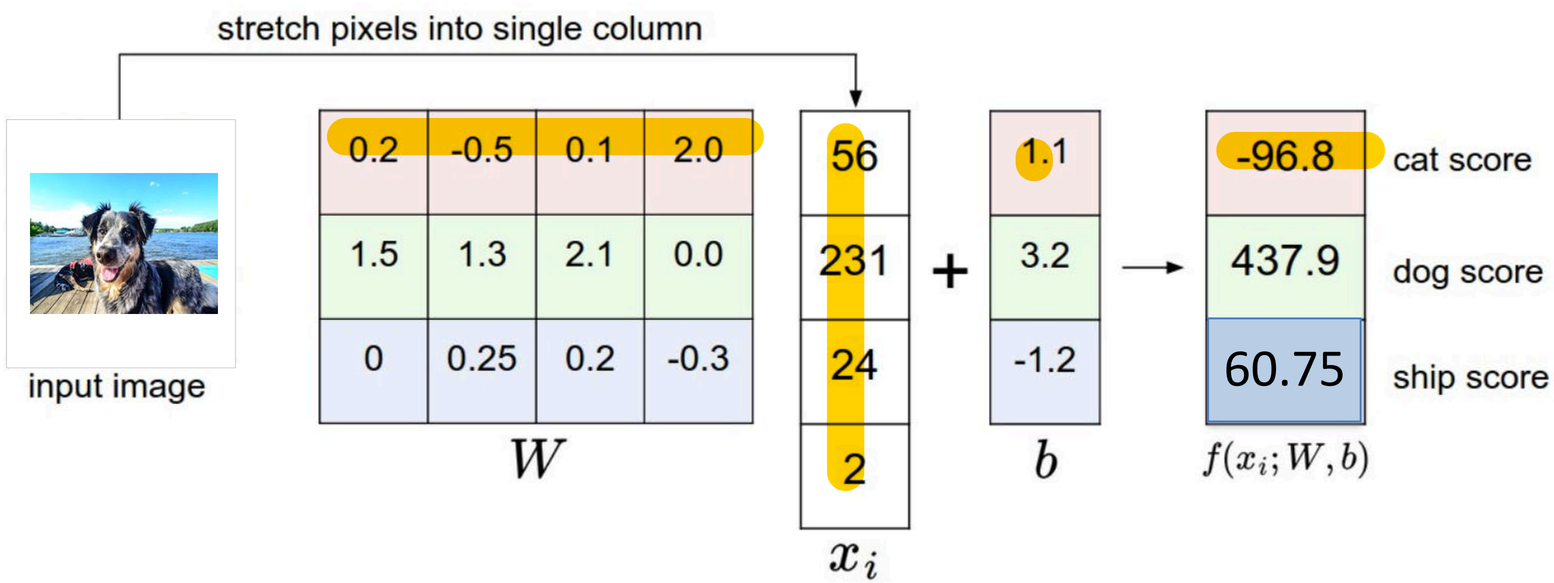
Example: K=3, x_i=  y_i = 2 {cat, dog, ship}

Training: (x_i,y_i) are given and fixed; W, b are the variables to be determined

Linear Classifier

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

- **Testing**: W, b are fixed, x is the input



- Training: learn W, b to discriminate the classes
- Each row of W extracts the features (template) of a specific class from input

