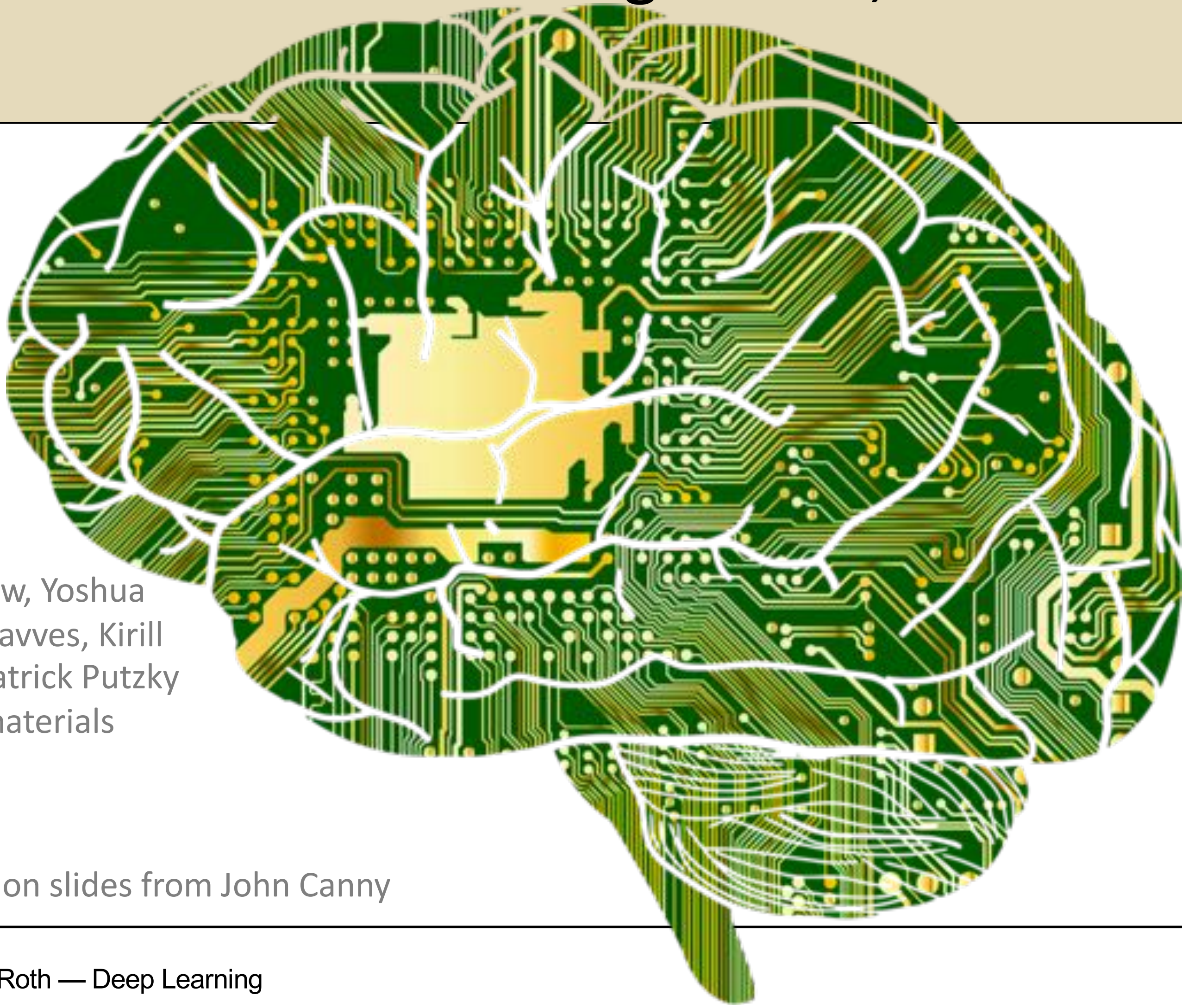
Deep Learning

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Architectures and Methods: Linear Regression, SVMs, Softmax



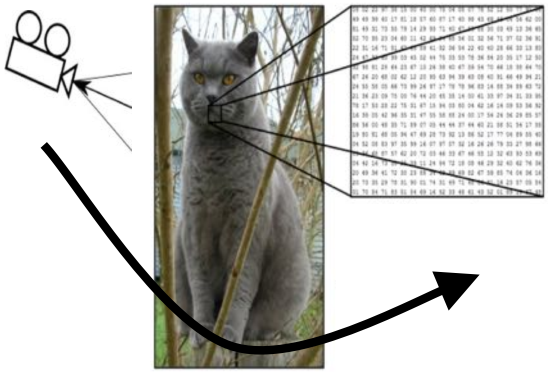
Thanks to John Canny, Ian Goodfellow, Yoshua Bengio, Aaron Courville, Efstratios Gavves, Kirill Gavrilyuk, Berkay Kicanaoglu, and Patrick Putzky and many others for making their materials publically available.

