Discrete Mathematics
(for Computer Science)

Zhaoguo Wang

Slides are adapted from Stanford CS231.

Part III: Machine Learning Part

Question 1. What does ML look like?

Image classification with linear model

Question 2. How does ML benefit System?

Learned Index + XIndex

I am not an expert on ML; I am actually a systems researcher.

All my claims are from a user's perspective.

Part III: Machine Learning Part

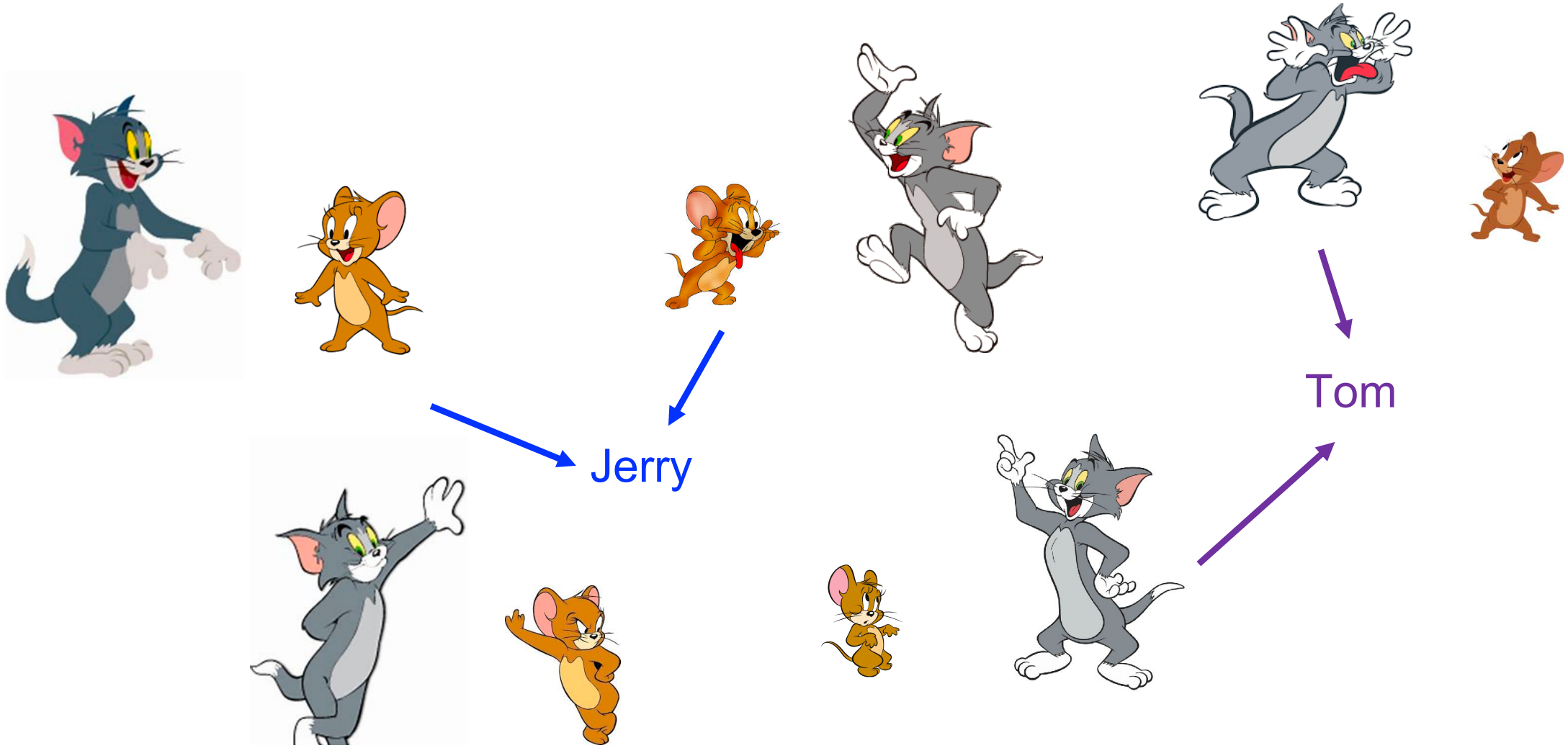
Question 1. What does ML look like?

Image classification with linear model

Question 2. How does ML benefit System?

