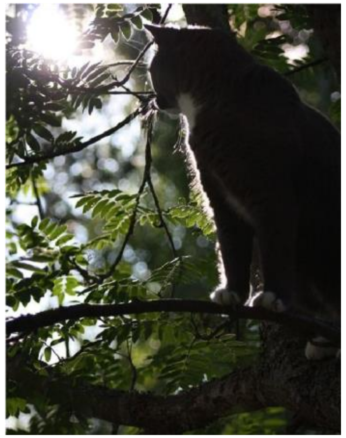
Lecture 2: Linear classifier

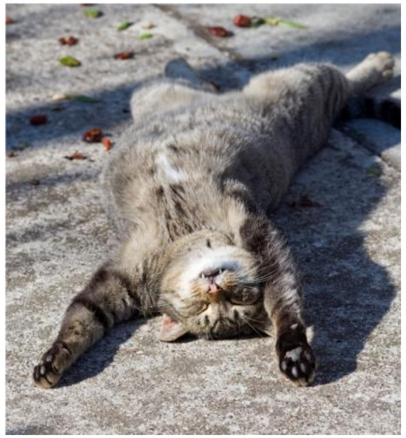
Dmitry Yashunin
IntelliVision

Recall from last time: Challenges of recognition

Illumination



Deformation



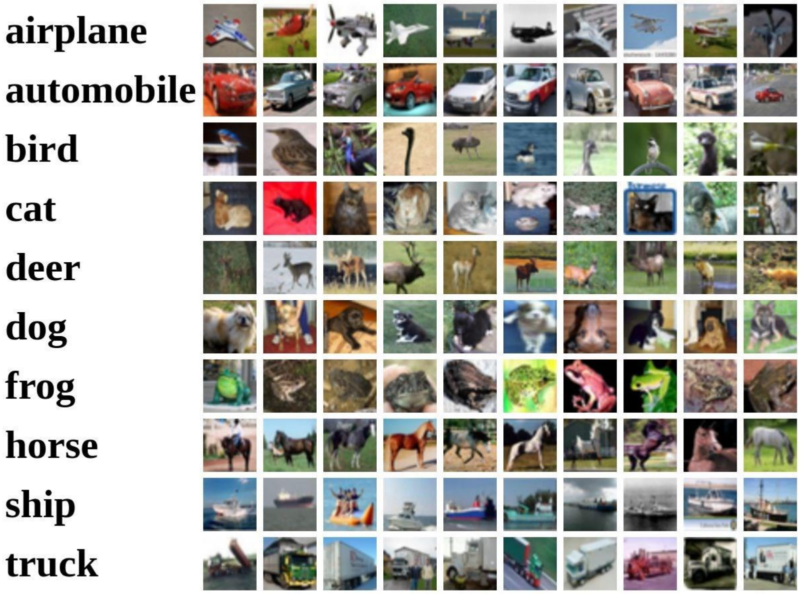
Occlusion

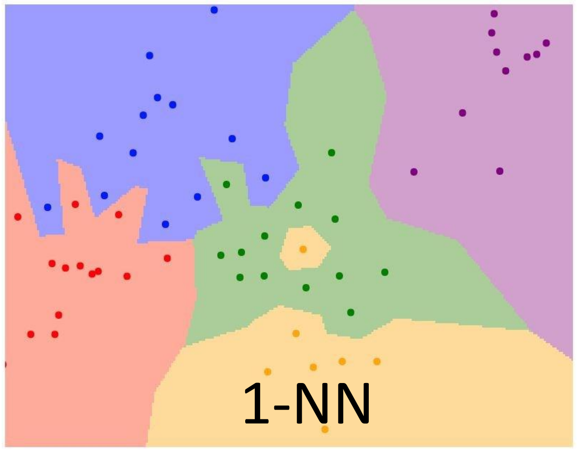
Background Clutter

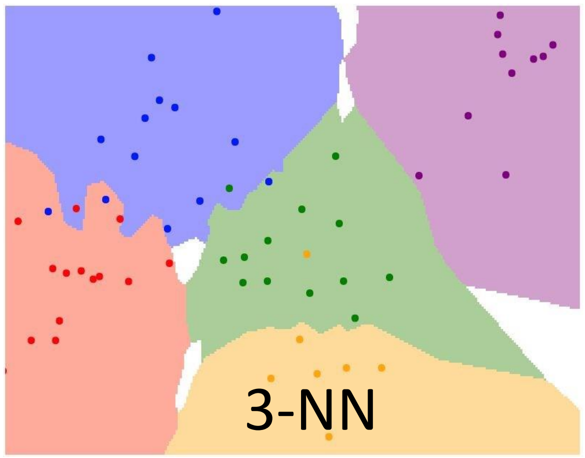
Intraclass variations

Recall from last time: data-driven approach, kNN