The present slides are mainly based on slides from John Canny

Recall from last time…

Challenges in Visual Recognition

Camera pose



Illumination

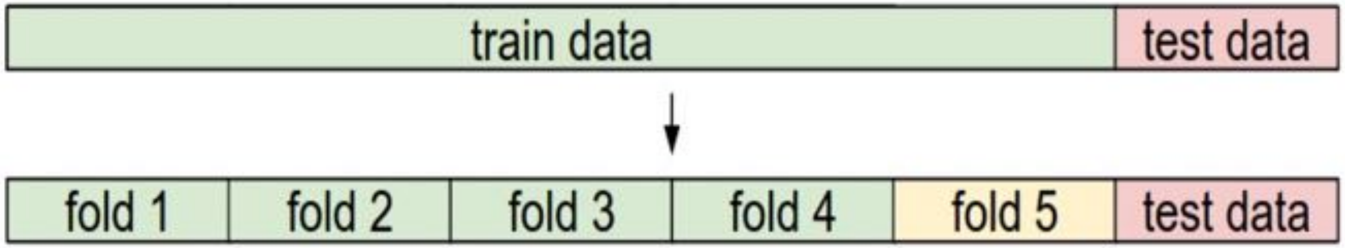
Deformation

Occlusion

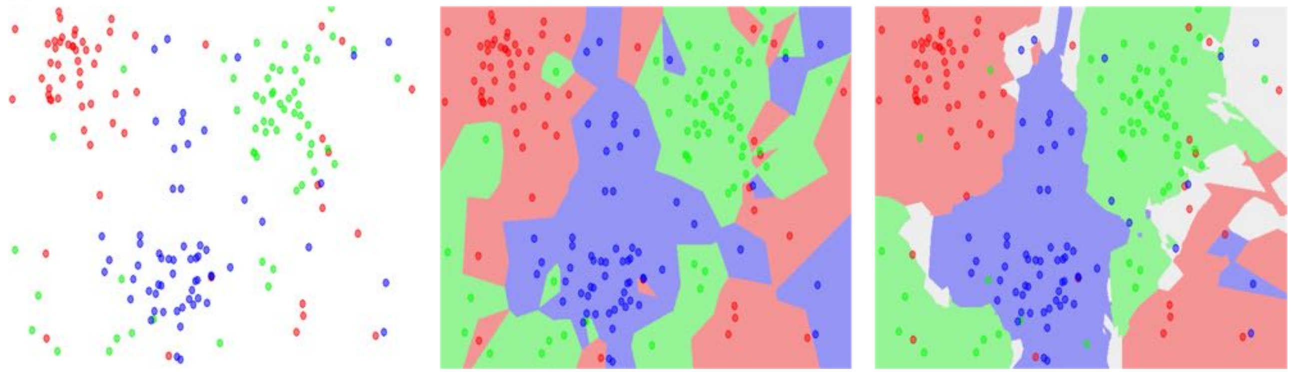
Background clutter

Intraclass variation