Learned Index + XIndex

Image Classification

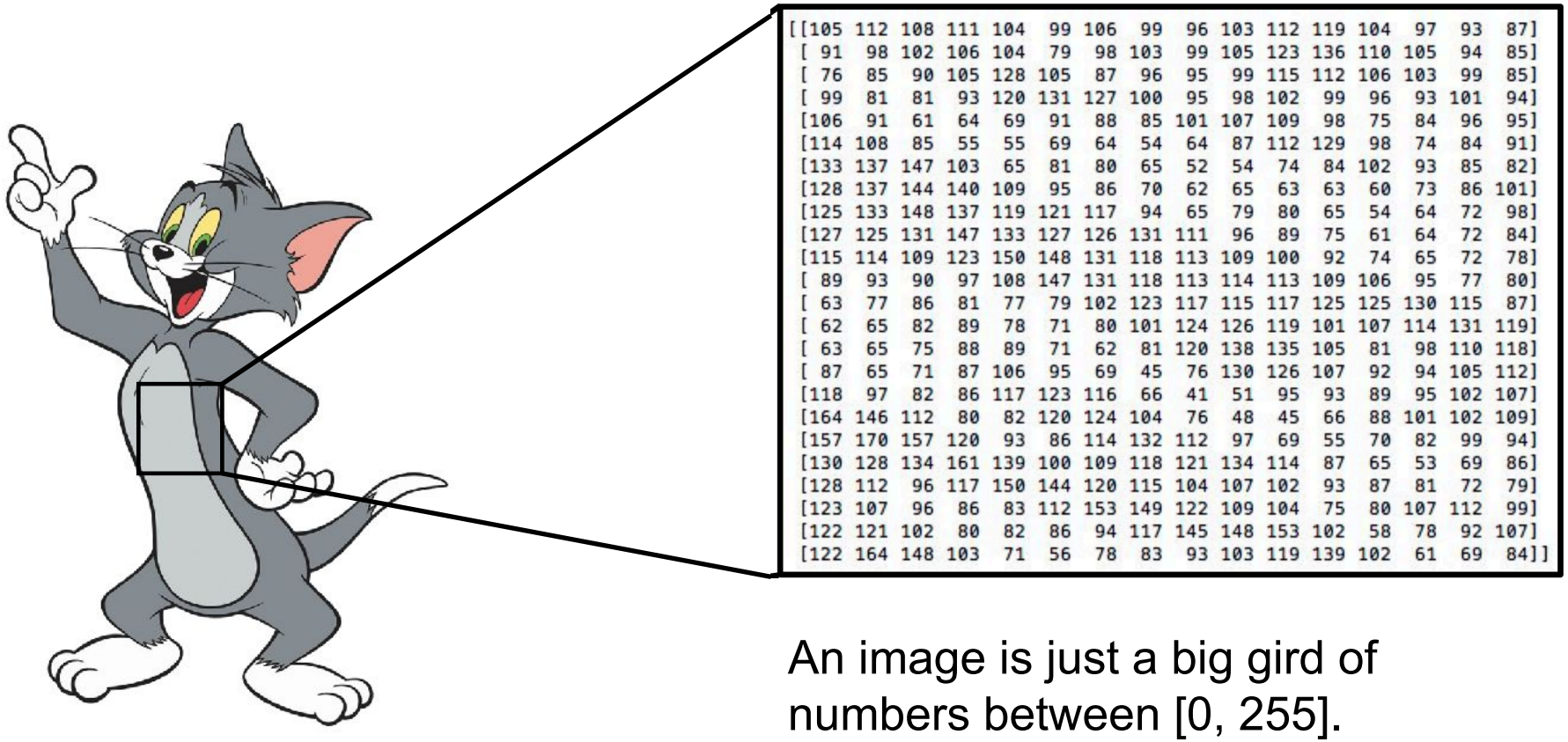


The Semantics Gap

→ Tom

What we see.

The Semantics Gap

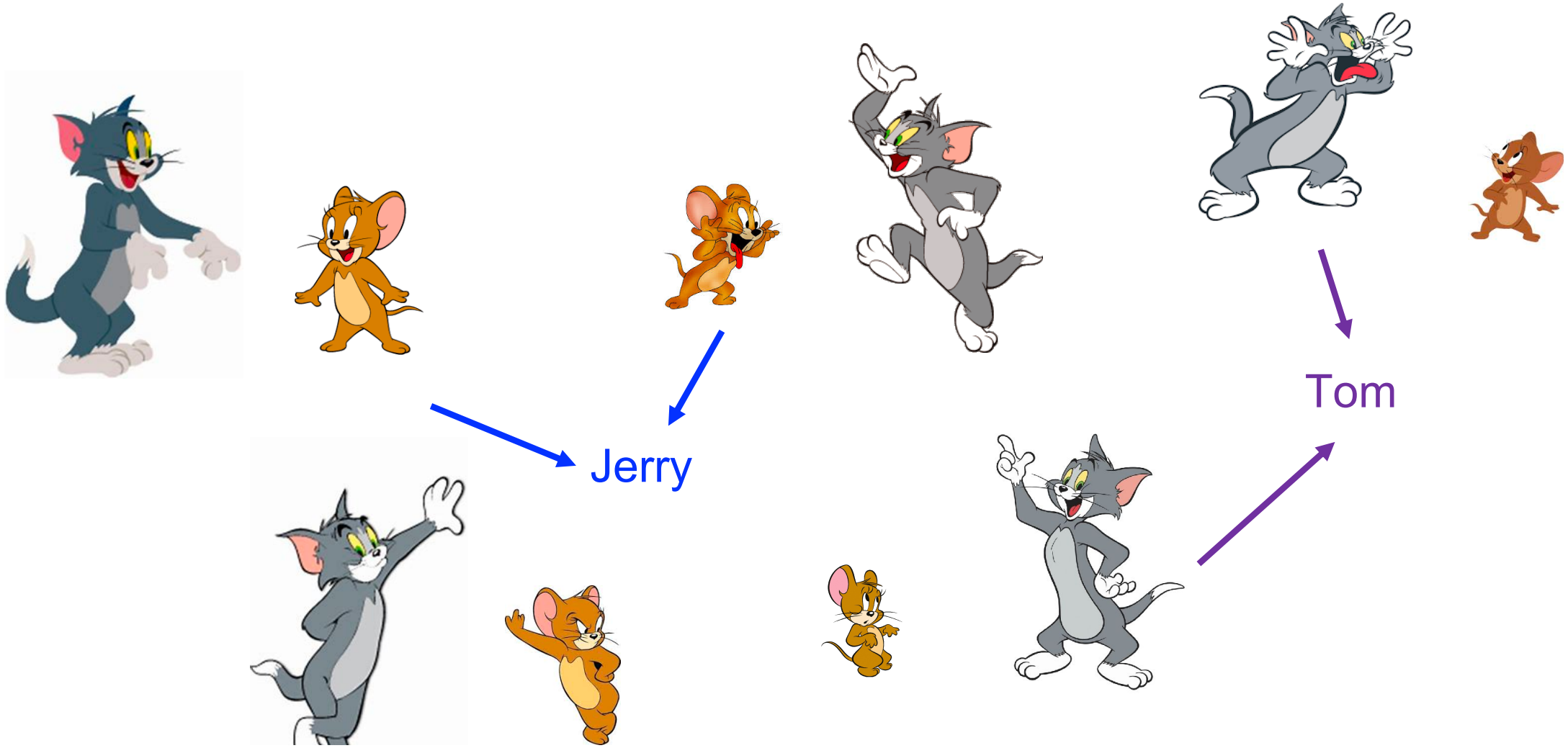


```
[[105 112 108 111 104  99 106  99  96 103 112 119 104  97  93  87]
 [ 91  98 102 106 104  79  98 103  99 105 123 136 110 105  94  85]
 [ 76  85  90 105 128 105  87  96  95  99 115 112 106 103  99  85]
 [ 99  81  81  93 120 131 127 100  95  98 102  99  96  93 101  94]
 [106  91  61  64  69  91  88  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]
 [115 114 109 123 150 148 131 118 113 109 100  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]]
```

An image is just a big gird of numbers between [0, 255].

What the computer sees.

How To Classify Two Kinds of Images?



Strawman I – Distance Metric

L1 Distance: $d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$

$$\left| \begin{bmatrix} 56 & 32 \\ 90 & 23 \end{bmatrix} - \begin{bmatrix} 10 & 20 \\ 8 & 10 \end{bmatrix} \right| = \begin{bmatrix} 46 & 12 \\ 82 & 13 \end{bmatrix} \xrightarrow{\text{sum}} 153$$

Two images are considered to be similar if their distance is smaller than a threshold (e.g., 200).

Strawman I – Distance Metric

Strawman I – Distance Metric





Belong to





But, how can we know this is *Tom*?