CIFAR10

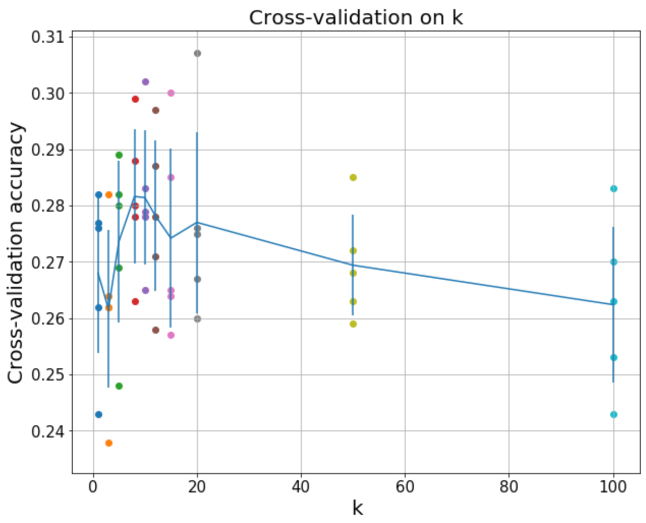


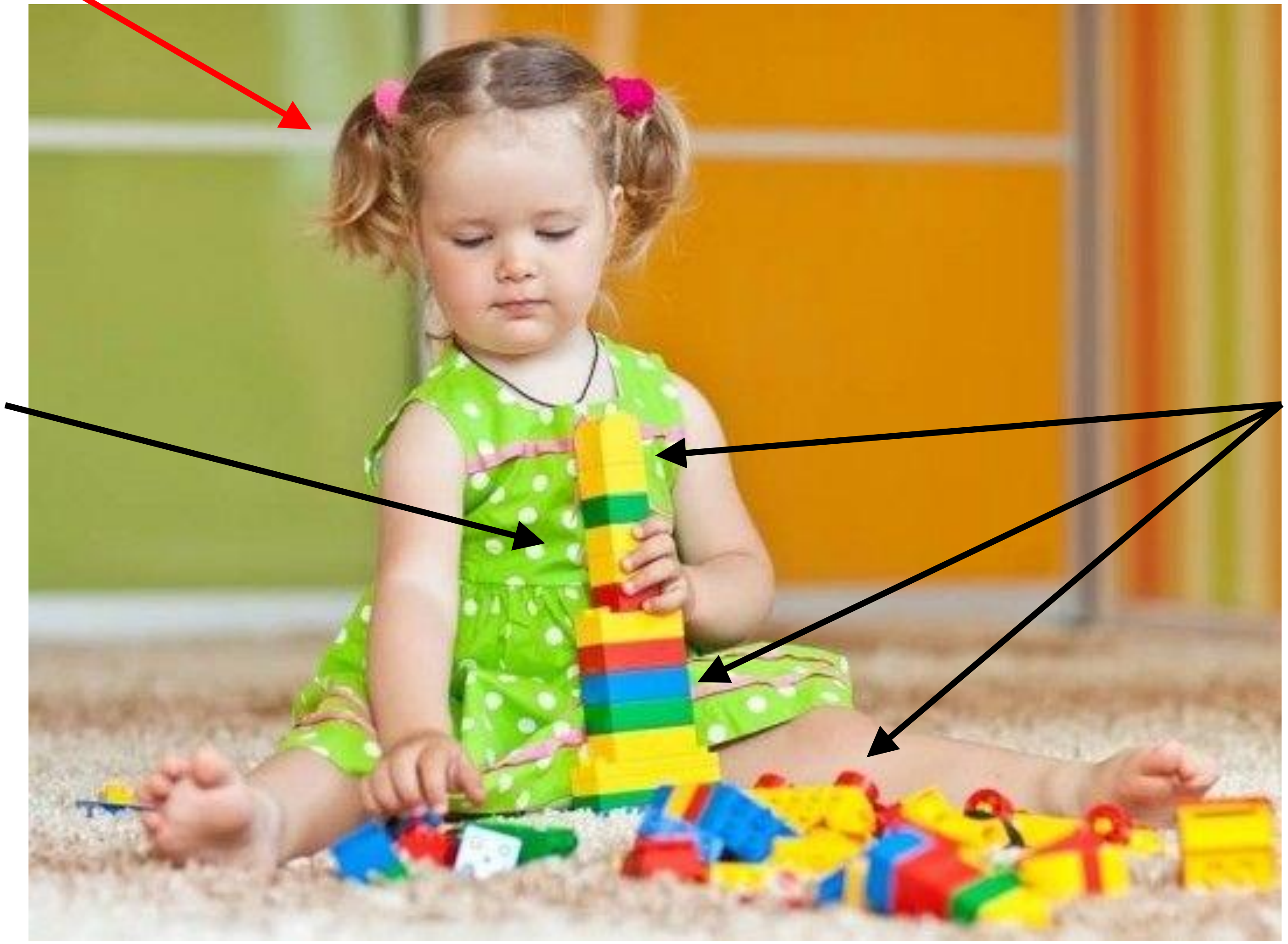




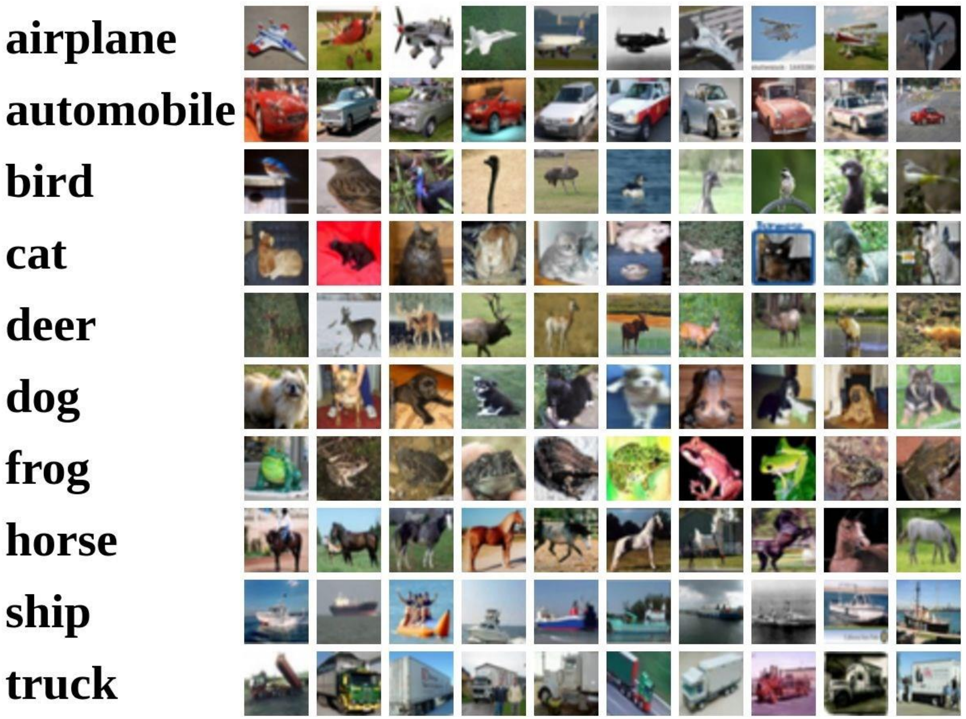
Cross-validation

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




Neural Network practitioner
Neural Network
Linear
classifiers

Recall: CIFAR10



10 classes
50,000 training images
10,000 testing images

Image size: **32x32x3**

width height channels (rgb)