Recall from last time… data-driven approach, kNN



the data NN classifier 5-NN classifier



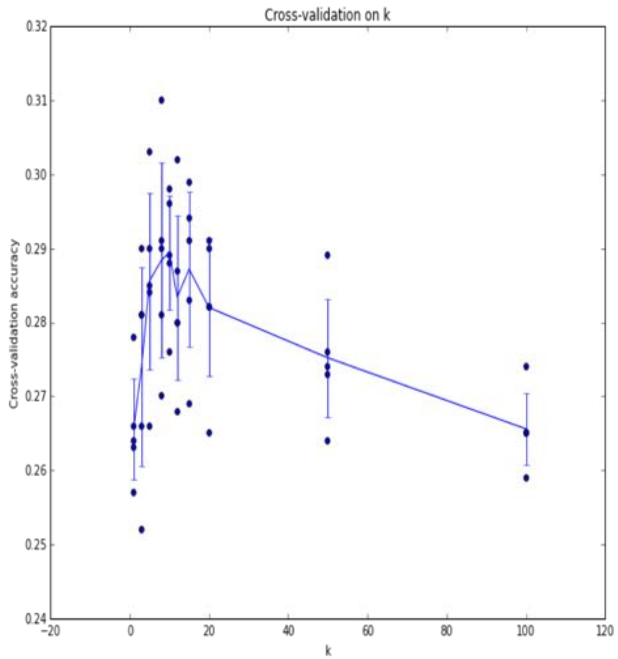
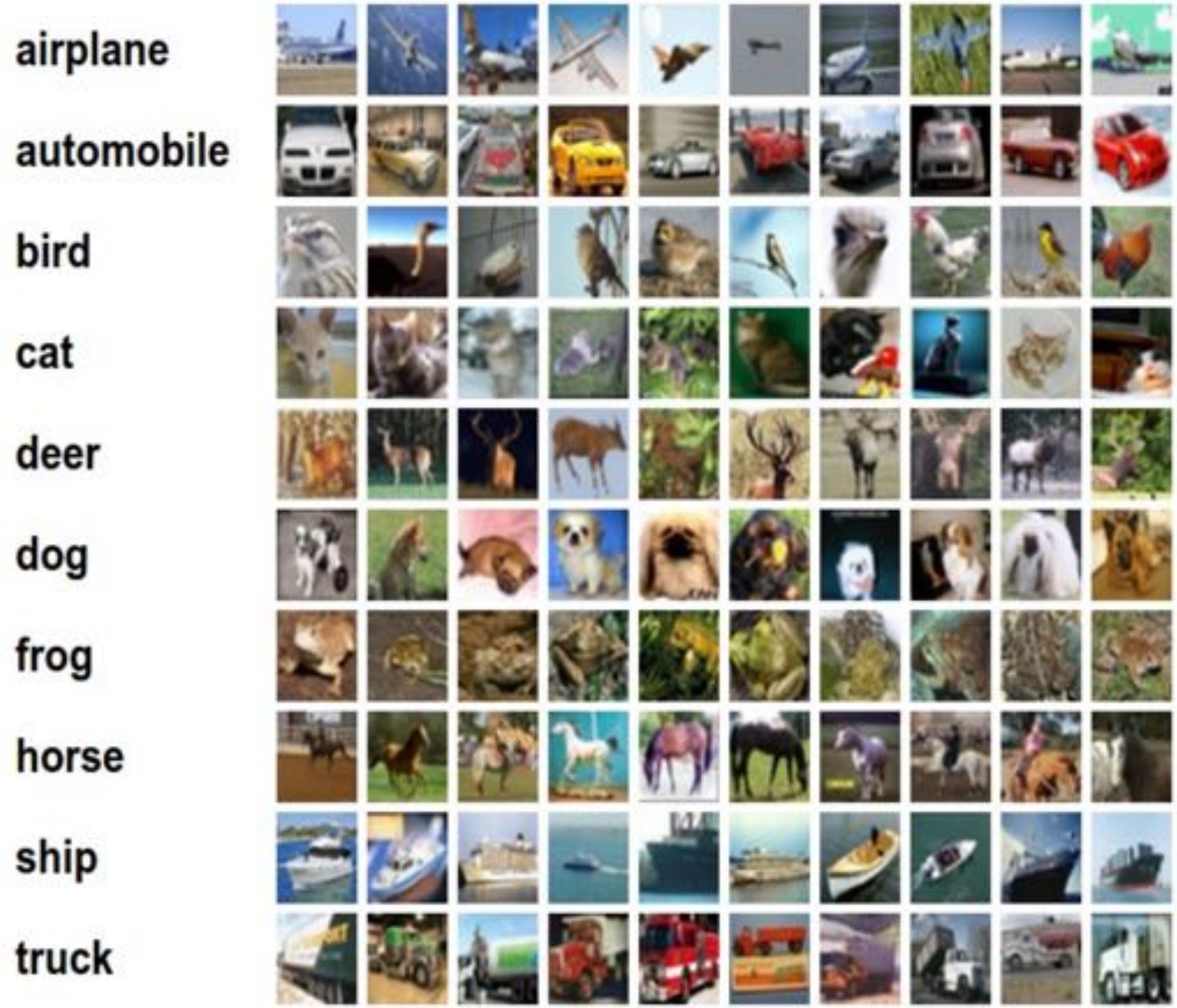


Image Classification



CIFAR-10
10 labels
50,000 training images
  each image is **32x32x3**
10,000 test images.