Strawman II – Distance Metric

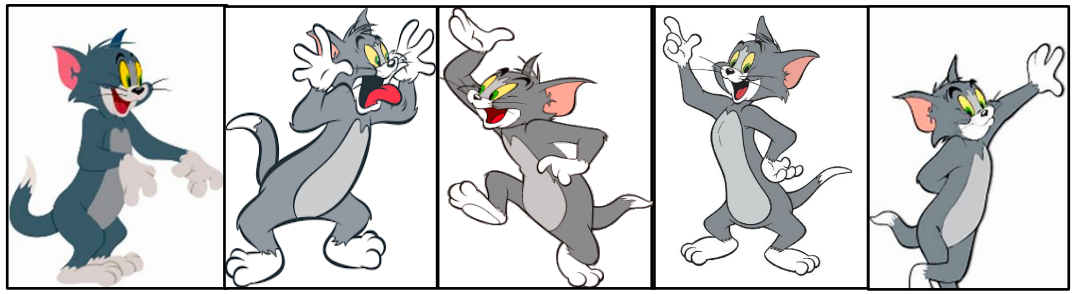
1. Collect a dataset of images and labels
2. Use machine learning to train a classifier

Memorize all data and labels

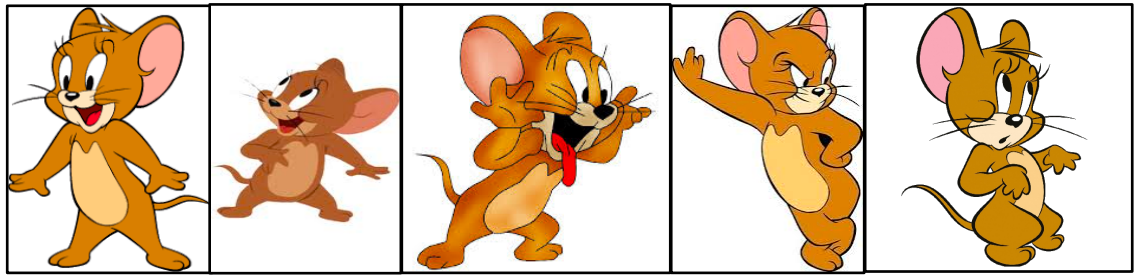
3. Evaluate the classifier on new images

Predict the the label of most similar training image

Strawman II – Distance Metric



Tom



Jerry

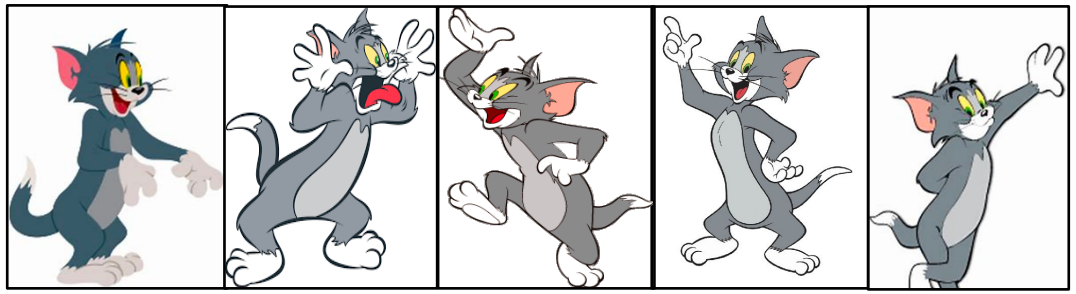
Training

Validation

Test

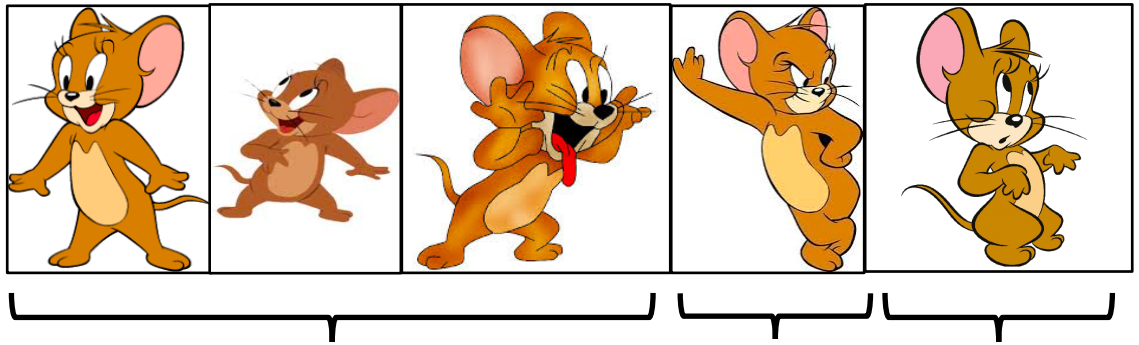
Strawman II – Distance Metric

???



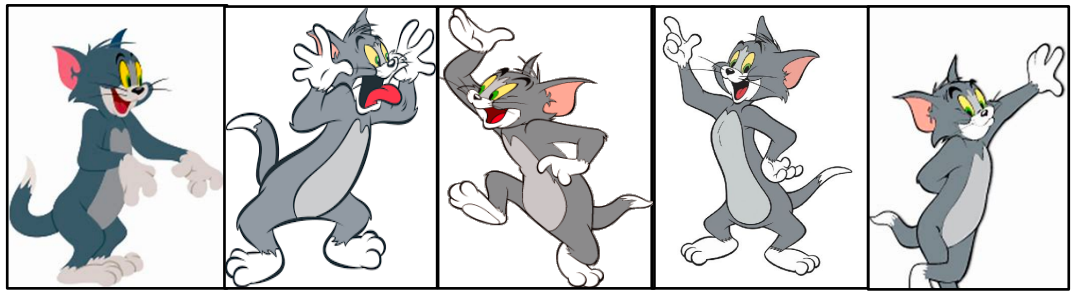
Tom

???