Parametric Approach

Image

Array of **32x32x3** numbers
(3072 numbers total)

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

10 numbers giving
class scores

W

parameters
or weights

Parametric Approach: Linear Classifier

Image

Array of **32x32x3** numbers
(3072 numbers total)

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

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

10 numbers giving class scores

W

parameters
or weights

Parametric Approach: Linear Classifier

Image

Array of **32x32x3** numbers
(3072 numbers total)

3072x1

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

10x1 10x3072

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

10 numbers giving class scores

W

parameters
or weights

Parametric Approach: Linear Classifier

Image

Array of **32x32x3** numbers
(3072 numbers total)

3072x1

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

10x1 10x3072 10x1

$\rightarrow f(\mathbf{x},\mathbf{W}) \rightarrow$ **10** numbers giving class scores

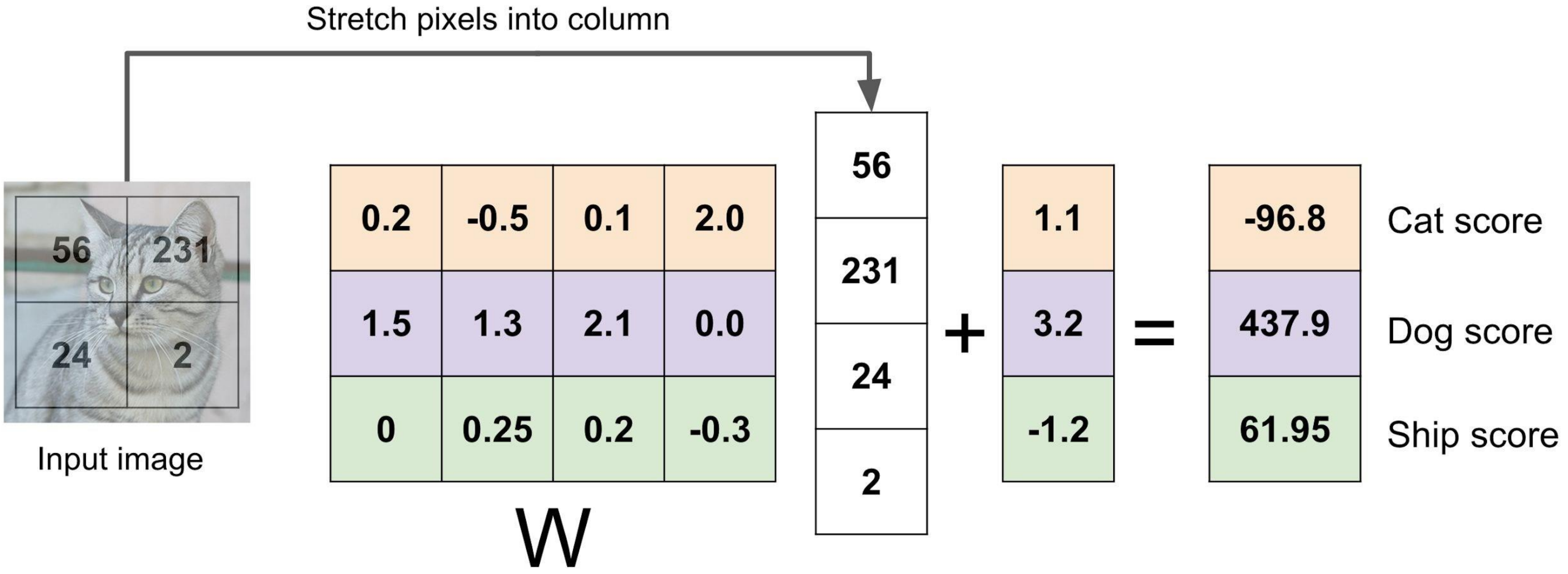
$\uparrow$

W

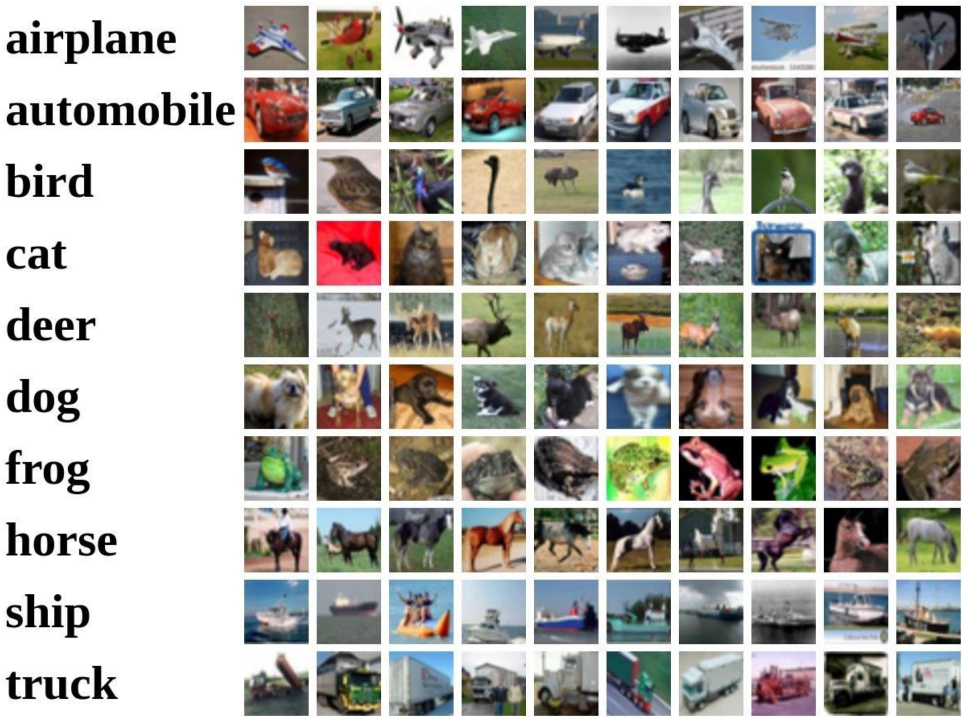
parameters
or weights

Toy example

Image with 4 pixels, and 3 classes (cat/dog/ship)