Parametric approach

image parameters

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

[32x32x3]
array of numbers 0...1
(3072 numbers total)

10 numbers,
indicating class
scores

Parametric approach: **Linear classifier**

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

[32x32x3]
array of numbers 0...1

10 numbers,
indicating class
scores

Parametric approach: **Linear classifier**

TECHNISCHE UNIVERSITÄT DARMSTADT

$$\underbrace{\boxed{f(x,W)}}_{10\text{x}1} = \underbrace{\boxed{W}}_{10\text{x}3072}\overbrace{\boxed{x}}^{3072\text{x}1}$$

[32x32x3]
array of numbers 0...1

10 numbers, indicating class scores

parameters, or “weights”

Parametric approach: **Linear classifier**

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

10x1   10x3072   3072x1   10x1

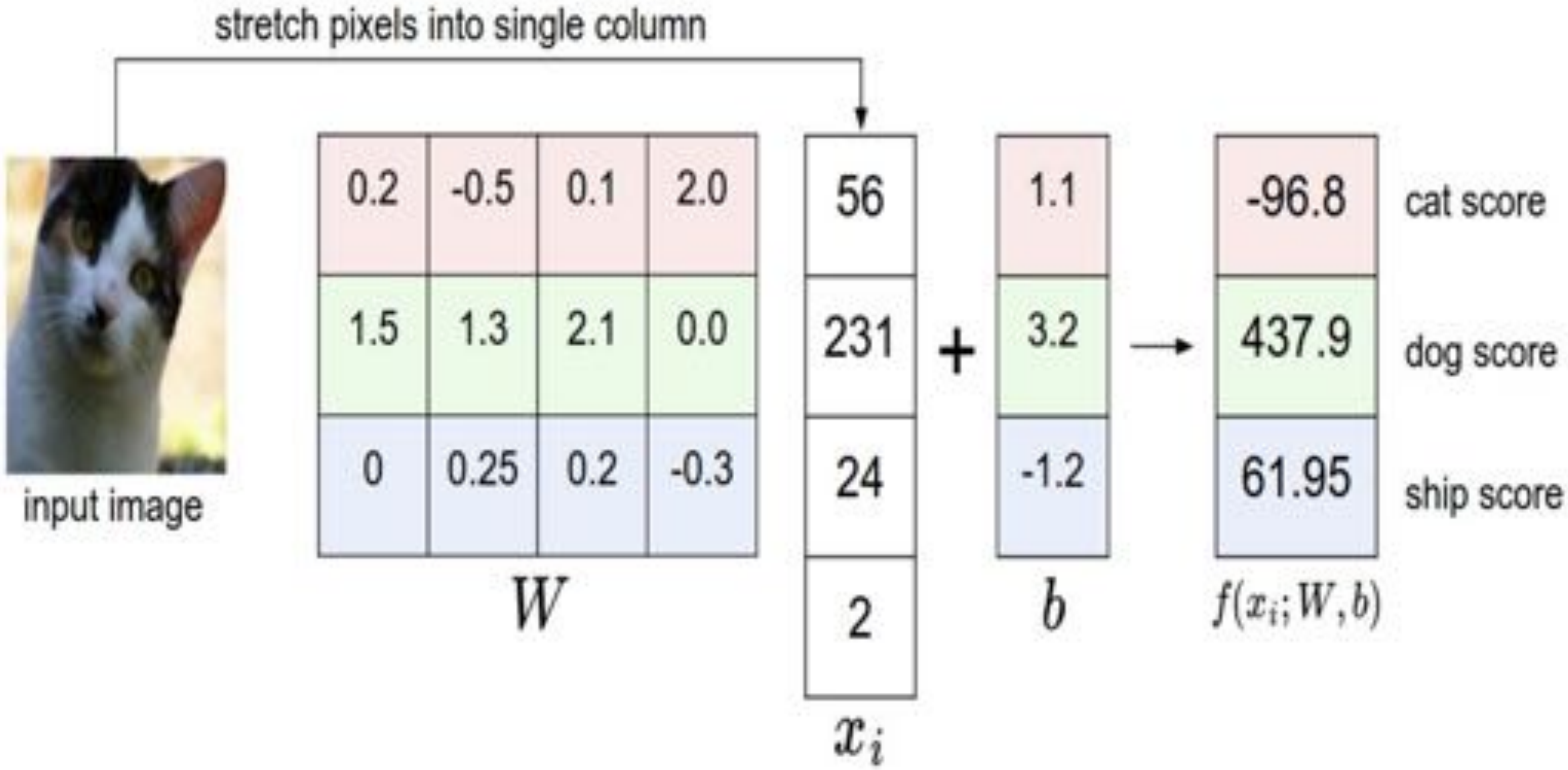
bias

[32x32x3]
array of numbers 0...1

10 numbers, indicating class scores

parameters, or "weights"