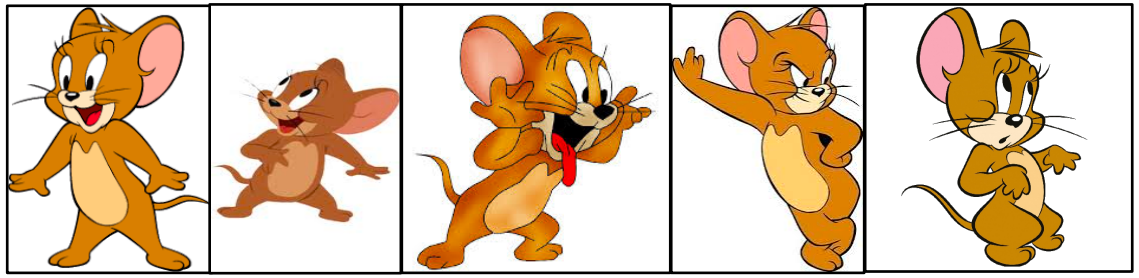


Jerry

Strawman II – Distance Metric



Tom



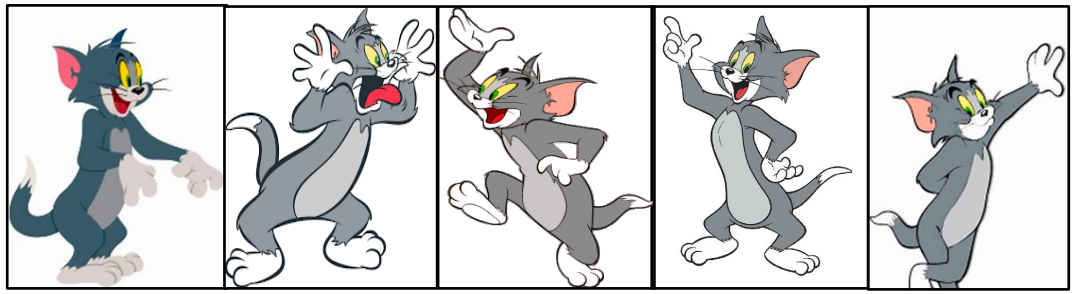
Jerry

Training

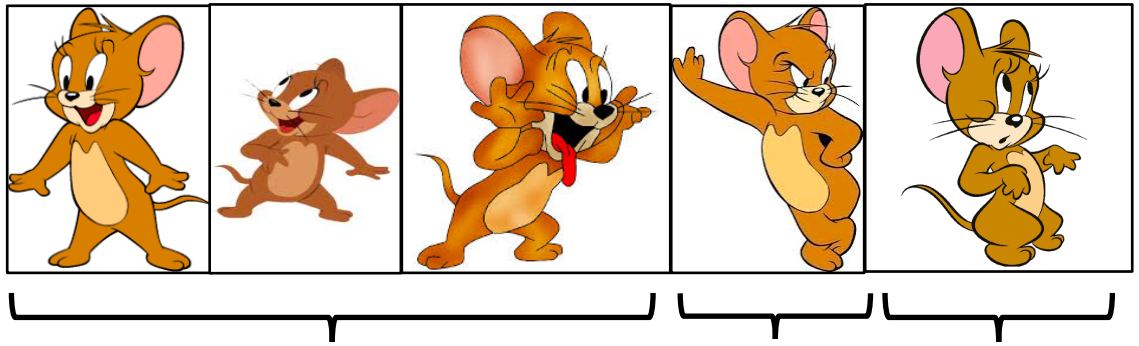
Validation

Test

Strawman II – Distance Metric



Tom

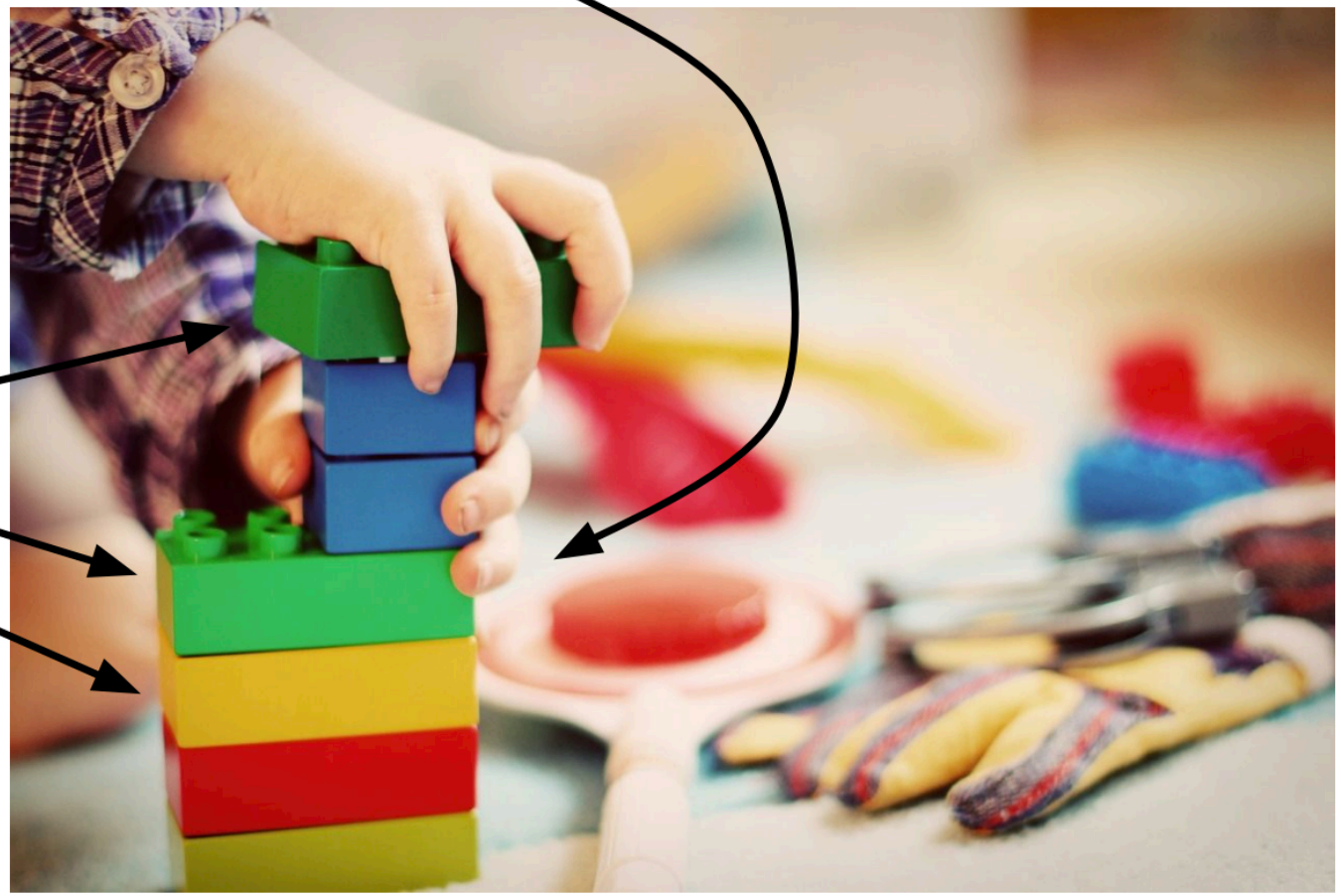


Jerry

Linear Classification

Neural Network

Linear classifiers



Parametric Approach

Image

Array of 32 x 32 x 3 numbers
(3072 numbers total)

f(x, W)

v

Parameter
or weights

Result vector
including K numbers
giving class scores

Parametric Approach: Linear Classifier

Image

Array of 32 x 32 x 3 numbers
(3072 numbers total)