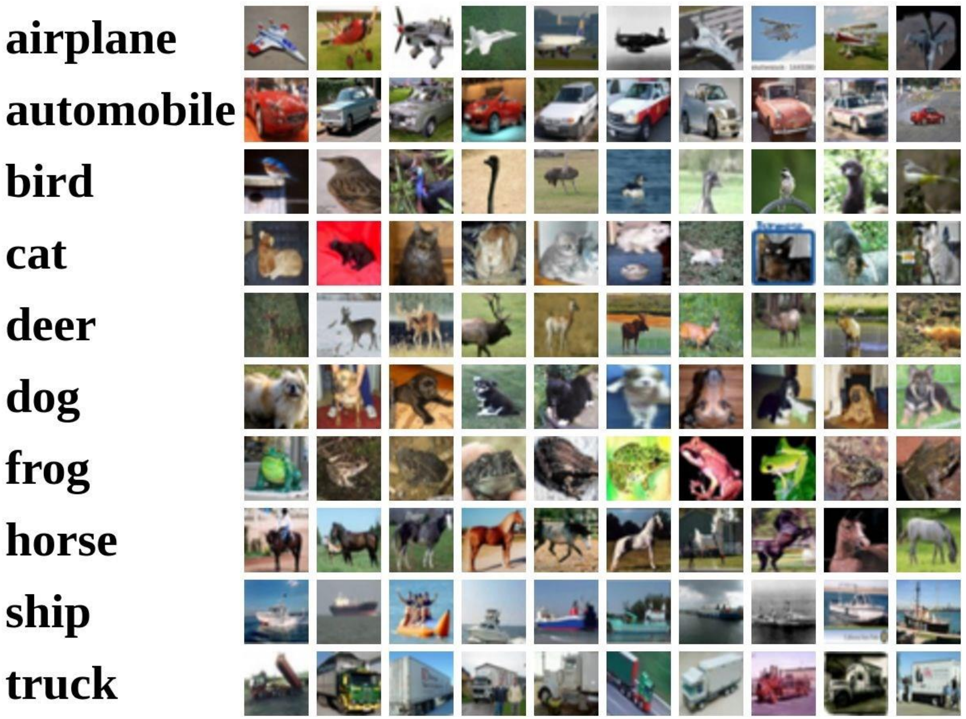
Interpreting a Linear Classifier



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

What is this thing doing?

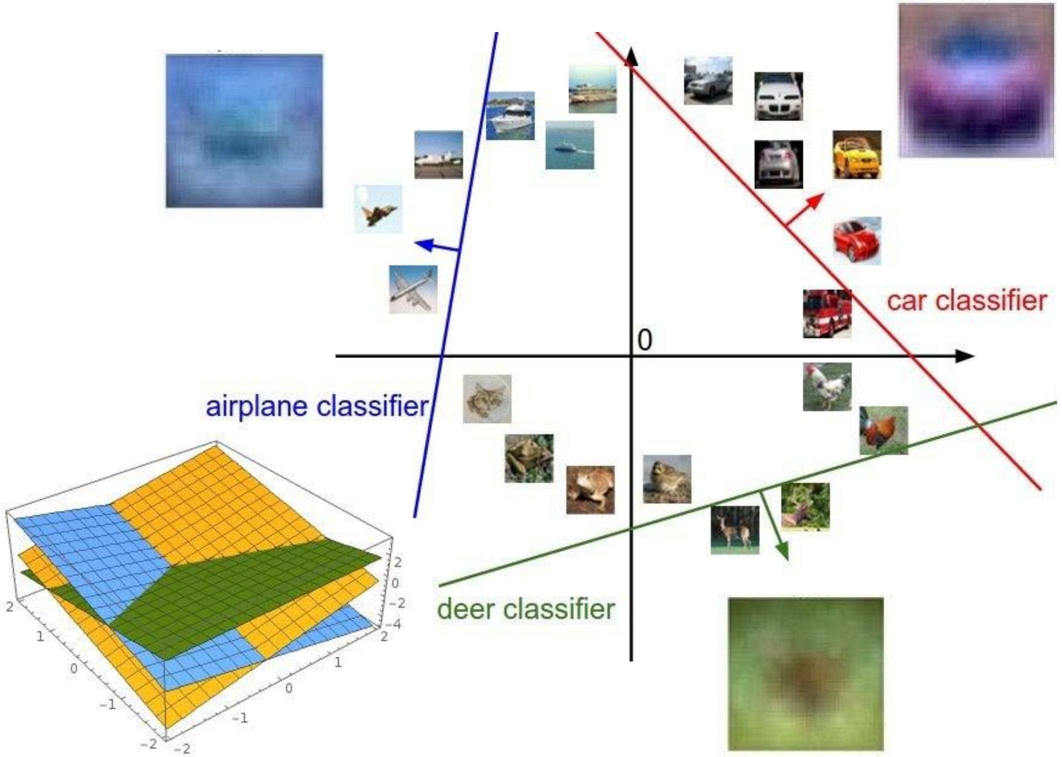
Interpreting a Linear Classifier



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

Example trained weights of a linear classifier trained on CIFAR-10:

Interpreting a Linear Classifier



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

Array of **32x32x3** numbers
(3072 numbers total)

How to define level of unhappiness with the scores?