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



Interpreting a Linear Classifier

$$f(x_i, W, b) = Wx_i + b$$

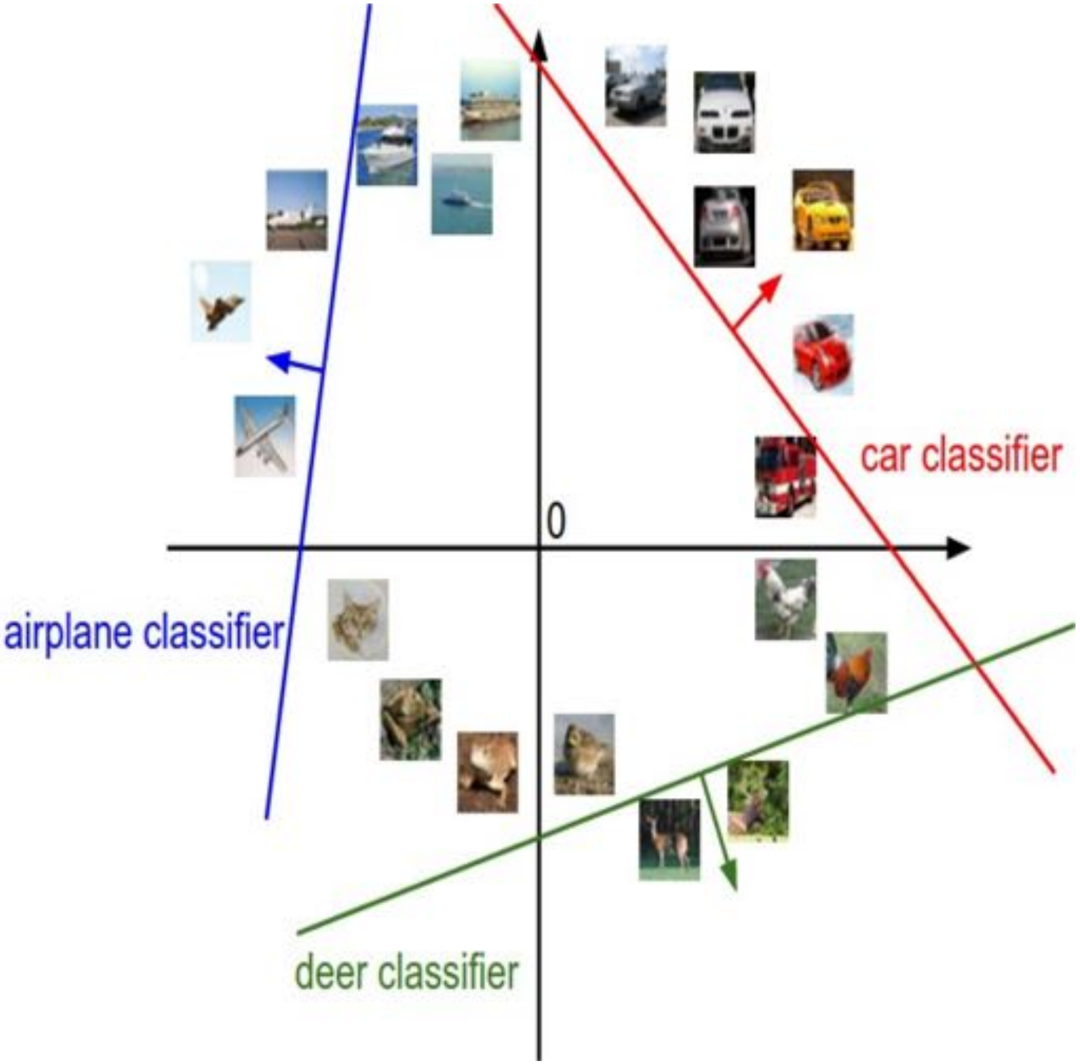
Q: what does the linear classifier do, in English?