$f(x, W)$

Parameter
or weights

v

Result vector
including K numbers
giving class scores

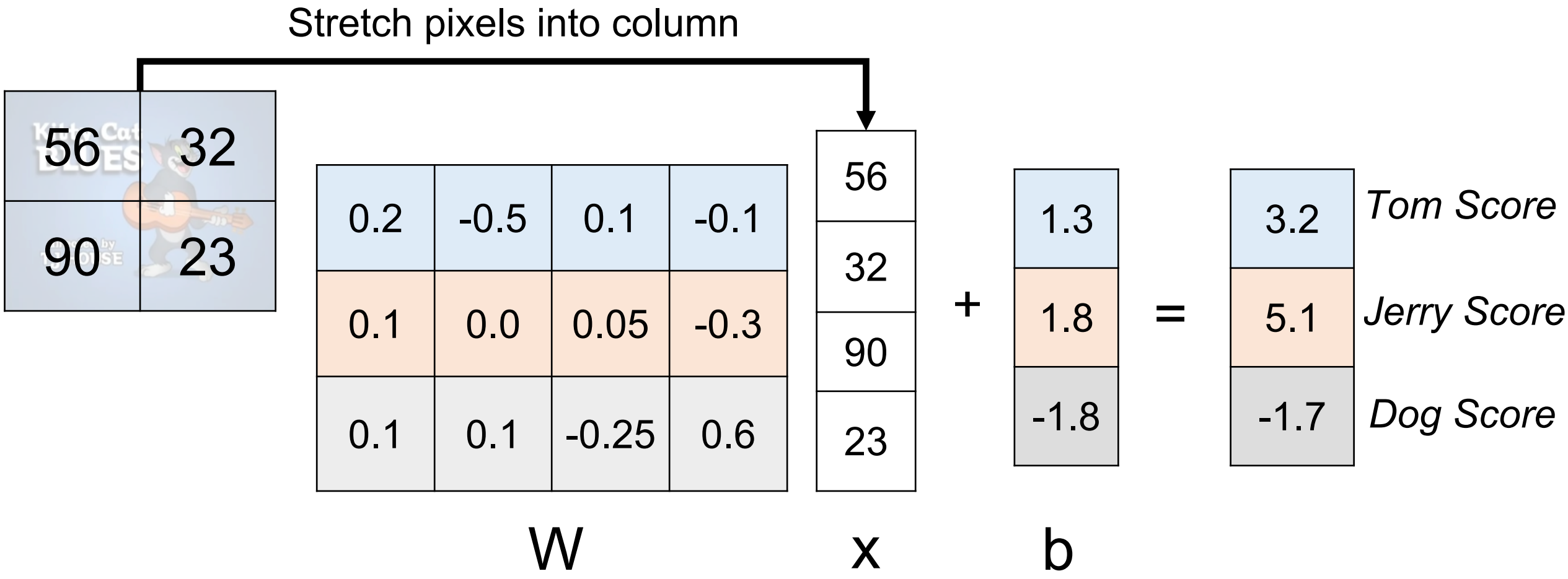
$$f(x, W) = Wx + b$$

3072 x 1

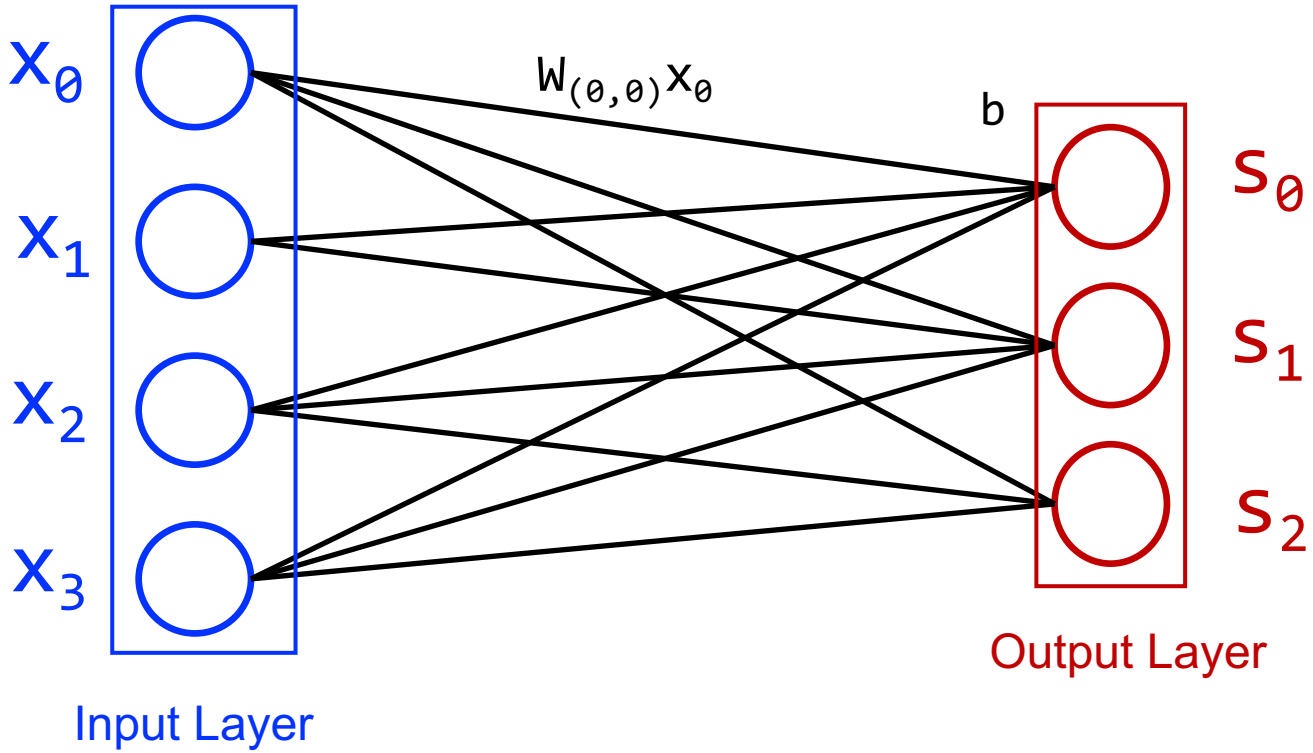
K x 3072

K x 1

Example with images of 4 pixels, and 3 classes.