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

Toy example

Consider: 3 training examples, 3 classes.
With the scores

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	-**3.1**

A **loss function** tells how good our classifier

x_i - image
y_i - label, element of a set {0, 1, …}

Loss over the dataset is a sum of loss over examples:

$$L = \frac{1}{N} \sum_i L_i(f(x_i, W), y_i)$$

Consider: 3 training examples, 3 classes.
With the scores

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

Multiclass SVM (hinge) loss:

x_i - image
y_i - label, element of a set {0, 1, …}

scores vector

$s = f(x_i, W) = [s_0, \ldots\, s_{y_i}, \ldots]$

s_{y_i} element corresponds
to ground truth label y_i

SVM loss

$$L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases}$$
$$= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Consider: 3 training examples, 3 classes.
With the scores



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9		

Multiclass SVM (hinge) loss:

x_i - image

y_i - label, element of a set {0, 1, …}

scores vector

$s = f(x_i, W) = [s_0, \ldots\ s_{y_i}, \ldots]$

s_{y_i} element corresponds to ground truth label y_i

SVM loss

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

$= \max(0, 5.1 - 3.2 + 1)$
$+ \max(0, -1.7 - 3.2 + 1)$
$= \max(0, 2.9) + \max(0, -3.9)$
$= 2.9 + 0$
$= 2.9$

Consider: 3 training examples, 3 classes.
With the scores

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9	0	

Multiclass SVM (hinge) loss:

x_i - image
y_i - label, element of a set {0, 1, …}

scores vector

$s = f(x_i, W) = [s_0, \dots s_{y_i}, \dots]$

s_{y_i} element corresponds
to ground truth label y_i

SVM loss

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

= max(0, 1.3 - 4.9 + 1)
+max(0, 2.0 - 4.9 + 1)
= max(0, -2.6) + max(0, -1.9)
= 0 + 0
= 0

Consider: 3 training examples, 3 classes.
With the scores



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9	0	12.9

Multiclass SVM (hinge) loss:

x_i - image
y_i - label, element of a set {0, 1, …}

scores vector

$s = f(x_i, W) = [s_0, \dots\, s_{y_i}, \dots]$

s_{y_i} element corresponds
to ground truth label y_i

SVM loss

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

= max(0, 2.2 - (-3.1) + 1)
+max(0, 2.5 - (-3.1) + 1)
= max(0, 6.3) + max(0, 6.6)
= 6.3 + 6.6
= 12.9

Consider: 3 training examples, 3 classes.
With the scores



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9	0	12.9

Multiclass SVM (hinge) loss:

x_i - image
y_i - label, element of a set {0, 1, …}

scores vector

$s = f(x_i, W) = [s_0, \dots s_{y_i}, \dots]$

s_{y_i} element corresponds
to ground truth label y_i

SVM loss

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

Loss over the dataset $L = \frac{1}{N} \sum_{i=1}^{N} L_i$

L = (2.9 + 0 + 12.9)/3
= **5.27**

Consider: 3 training examples, 3 classes.
With the scores



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9	0	12.9

Multiclass SVM (hinge) loss:

x_i - image
y_i - label, element of a set {0, 1, …}

scores vector

$s = f(x_i, W) = [s_0, \ldots s_{y_i}, \ldots]$

s_{y_i} element corresponds
to ground truth label y_i

SVM loss

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

Q: What happens to
loss if car scores
change a bit?

Consider: 3 training examples, 3 classes.
With the scores

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9	0	12.9

Multiclass SVM (hinge) loss:

x_i - image

y_i - label, element of a set {0, 1, …}

scores vector

$s = f(x_i, W) = [s_0, \dots s_{y_i}, \dots]$

s_{y_i} element corresponds to ground truth label y_i

SVM loss

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

Q2: what is the min/max possible loss?

Consider: 3 training examples, 3 classes.
With the scores



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	-**3.1**
Loss	2.9	0	12.9

Multiclass SVM (hinge) loss:

x_i - image
y_i - label, element of a set {0, 1, …}

scores vector

$s = f(x_i, W) = [s_0, \dots s_{y_i}, \dots]$

s_{y_i} element corresponds
to ground truth label y_i

SVM loss

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

Q3: At initialization W
is small so all s ≈ 0.
What is the loss?

Consider: 3 training examples, 3 classes.
With the scores



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9	0	12.9

Multiclass SVM (hinge) loss:

x_i - image

y_i - label, element of a set {0, 1, …}

scores vector

$s = f(x_i, W) = [s_0, \dots\ s_{y_i}, \dots]$

s_{y_i} element corresponds to ground truth label y_i

SVM loss

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

Q4: What if the sum was over all classes? (including $j = y_i$)

Consider: 3 training examples, 3 classes.
With the scores



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9	0	12.9

Multiclass SVM (hinge) loss:

x_i - image

y_i - label, element of a set {0, 1, …}

scores vector

$s = f(x_i, W) = [s_0, \dots\ s_{y_i}, \dots]$

s_{y_i} element corresponds to ground truth label y_i

SVM loss

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

Q5: What if we used mean instead of sum?

Consider: 3 training examples, 3 classes.
With the scores



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9	0	12.9

Multiclass SVM (hinge) loss:

x_i - image

y_i - label, element of a set {0, 1, …}

scores vector

$s = f(x_i, W) = [s_0, \dots s_{y_i}, \dots]$

s_{y_i} element corresponds
to ground truth label y_i

SVM loss

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

Q6: What if we used

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

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

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

E.g. Suppose that we found a W such that L = 0.
Is this W unique?

No! 2W is also has L = 0!

Consider: 3 training examples, 3 classes.
With the scores



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9	0	

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

Before:

= max(0, 1.3 - 4.9 + 1)
+max(0, 2.0 - 4.9 + 1)
= max(0, -2.6) + max(0, -1.9)
= 0 + 0
= 0

With 2W:

= max(0, 2.6 - 9.8 + 1)
+max(0, 4.0 - 9.8 + 1)
= max(0, -6.2) + max(0, -4.8)
= 0 + 0

Regularization

λ - regularization strength (hyperparameter)

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

L2 regularization $R(W) = \sum_k \sum_l W_{k,l}^2$

L1 regularization $R(W) = \sum_k \sum_l |W_{k,l}|$

Elastic net (L1 + L2) $R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$

Dropout (will see later)

Batch normalization (will see later)

L2 regularization: motivation

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

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

$$w_1 = [1, 0, 0, 0]$$
$$w_2 = [0.25, 0.25, 0.25, 0.25]$$

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

Softmax Classifier (Multinomial Logistic Regression)

scores = unnormalized log probabilities of the classes.