Interpreting a Linear Classifier

$$f(x_i, W, b) = Wx_i + b$$

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

Interpreting a Linear Classifier



$$f(x_i, W, b) = Wx_i + b$$

[32x32x3]
array of numbers 0...1
(3072 numbers total)

Interpreting a Linear Classifier

$$f(x_i, W, b) = Wx_i + b$$

Q2: what would be a very hard set of classes for a linear classifier to distinguish?

Bias and Variance

How do bias and variance for the linear classifier compare to kNN:

Bias: ?

Variance: ?

Bias and Variance

How do bias and variance for the linear classifier compare to kNN:

Bias: ? Higher for linear classifier. Will have errors on images which aren't linearly separable from other classes.

Variance: ?

Bias and Variance

How do bias and variance for the linear classifier compare to kNN:

Bias: ? Higher for linear classifier. Will have errors on images which aren't linearly separable from other classes.

Variance: ? Lower: linear output is a weighted sum of all the input pixels.

Going forward: Loss functions/optimization

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

TODO:

1. Define a **loss function** that quantifies our unhappiness with the scores across the training data.

1. Come up with a way of efficiently finding the parameters that minimize the loss function. **(optimization)**

Loss functions

Suppose: 3 training examples, 3 classes.
For some W the scores $f(x, W) = Wx$ are:

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

Loss functions

Suppose: 3 training examples, 3 classes.
For some W the scores $f(x, W) = Wx$ are:

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

Multiclass SVM loss:

Given an example (x_i, y_i) where x_i is the image and where y_i is the (integer) label, and using the shorthand for the scores vector:

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

the SVM loss has the form:

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

Loss functions

Suppose: 3 training examples, 3 classes.
For some W the scores $f(x, W) = Wx$ are:

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

Multiclass SVM loss:

Given an example (x_i, y_i) where x_i is the image and where y_i is the (integer) label, and using the shorthand for the scores vector:

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

the SVM loss has the form:

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

= 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

Loss functions

Suppose: 3 training examples, 3 classes.
For some W the scores $f(x, W) = Wx$ are:

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

Multiclass SVM loss:

Given an example (x_i, y_i) where x_i is the image and where y_i is the (integer) label, and using the shorthand for the scores vector:

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

the SVM loss has the form:

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

= 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

Loss functions

Suppose: 3 training examples, 3 classes.
For some W the scores $f(x, W) = Wx$ are:



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	10.9

Multiclass SVM loss:

Given an example (x_i, y_i) where x_i is the image and where y_i is the (integer) label, and using the shorthand for the scores vector:

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

the SVM loss has the form:

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

= max(0, 2.2 - (-3.1) + 1)
+max(0, 2.5 - (-3.1) + 1)
= max(0, 5.3) + max(0, 5.6)
= 5.3 + 5.6
= 10.9

Loss functions

Suppose: 3 training examples, 3 classes.
For some W the scores $f(x,W) = Wx$ are:

			
cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	10.9

Multiclass SVM loss:

Given an example (x_i, y_i) where x_i is the image and where y_i is the (integer) label, and using the shorthand for the scores vector:

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

the SVM loss has the form:

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

and the full training loss is the mean over all the examples:

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

L = (2.9 + 0 + 10.9)/3 = **4.6**

Loss functions

Suppose: 3 training examples, 3 classes.
For some W the scores $f(x, W) = Wx$ are:



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	10.9

Multiclass SVM loss:

Given an example (x_i, y_i) where x_i is the image and where y_i is the (integer) label, and using the shorthand for the scores vector:

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

the SVM loss has the form:

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

Q: what if the sum was instead over all classes?
(including j = y_i)

Loss functions

Suppose: 3 training examples, 3 classes.
For some W the scores $f(x, W) = Wx$ are:

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	10.9

Multiclass SVM loss:

Given an example (x_i, y_i) where x_i is the image and where y_i is the (integer) label, and using the shorthand for the scores vector:

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

the SVM loss has the form:

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

Q2: what if we used a mean instead of a sum here?