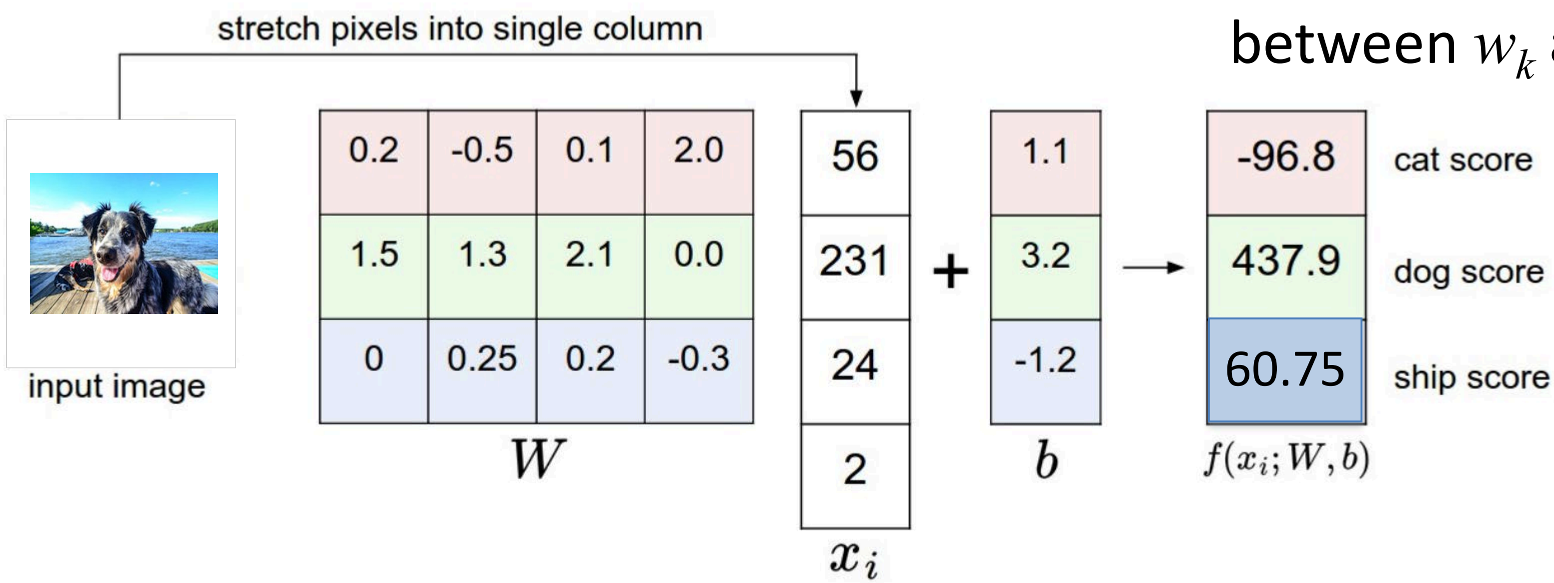
Linear Classifier

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

$$w_k^T x_i = \|w_k\| \|x_i\| \cos\theta$$

- **Testing**: W, b are fixed, x is the input

Measure similarity between w_k and x_i



- Training: learn W, b to discriminate the classes
- Each row of W extracts the features (template) of a specific class from input

Linear Classifier

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

- **Shorthand notation**

$$s = [W\ b][x\ 1]^T$$

$$W \qquad x$$

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

Input x: (D+1)x1
Weight W: Kx(D+1)
Score s:Kx1

Loss function

- **Training**: Given N training samples (x_i, y_i), y_i takes value in [1,...,K], learn W
- Loss function: measure how consistent are the ground-truth labels and the score function outputs, for some W
- Small loss: good W
- Softmax classifier with cross-entropy loss
- Multiclass Support Vector Machine (SVM) loss

Softmax classifier

- Regard output of the score function f(x; W) as the *unnormalized log probability* of each class

- Probability of each class can be obtained by applying a softmax function (exp, then normalize):

$$\text{softmax}(f) = \frac{e^{f_m}}{\sum_{j=1}^{K} e^{f_j}}$$

For the m-th class

- Cross-entropy loss (apply –log(.) to only the ground-truth class):

$$L_i = -\log \frac{e^{f_{y_i}}}{\sum_{j=1}^{K} e^{f_j}}$$

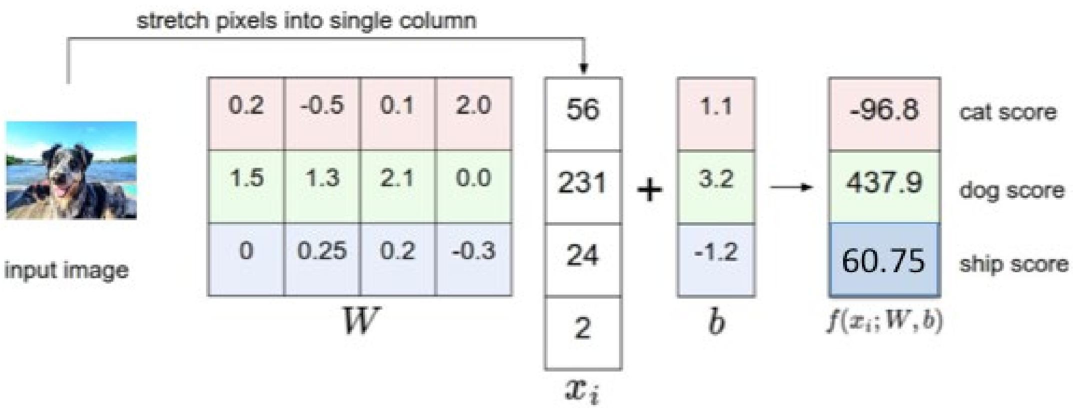
For the i-th training sample

Softmax classifier

- Cross-entropy loss (apply –log(.) to only the ground-truth class):