Example with images of 4 pixels, and 3 classes.



Example with images of 4 pixels, and 3 classes.

Tom	3.2	1.3	2.2
Jerry	5.1	**4.9**	2.5
Dog	-1.7	2.0	-3.1

How to measure the happiness with the score?

Loss function tells how good our current model is.

Give a dataset of examples:

$$\{(x_i, y_i)\}_{i=1}^{N}$$

Where x_i's image and y_i's (integer) label

Loss over the dataset is an average of loss over examples:

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

Example with images of 4 pixels, and 3 classes.

Tom	3.2	1.3	2.2
Jerry	5.1	**4.9**	2.5
Dog	-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:

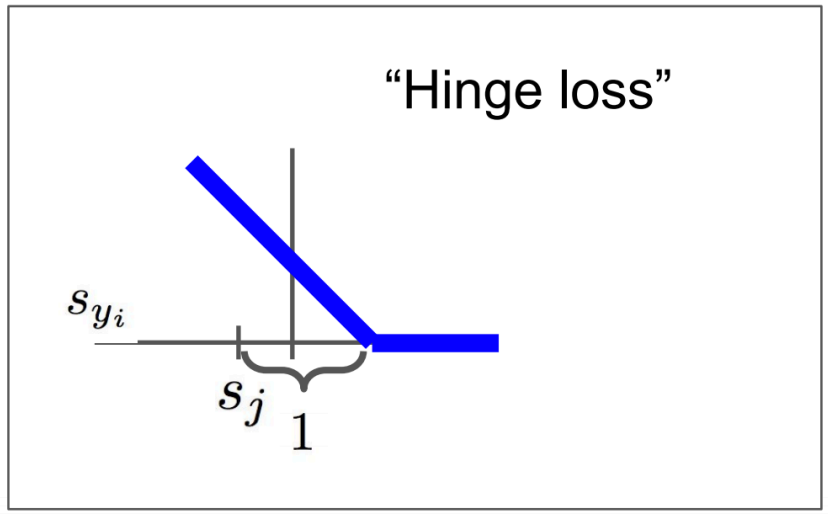
$$L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases}$$

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

Example with images of 4 pixels, and 3 classes.

Tom	3.2	1.3	2.2
Jerry	5.1	**4.9**	2.5
Dog	-1.7	2.0	-3.1

Multiclass SVM loss:



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

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

Example with images of 4 pixels, and 3 classes.

Tom	3.2	1.3	2.2
Jerry	5.1	**4.9**	2.5
Dog	-1.7	2.0	-3.1
Losses			

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:

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

Example with images of 4 pixels, and 3 classes.

Tom	3.2	1.3	2.2
Jerry	5.1	**4.9**	2.5
Dog	-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:

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

= 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

Example with images of 4 pixels, and 3 classes.

Tom	3.2	1.3	2.2
Jerry	5.1	**4.9**	2.5
Dog	-1.7	2.0	-3.1
Losses	2.9	0.0	12.9

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:

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

Example with images of 4 pixels, and 3 classes.

Tom	3.2	1.3	2.2
Jerry	5.1	**4.9**	2.5
Dog	-1.7	2.0	-3.1
Losses	2.9	0.0	12.9

Other Losses:

Softmax

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

SVM

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

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

Example with images of 4 pixels, and 3 classes.

How to minimize the loss?

Update the W.

Strawman: random search.

Tom	3.2	1.3	2.2
Jerry	5.1	**4.9**	2.5
Dog	-1.7	2.0	-3.1
Losses	2.9	0.0	12.9

Gradient Descent

How to decide the direction?

$$f(x) = \frac{f(x_0)}{0!} + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \ldots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n + R_n(x)$$

Gradient Descent

How to decide the direction?

$$L(w', x, y) = L(w, x, y) + \frac{dL(w, x, y)}{dw}(w' - w)$$

The direction should be opposite with $\frac{\boldsymbol{dL(w,x,y)}}{\boldsymbol{dw}}$

The gradient should be $\boldsymbol{\alpha}\frac{\boldsymbol{dL(w,x,y)}}{\boldsymbol{dw}}$

Learning Rate

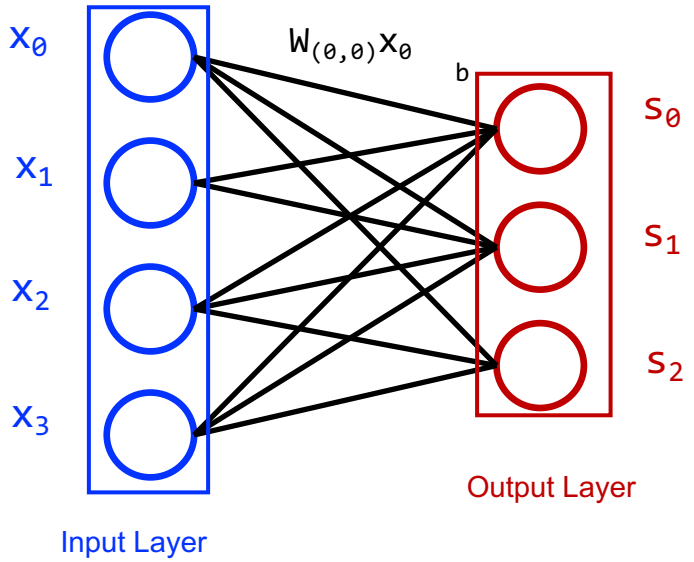
Example with images of 4 pixels, and 3 classes.

$$L(w', x, y) = L(w, x, y) + \frac{dL(w,x,y)}{dw}(w' - w)$$

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

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

Tom	3.2	1.3	2.2
Jerry	5.1	**4.9**	2.5
Dog	-1.7	2.0	-3.1
Losses	2.9	0.0	12.9



Example with images of 4 pixels, and 3 classes.

56	32
90	23

$$L(w', x, y) = L(w, x, y) + \frac{dL(w,x,y)}{dw}(w' - w)$$

Tom	3.2	1.3	2.2
Jerry	5.1	**4.9**	2.5
Dog	-1.7	2.0	-3.1
Losses	2.9	0.0	12.9

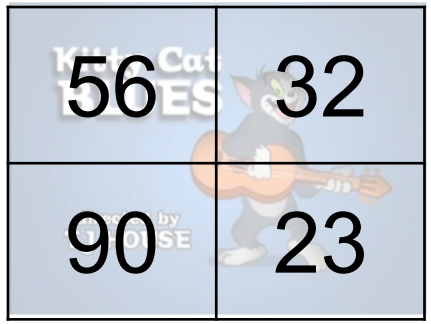
$L(W, x, y) = s_1 - s_0 + 1$
$= \sum_{i=0}^{3} (xi * W_{1,i})$
$- \sum_{i=0}^{3} (x_i * W_{0,i}) + 1$

$$\frac{dL(w,x,y)}{dw_{(0,0)}} = -x_0 = -56$$

$$\frac{dL(w,x,y)}{dw_{(0,1)}} = -x_1 = -32$$

...