Loss functions

Suppose: 3 training examples, 3 classes.
For some W the scores $f(x, W) = Wx$ are:

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	10.9

Multiclass SVM loss:

Given an example (x_i, y_i) where x_i is the image and where y_i is the (integer) label, and using the shorthand for the scores vector:

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

the SVM loss has the form:

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

Q3: what if we used a squared loss:

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

Loss functions

Suppose: 3 training examples, 3 classes.
For some W the scores $f(x, W) = Wx$ are:



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	10.9

Multiclass SVM loss:

Given an example (x_i, y_i) where x_i is the image and where y_i is the (integer) label, and using the shorthand for the scores vector:

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

the SVM loss has the form:

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

Q4: what is the min/max possible loss?

Loss functions

Suppose: 3 training examples, 3 classes.
For some W the scores $f(x, W) = Wx$ are:

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	10.9

Multiclass SVM loss:

Given an example (x_i, y_i) where x_i is the image and where y_i is the (integer) label, and using the shorthand for the scores vector:

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

the SVM loss has the form:

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

Q5: usually at initialization W are small numbers, so all $s \approx 0$
What is the loss?

Example numpy code:

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

```python
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
```

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

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

There is a bug with the loss:

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

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

There is a bug with the loss:

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

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

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

Loss functions

Suppose: 3 training examples, 3 classes.
For some W the scores $f(x, W) = Wx$ are:



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	10.9

Multiclass SVM loss:

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

Original Car Image Loss:
= 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

W twice as large:
= max(0, 2.6 - 9.8 + 1)
+max(0, 4.0 - 9.8 + 1)
= max(0, -6.2) + max(0, -4.8)
= 0 + 0
= 0

Weight Regularization

λ= regularization strength (hyperparameter)

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

In common use:

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

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

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

Dropout (will see later)

Max norm regularization (might see later)

L2 regularization: motivation

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

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

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

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

What will L2 regularization do?

What will L1 regularization do?

Softmax Classifier (Multinomial Logistic Regression)

cat	**3.2**
car	5.1
frog	-1.7

Softmax Classifier (Multinomial Logistic Regression)

Scores = unnormalized log prob. of the classes

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

cat	**3.2**
car	5.1
frog	-1.7

Softmax Classifier (Multinomial Logistic Regression)

Scores = unnormalized log prob. of the classes

$\boxed{P(Y = k | X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}}$ where $\boxed{s = f(x, W)}$

cat	**3.2**
car	5.1
frog	-1.7

Softmax Classifier (Multinomial Logistic Regression)

Scores = unnormalized log prob. of the classes

$$P(Y = k | X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \quad \text{where} \quad s = f(x, W)$$

Softmax function

cat	**3.2**
car	5.1
frog	-1.7

Softmax Classifier (Multinomial Logistic Regression)

Scores = unnormalized log prob. of the classes

$$\boxed{P(Y = k | X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}} \quad \text{where} \quad \boxed{s = f(x, W)}$$

Want to maximize the log likelihood, or (for a loss function) to minimize the negative log likelihood of the correct class:

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

cat	**3.2**
car	5.1
frog	-1.7

Softmax Classifier (Multinomial Logistic Regression)

Scores = unnormalized log prob. of the classes

$$\boxed{P(Y = k|X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}}$$ where $$\boxed{s = f(x, W)}$$

Want to maximize the log likelihood, or (for a loss function) to minimize the negative log likelihood of the correct class:

cat	**3.2**
car	5.1
frog	-1.7

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

In summary:

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

Softmax Classifier (Multinomial Logistic Regression)

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

cat	**3.2**
car	5.1
frog	-1.7

unnormalized log probabilities

Softmax Classifier (Multinomial Logistic Regression)

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

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)

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

unnormalized probabilities

	unnormalized log probabilities	$\xrightarrow{\text{exp}}$	unnormalized probabilities	$\xrightarrow{\text{normalize}}$	probabilities
cat	**3.2**		**24.5**		**0.13**
car	5.1		164.0		0.87
frog	-1.7		0.18		0.00

unnormalized log probabilities

probabilities

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

Softmax Classifier (Multinomial Logistic Regression)

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

Q: What is the min/max possible L_i ?

unnormalized probabilities

	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

unnormalized log probabilities

probabilities

→ $L_i = -\log(0.13)$

= **0.89**

Softmax Classifier (Multinomial Logistic Regression)

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

Q: usually at initialization W are small numbers, so all s ≈ 0. What is the loss?