$$L_i = -\log \frac{e^{f_{y_i}}}{\sum_{j=1}^{K} e^{f_j}}$$

For the i-th training sample



Example: a dog (which looks like a dog)

```
print(np.exp(f))
print(np.exp(f) / sum(np.exp(f)))

[ 9.12628762e-043   1.50505935e+190   2.41762966e+026]
[ 6.06373937e-233   1.00000000e+000   1.60633510e-164]
```

y_i=2

L_i = -log(1) = 0

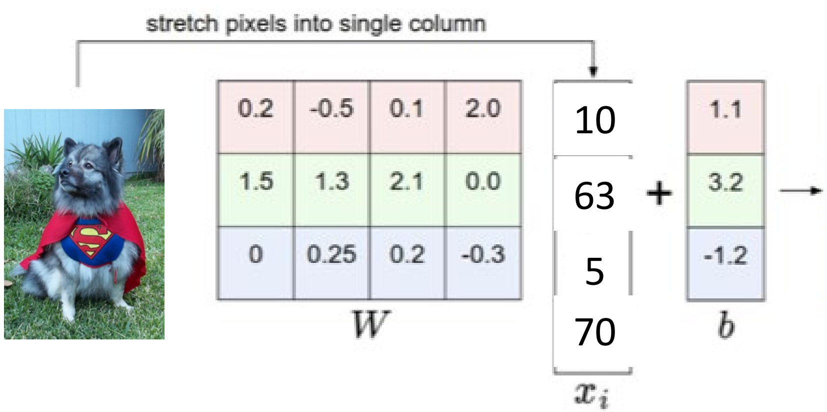
← Probability

Softmax classifier

- Cross-entropy loss (apply –log(.) to only the ground-truth class):

$$L_i = -\log \frac{e^{f_{y_i}}}{\sum_{j=1}^{K} e^{f_j}}$$

For the i-th training sample



Example: a dog (which does not look like a dog)

Softmax classifier

- The entire loss for N training samples (x_i, y_i):

$$L = \frac{1}{N} \sum_i L_i$$

- We determine W to minimize this loss given the training dataset

- Additional regularization of W

Understand Softmax classifier

- Cross-entropy loss
  - First apply softmax function
  - Then apply –log(.) to only the ground-truth class

$$L_i = -\log \frac{e^{f_{y_i}}}{\sum_{j=1}^{K} e^{f_j}}$$

For the i-th training sample

- The term $\frac{e^{f_{y_i}}}{\sum_{j=1}^{K} e^{f_j}}$ is the probability of the correct class (i.e., y_i)

- Therefore, want this to be large, i.e., $\max_W$ log(.)
- Thus, want this to be small $\min_W$ –log(.)

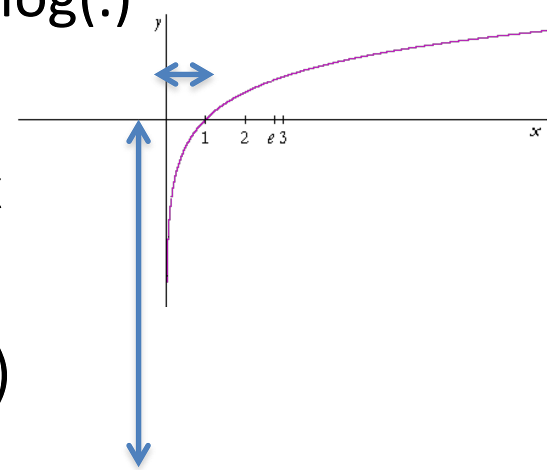
Understand Softmax classifier

$$L_i = -\log \frac{e^{f_{y_i}}}{\sum_{j=1}^{K} e^{f_j}}$$

- Why min -log(.) or max log(.)

- The term $\frac{e^{f_{y_i}}}{\sum_{j=1}^{K} e^{f_j}}$ is the probability, between [0,1]

log(.)