Example with images of 4 pixels, and 3 classes.



Tom	3.2
Jerry	5.1
Dog	-1.7
Losses	2.9

0.2	-0.5	0.1	-0.1
0.1	0.0	0.05	-0.3
0.1	0.1	-0.25	0.6

$$L(w', x, y) = L(w, x, y) + \frac{dL(w,x,y)}{dw}(w' - w)$$

$$L(W,\ x,\ y) = s_1 - s_0 + 1 = \sum_{i=0}^{3} (xi_{*} W_{1i}) - \sum_{i=0}^{3} (x_{i*} W_{0i}) + 1$$

$w'_{(0,0)} = w_{(0,0)} + 56 * 0.01 = 0.76$

$w'_{(0,1)} = w_{(0,1)} + 32 * 0.01 = -0.18$

$w'_{(0,2)} = w_{(0,2)} + 90 * 0.01 = 1.1$

$w'_{(0,3)} = w_{(0,3)} + 23 * 0.01 = 0.13$

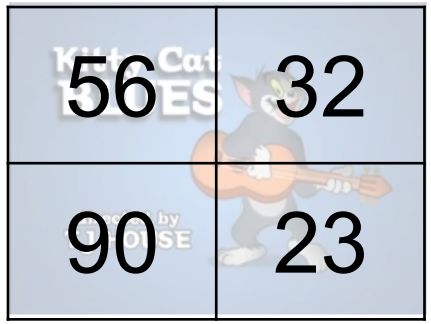
learning rate: 0.01

$$\frac{dL(w,x,y)}{dw_{(0,0)}} = -x_0 = -56$$

$$\frac{dL(w,x,y)}{dw_{(0,1)}} = -x_1 = -32$$

...

Example with images of 4 pixels, and 3 classes.



0.2	-0.5	0.1	-0.1
0.1	0.0	0.05	-0.3
0.1	0.1	-0.25	0.6

$$L(w', x, y) = L(w, x, y) + \frac{dL(w,x,y)}{dw}(w' - w)$$

$$L(W,\ x,\ y) = s_1 - s_0 + 1$$
$$= \sum_{i=0}^{3} (xi_{*} W_{1i})$$
$$- \sum_{i=0}^{3} (x_{i\ *} W_{0i}) + 1$$

Tom	3.2
Jerry	5.1
Dog	-1.7
Losses	2.9

learning rate: 0.01

$w'_{(0,0)} = w_{(0,0)} + 56 * 0.01 = 0.76$

$w'_{(0,1)} = w_{(0,1)} + 32 * 0.01 = -0.18$

$w'_{(0,2)} = w_{(0,2)} + 90 * 0.01 = 1.1$

$w'_{(0,3)} = w_{(0,3)} + 23 * 0.01 = 0.13$

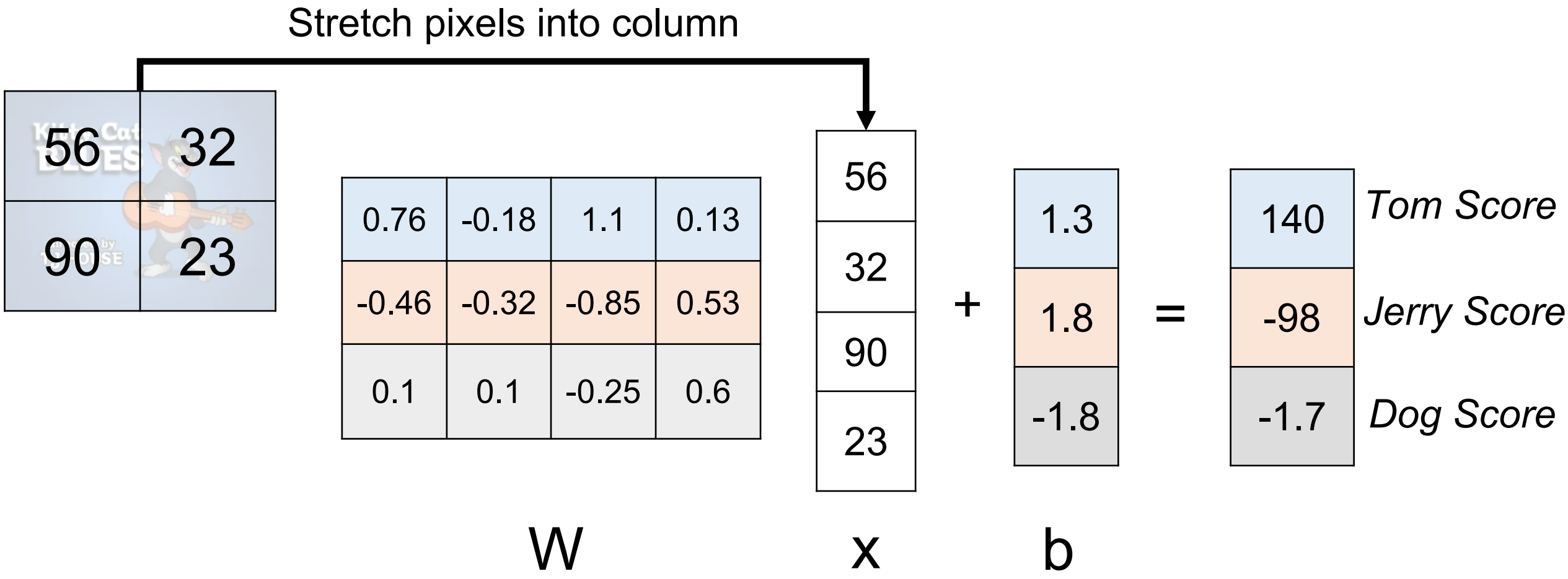
$w'_{(1,0)} = w_{(1,0)} - 56 * 0.01 = -0.46$

$w'_{(1,1)} = w_{(1,1)} - 32 * 0.01 = -0.32$

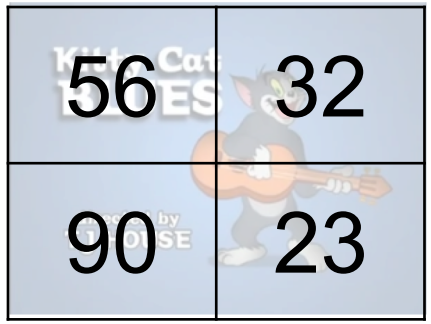
$w'_{(1,2)} = w_{(1,2)} - 90 * 0.01 = -0.85$

$w'_{(1,3)} = w_{(1,3)} - 23 * 0.01 = 0.53$

Example with images of 4 pixels, and 3 classes.