$s = f(x_i, W)$

$P(Y = y_i | X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}$ probability of x_i image has y_i label

cat	**3.2**
car	5.1
frog	-1.7

Softmax Classifier (Multinomial Logistic Regression)

scores = unnormalized log probabilities of the classes.

$s = f(x_i, W)$

$$P(Y = y_i | X = x_i) = \boxed{\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}}$$ probability of x_i image has y_i label

Softmax function

cat	**3.2**
car	5.1
frog	-1.7

Softmax Classifier (Multinomial Logistic Regression)

cat	**3.2**
car	5.1
frog	-1.7

scores = unnormalized log probabilities of the classes.

$s = f(x_i, W)$

$P(Y = y_i|X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}$ probability of x_i image has y_i label

Want to maximize the likelihood $\prod_i P(Y = y_i|X = x_i)$

or log likelihood $\sum_i \log P(Y = y_i|X = x_i)$

or to minimize the negative log likelihood of the correct class:

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

Softmax Classifier (Multinomial Logistic Regression)

scores = unnormalized log probabilities of the classes.

$s = f(x_i, W)$

$P(Y = y_i | X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}$ probability of x_i image has y_i label

cat	**3.2**
car	5.1
frog	-1.7

Want to minimize the negative log likelihood of the correct class:

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

or

$$\boxed{L_i = -\log \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}}$$

Softmax Classifier (Multinomial Logistic Regression)

$$L_i = -\log \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}$$

$y_i = 0$

cat	**3.2**
car	5.1
frog	-1.7

unnormalized log probabilities

Softmax Classifier (Multinomial Logistic Regression)

$$L_i = -\log \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}$$

unnormalized probabilities

cat	**3.2**		**24.5**
car	5.1	$\xrightarrow{\text{exp}}$	164.0
frog	-1.7		0.18

unnormalized log probabilities

Softmax Classifier (Multinomial Logistic Regression)

$$L_i = -\log \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}$$

unnormalized probabilities

		exp		normalize	
cat	**3.2**	$\rightarrow$	**24.5**	$\rightarrow$	**0.13**
car	5.1		164.0		0.87
frog	-1.7		0.18		0.00

unnormalized log probabilities

probabilities

Softmax Classifier (Multinomial Logistic Regression)

$$L_i = -\log \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}$$

unnormalized probabilities

y_i = 0

	unnormalized log probabilities	exp	unnormalized probabilities	normalize	probabilities
cat	**3.2**		**24.5**		**0.13**
car	5.1		164.0		0.87
frog	-1.7		0.18		0.00

L_i = -log(0.13) = 0.89

unnormalized log probabilities

probabilities

Softmax Classifier (Multinomial Logistic Regression)

$$L_i = -\log \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}$$

unnormalized probabilities

y_i = 0

	unnormalized log probabilities		unnormalized probabilities		probabilities
cat	**3.2**	exp →	**24.5**	normalize →	**0.13**
car	5.1		164.0		0.87
frog	-1.7		0.18		0.00

L_i = -log(0.13) = 0.89

unnormalized log probabilities

probabilities

Softmax Classifier (Multinomial Logistic Regression)

$$L_i = -\log \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}$$

Q: What is the possible min/max loss L_i?

unnormalized probabilities

$y_i = 0$

	unnormalized log probabilities	exp →	unnormalized probabilities	normalize →	probabilities
cat	**3.2**		**24.5**		**0.13**
car	5.1		164.0		0.87
frog	-1.7		0.18		0.00

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

unnormalized log probabilities

probabilities

Softmax Classifier (Multinomial Logistic Regression)

$$L_i = -\log \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}$$

Q2: Usually at initialization W is small so all s ≈ 0.
What is the loss?

unnormalized probabilities

y_i = 0

	unnormalized log probabilities	exp →	unnormalized probabilities	normalize →	probabilities
cat	**3.2**		**24.5**		**0.13**
car	5.1		164.0		0.87
frog	-1.7		0.18		0.001

L_i = -log(0.13) = 0.89

unnormalized log probabilities

probabilities

Cross-entropy loss (softmax loss)

Kullback–Leibler divergence – **nonsymmetrical** distance between two probability distribution P and Q

$$D_{KL}(P||Q) = \sum_x p(x)\log\frac{p(x)}{q(x)} = \underbrace{\sum_x p(x)\log p(x)}_{\boldsymbol{H(P)}\text{ entropy of }P} - \underbrace{\sum_x p(x)\log q(x)}_{\boldsymbol{H(P,\,Q)}\text{ cross-entropy of }P\text{ and }Q} = H(P) + H(P,Q)$$

Image classification:

$P(Y = y_i|X = x_i)$ – "true" distribution of class probabilities

$Q(Y = \widehat{y}_i|X = x_i)$ – estimated class probabilities, softmax output of classifier

Cross-entropy loss (softmax loss)

Want to minimize Kullback–Leibler divergence

$$D_{KL}(P||Q) = \underbrace{\sum_i P(y_i|x_i)\log P(y_i|x_i)}_{\text{constant}} - \underbrace{\sum_i P(y_i|x_i)\log Q(\hat{y}_i|x_i)}_{\textbf{\textit{H(P, Q)}}\text{ cross-entropy of } P \text{ and } Q}$$

minimizing Kullback–Leibler divergence is equivalent to minimizing cross-entropy of *P* and *Q*

Cross-entropy loss (softmax loss)

Want to minimize cross entropy loss for an image x_i

$$\boxed{L_i = -P(y_i|x_i)\log Q(\hat{y}_i|x_i)}$$

If image labeled with single class y_i then

one-hot label encoding

$$P(y_i|x_i) = [0, \ldots 1, \ldots 0]$$

single 1 at y_i-th position

softmax of classifier scores

$$Q(\hat{y}_i|x_i) = \frac{e^{s_i}}{\sum_j e^{s_j}}$$

$$\boxed{L_i = - -\log \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}}$$