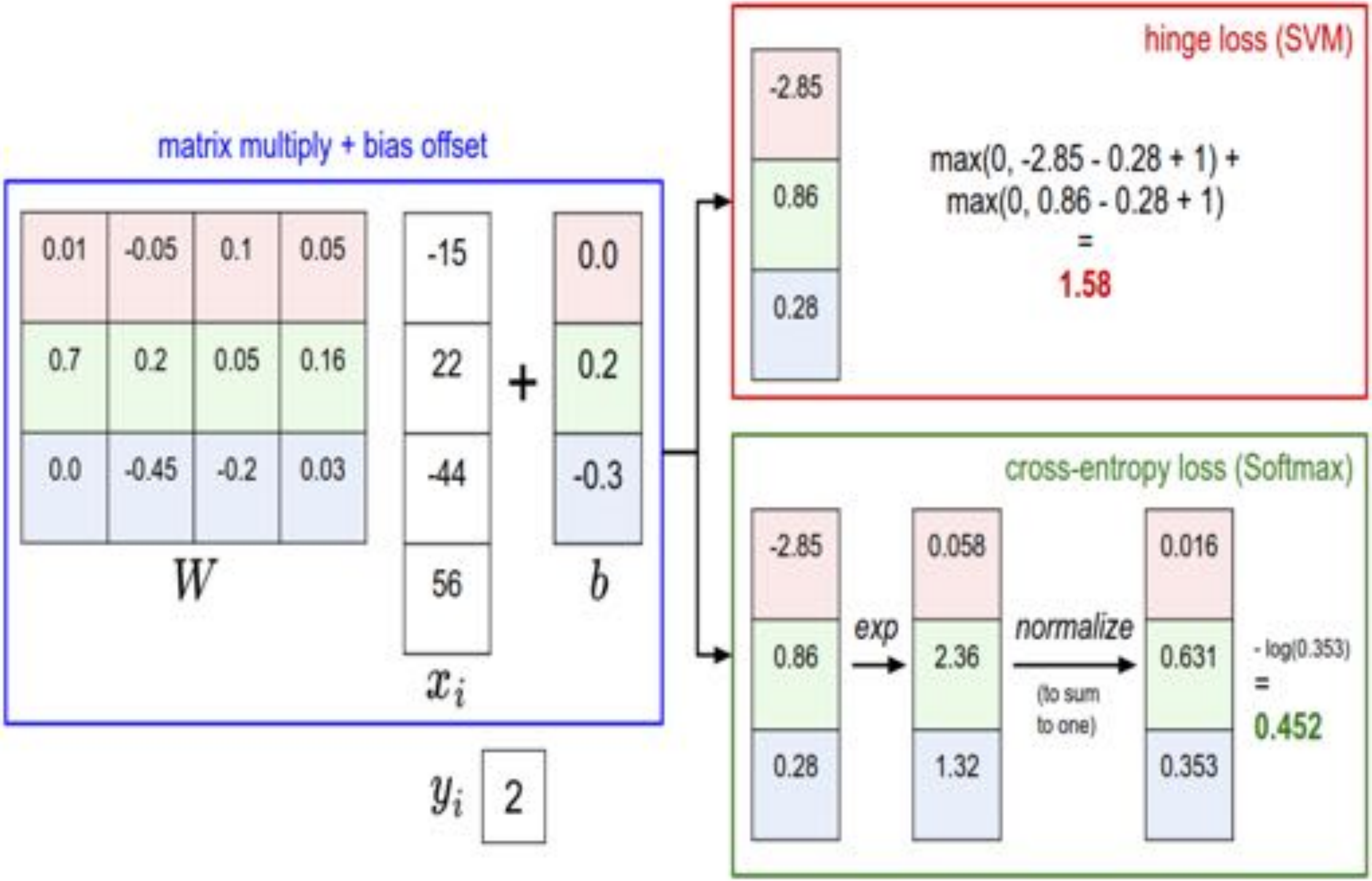
unnormalized probabilities

	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



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

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

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

Softmax vs. SVM

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

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

assume scores:
[10, -2, 3]
[10, 9, 9]
[10, -100, -100]
and

$$\boxed{y_i = 0}$$

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

Interactive Web Demo time....

http://vision.stanford.edu/teaching/cs231n/linear-classify-demo/

What have you learnt?

- **Linear classifiers:** The simplest ***parametric*** classifier.
- You must define a loss function in order to optimize the weights of a linear classifier. We first described **SVM loss**.
- Certain loss functions (esp. SVM loss) underconstrain the model parameters. **Regularization** can fix this.
- To produce a continuous prediction (probability of class membership) we described the **softmax loss**.
- Finally we explored these methods with an **interactive visualization**.