- Stretch the numerical range during min/max
- Often used when working with probability
- p1xp2 is small: log(p1xp2) = log(p1) + log(p2)
- Maximum Likelihood Estimation (MLE)
  – Minimize the negative log likelihood of the correct class

Understand Softmax classifier

$$L_i = -\log \frac{e^{f_{y_i}}}{\sum_{j=1}^{K} e^{f_j}}$$

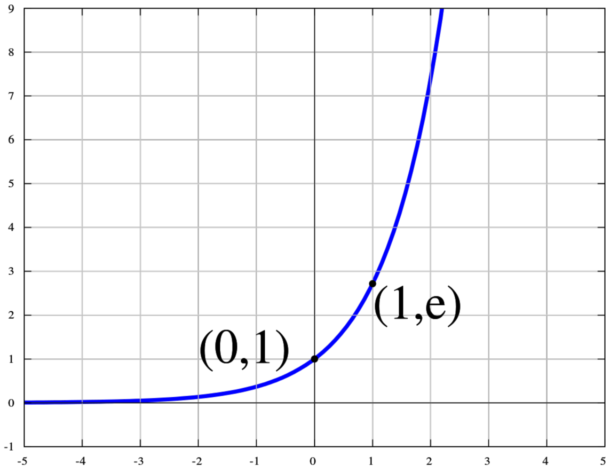
- Why exp before normalization?
- Much higher confidence if the activation is large (clear images)

```
s = np.array([1,2])
print(np.exp(s) / sum(np.exp(s)))
```

[0.26894142   0.73105858]

```
s = np.array([10,20])
print(np.exp(s) / sum(np.exp(s)))
```

[4.53978687e-05   9.99954602e-01]



Understand Softmax classifier

$$L_i = -\log \frac{e^{f_{y_i}}}{\sum_{j=1}^{K} e^{f_j}}$$

- Another interpretation of the cross entropy loss
- Difference between:
  - The model (estimated) prob Q: $\mathrm{softmax}(f) = \frac{e^{f_m}}{\sum_{j=1}^{K} e^{f_j}}$
  - The data (true) prob P: [0,0,....,1,....,0] (1 at the y_i-th position)
  - Measured by Kullback-Leibler (KL) divergence (D_{KL}=0 when P, Q are “the same”

$$D_{\mathrm{KL}}(P\|Q) = -\sum_i P(i) \log \frac{Q(i)}{P(i)}$$

Want Q to be close to P

Understand Softmax classifier

$$L_i = -\log \frac{e^{f_{y_i}}}{\sum_{j=1}^{K} e^{f_j}}$$

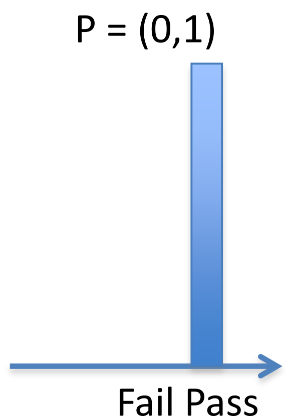
- Another interpretation of the cross entropy loss
- Difference between P and Q as measured by Kullback-Leibler (KL) divergence

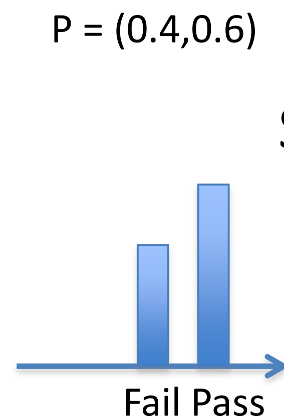
$$\begin{aligned} D_{\mathrm{KL}}(P\|Q) &= -\sum_x p(x)\log q(x) \quad + \quad \sum_x p(x)\log p(x) \\ &= \qquad H(P,Q) \qquad - \qquad H(P) \end{aligned}$$

Cross entropy of P and Q

Entropy of P
H(P) = 0 in this case

Probability, self information, entropy





Self information=

$$\log \frac{1}{P(i)}$$

Information theoretic entropy = average amount of information an observer would gain when sampling a random variable

Rare event has more information

Information ~= “surprise”

$$H(P) = \sum_i P(i) \log \frac{1}{P(i)}$$

bit if base 2; nat for base e

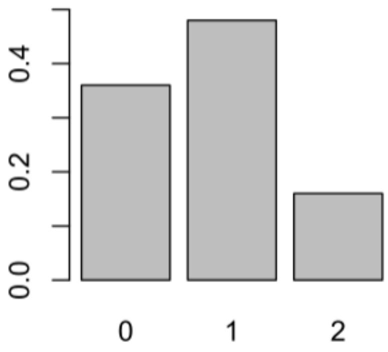
KL Divergence

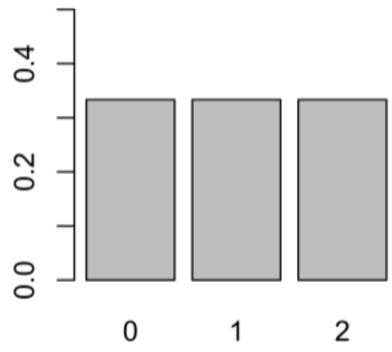
$$D_{\mathrm{KL}}(P\|Q) = -\sum_i P(i) \log \frac{Q(i)}{P(i)} \quad = \sum_i P(i)(\log \frac{1}{Q(i)} - \log \frac{1}{P(i)})$$