Cross-entropy loss (softmax loss)

Want to minimize cross entropy loss for an image x_i

$$\boxed{L_i = -P(y_i|x_i)\log Q(\hat{y}_i|x_i)}$$

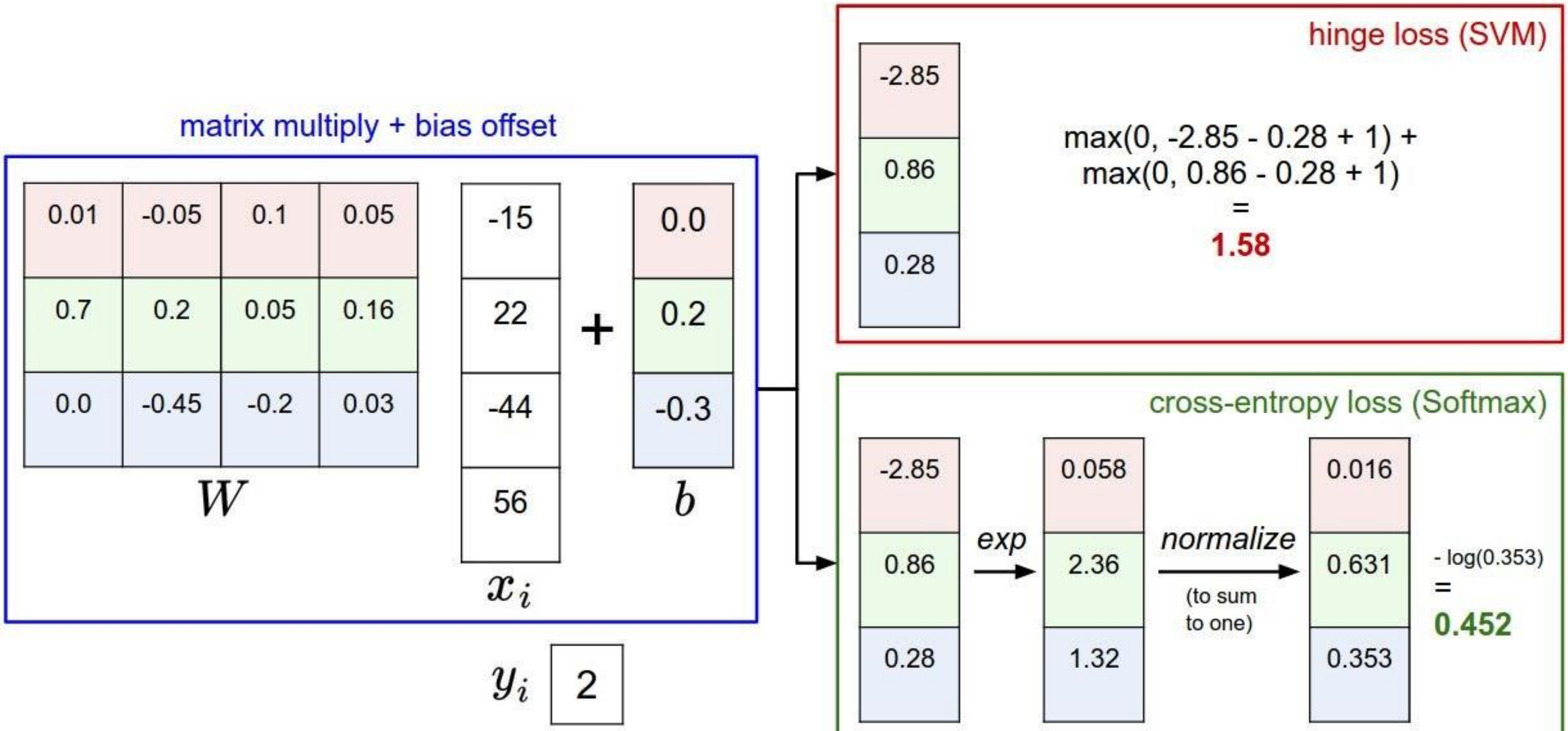
$P(y_i|x_i)$ - ground truth probabilities

$Q(\hat{y}_i|x_i) = \frac{e^{s_i}}{\sum_j e^{s_j}}$ - softmax of classifier scores

			$Q(\hat{y}_i\|x_i)$				$P(y_i\|x_i)$
cat	3.2	softmax →	0.13	-log →	0.89	multiply and sum	1.0
car	5.1		0.87		0.06		0
frog	-1.7		0.00		3.0		0

$L_i = 0.89$

one-hot label encoding


matrix multiply + bias offset
0.01
-0.05
0.1
0.05
0.7
0.2
0.05
0.16
0.0
-0.45
-0.2
0.03
W
-15
22
-44
56
x_i
+
0.0
0.2
-0.3
b
y_i
2
hinge loss (SVM)
-2.85
0.86
0.28
max(0, -2.85 - 0.28 + 1) +
max(0, 0.86 - 0.28 + 1)
=
1.58
cross-entropy loss (Softmax)
-2.85
0.86
0.28
exp
0.058
2.36
1.32
normalize
(to sum
to one)
0.016
0.631
0.353
- log(0.353)
=
0.452