Example with images of 4 pixels, and 3 classes.

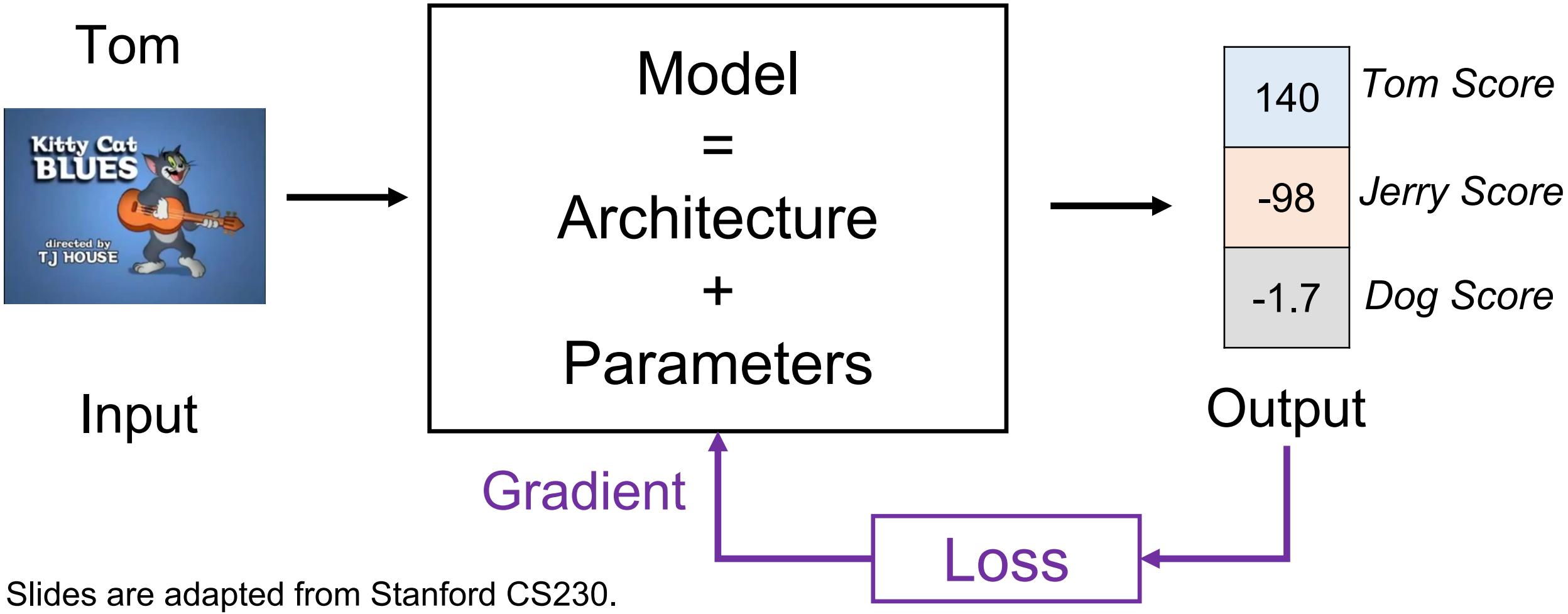


0.2	-0.5	0.1	-0.1
0.1	0.0	0.05	-0.3
0.1	0.1	-0.25	0.6

0.76	-0.18	1.1	0.13
-0.46	-0.32	-0.85	0.53
0.1	0.1	-0.25	0.6

Tom	3.2	140
Jerry	5.1	-98
Dog	-1.7	-1.7
Losses	2.9	0

Learning Process

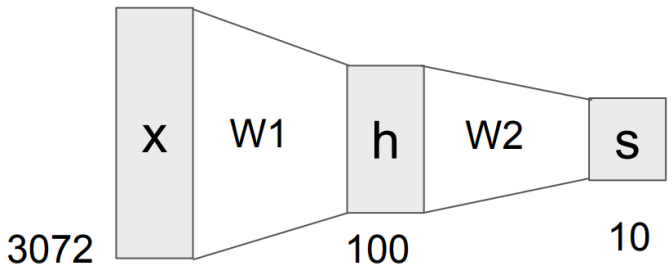


Slides are adapted from Stanford CS230.

Neural Network

$$s = W_2 f(x, W_1)$$

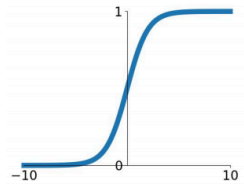
Active function



Activation functions

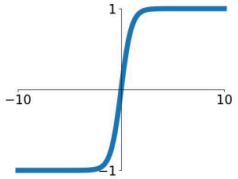
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



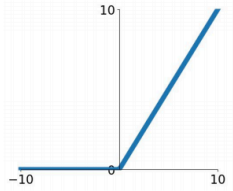
tanh

$$\tanh(x)$$



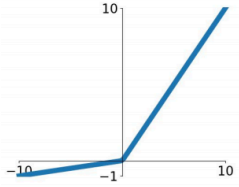
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

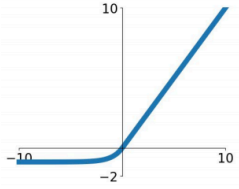


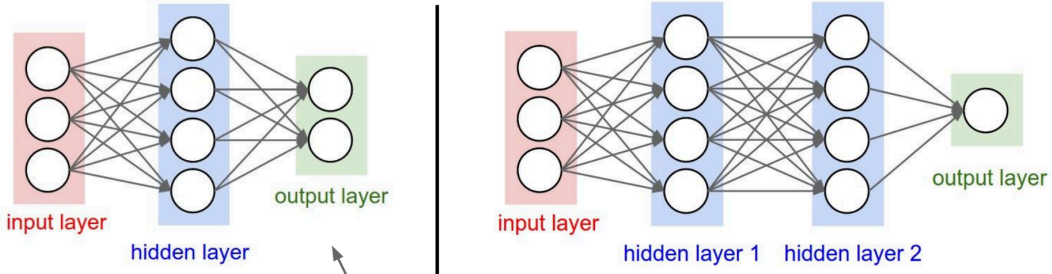
Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$





"2-layer Neural Net", or
"1-hidden-layer Neural Net"

"3-layer Neural Net", or
"2-hidden-layer Neural Net"

"Fully-connected" layers

Back Propagation

Chain rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}$$

Thanks & Questions