- Measure how one probability distribution is different from the other

- $D_{KL}(P||Q) \geq 0$; Equality holds iff P=Q almost everywhere

KL Divergence





x	0	1	2
Distribution *P*(x)	0.36	0.48	0.16
Distribution *Q*(x)	0.333	0.333	0.333

$$
\begin{aligned}
D_{\mathrm{KL}}(P \parallel Q) &= \sum_{x \in \mathcal{X}} P(x) \ln\left(\frac{P(x)}{Q(x)}\right) \\
&= 0.36 \ln\left(\frac{0.36}{0.333}\right) + 0.48 \ln\left(\frac{0.48}{0.333}\right) + 0.16 \ln\left(\frac{0.16}{0.333}\right) \\
&= 0.0852996
\end{aligned}
$$

Is it true: $D_{KL}(P||Q) = D_{KL}(Q||P)$

Softmax classifier

- Additional regularization of W (L2 or L1 norm of weights)

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

- Prefer small $W_{k,l}$, less likely to overfit the training dataset
- Regularize only W, not the bias b
- The entire loss for N training samples (x_i, y_i): data loss and regularization loss

$$L = \frac{1}{N} \sum_i L_i + \lambda R(W)$$

- We determine W to minimize this loss given the training dataset

Multiclass SVM loss

- SVM loss: The correct class has a score higher than the incorrect class by some fixed margin d

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

- For this test image i, **zero score** contributed by class j iff

$0 >= s_j - s_{yi} + d$

$s_{yi} >= s_j + d$

Correct class score s_{yi} is higher than class j score s_j by at least d (otherwise, +ve. contribution of loss from class j)

Multiclass SVM loss

- $S = [13, -7, 11]$
- $y_i = 1$ For test image i
- $d = 10$

$L_i = \max(0, -7-13+10) + \max(0, 11-13+10)$

$= \quad 0 \quad + \quad 8$

- Ground-truth score 13 is higher than -7 by more than the margin d=10
- Ground-truth score 13 is not higher than 11 by d=10

Train W so that the correct class y_i has a score higher than the incorrect classes by at least d

Linear Classifier

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

- After learning of the parameter W, do not need the training data in deployment

- Fast in deployment

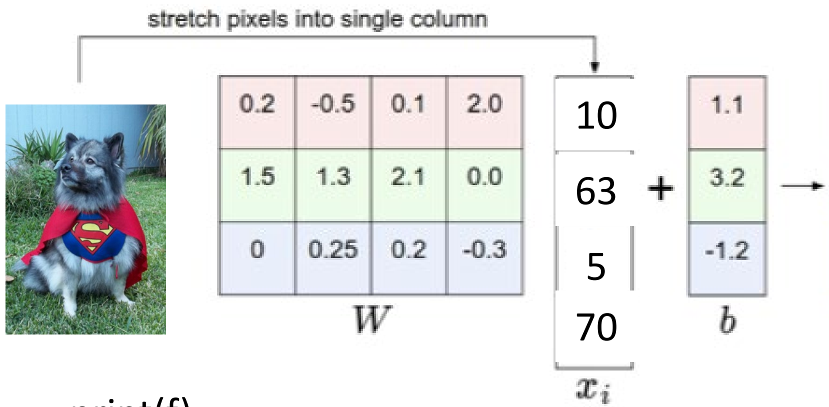
- How to learn W?

Softmax classifier

- Cross-entropy loss (apply –log(.) to only the ground-truth class):

$$L_i = -\log \frac{e^{f_{y_i}}}{\sum_{j=1}^{K} e^{f_j}}$$

For the i-th training sample



Example: a dog (which does not look like a dog)

print(f)
print(np.exp(f))
print(np.exp(f) / sum(np.exp(f)))
[112.1 110.6 -5.45]
[4.83516636e+48 1.07887144e+48 4.29630469e-03]
[8.17574476e-01 1.82425524e-01 7.26458780e-52]

y_i=2
L_i = -log(0.182) = 1.7

← Probability