Softmax vs SVM

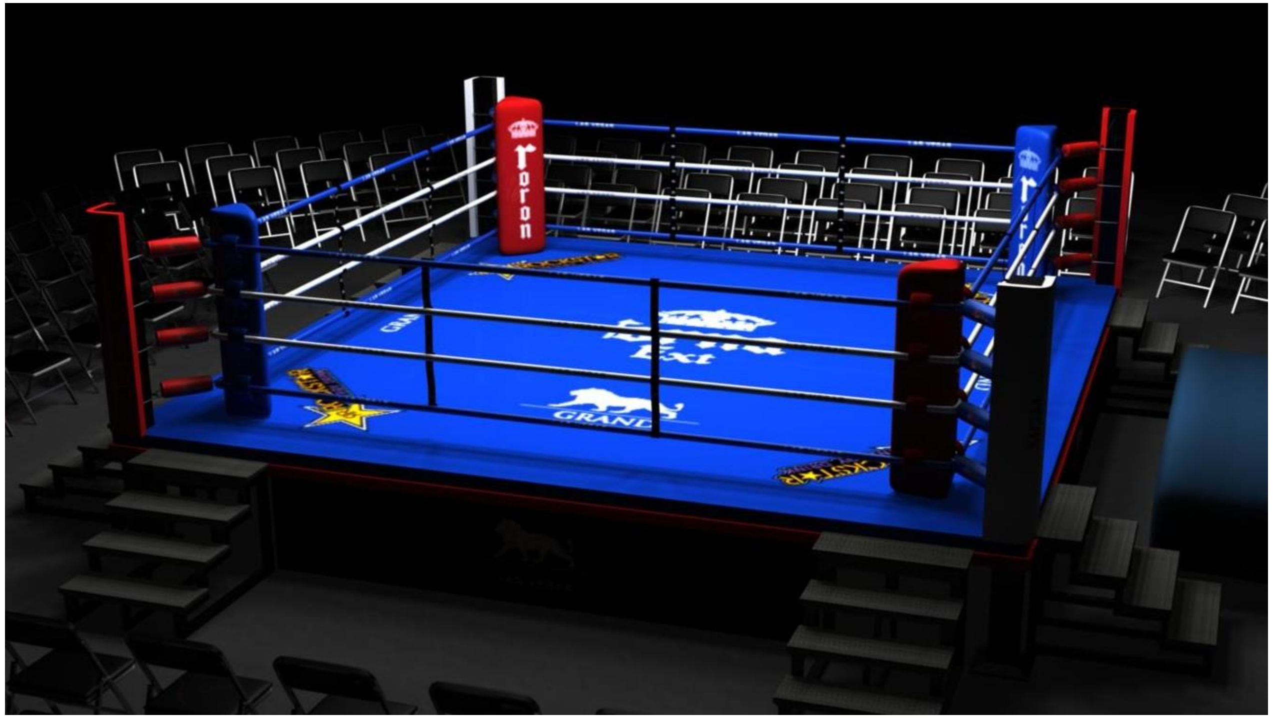
Softmax

$$L_i = -\log \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}$$

vs

SVM

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



Softmax vs SVM

Softmax

$$L_i = -\log \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}$$

vs

SVM

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

assume scores:

[10, -2, 3]

[10, 9, 9]

[10, -100, -100]

and $y_i = 0$

Q: Suppose I take a datapoint and I jiggle a bit (changing its score slightly). What happens to the loss in both cases?

Recap

- We have some dataset of (x_i , y_i)
- We have a **score function:** $s = f(x_i, W)$
- We have a **loss function**:

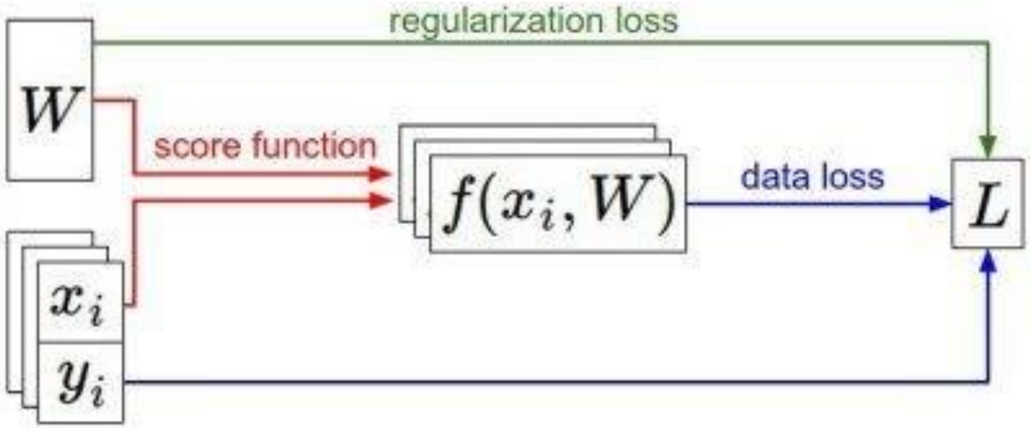
$$L_i = -\log \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}$$ Softmax

SVM

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

Full loss

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



Recap

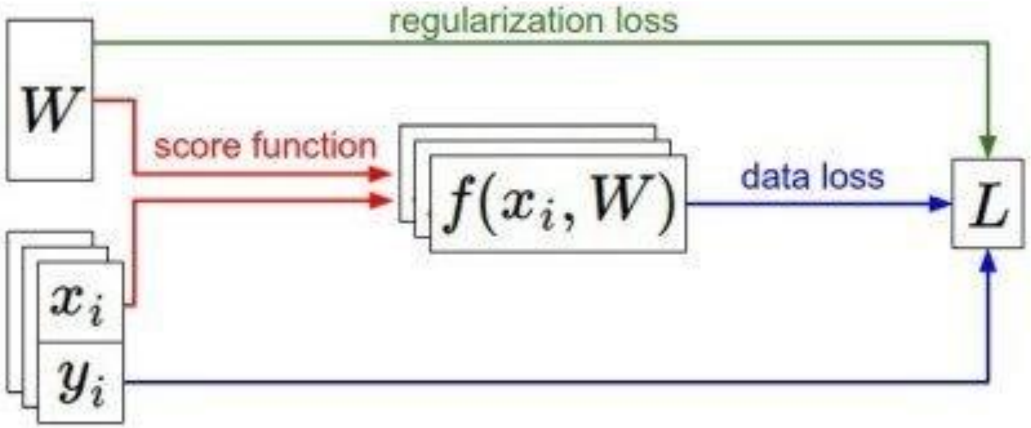
How do we find the best W?

- We have some dataset of (x_i, y_i)
- We have a **score function:** $s = f(x_i, W)$
- We have a **loss function**:

$$L_i = -\log \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}$$ Softmax

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

Full loss
$$L = \frac{1}{N}\sum_{i=1}^{N} L_i + R(W)$$



Next time

Optimization

Introduction to neural networks

Backpropagation