Image Classification, Linear Classifiers and Losses

Deep Learning
26 September 2024
Profs. Luigi Cinque, Fabio Galasso and Marco Raoul Marini

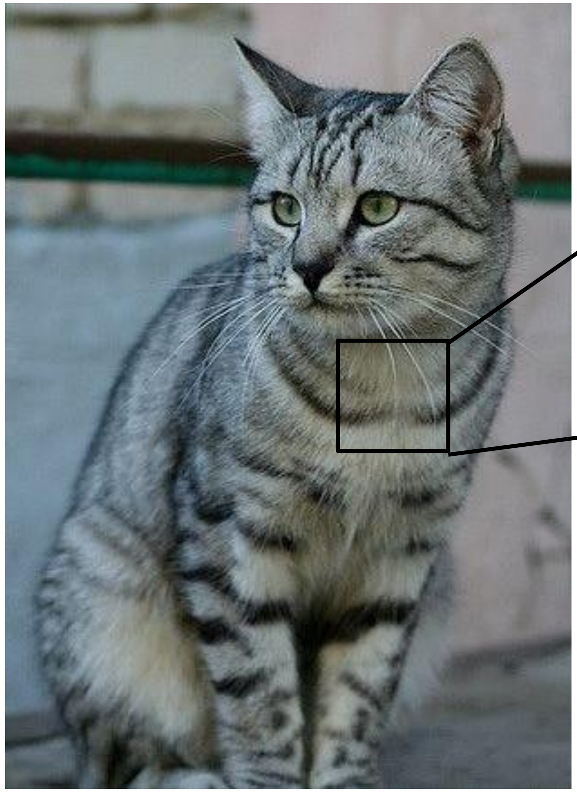
SAPIENZA
UNIVERSITÀ DI ROMA

Image Classification: A core task in Computer Vision

(assume given set of discrete labels)
{dog, cat, truck, plane, ...}

$\longrightarrow$ cat

The Problem: Semantic Gap



This image by Nikita is licensed under CC-BY 2.0

```
[[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]]
```

What the computer sees

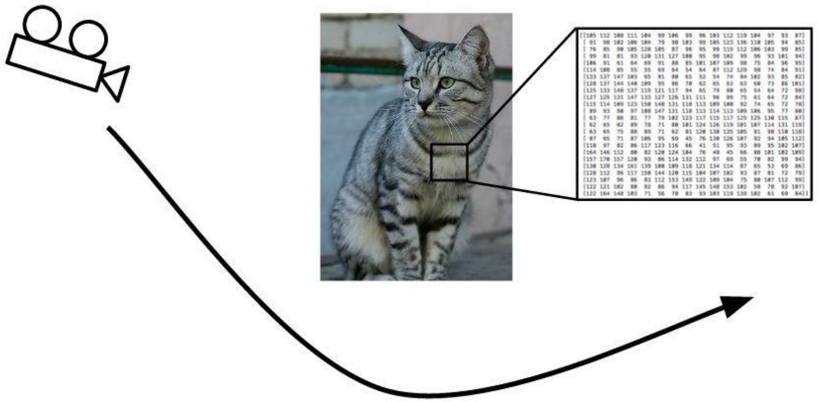
An image is just a big grid of numbers between [0, 255]:

e.g. 800 x 600 x 3
(3 channels RGB)

Luigi Cinque, Fabio Galasso and Marco Raoul Marini

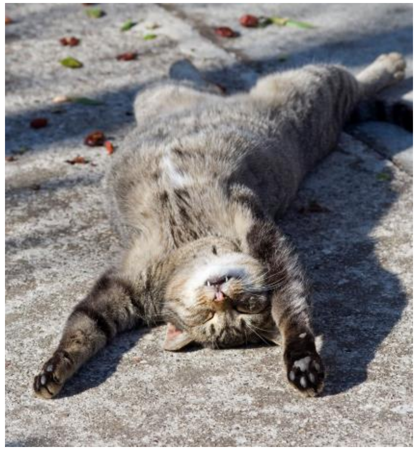
Recall: Challenges of recognition

Viewpoint



Illumination

Deformation



Occlusion

Clutter

Intraclass Variation

An image classifier

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

Unlike e.g. sorting a list of numbers,

no obvious way to hard-code the algorithm for recognizing a cat, or other classes.

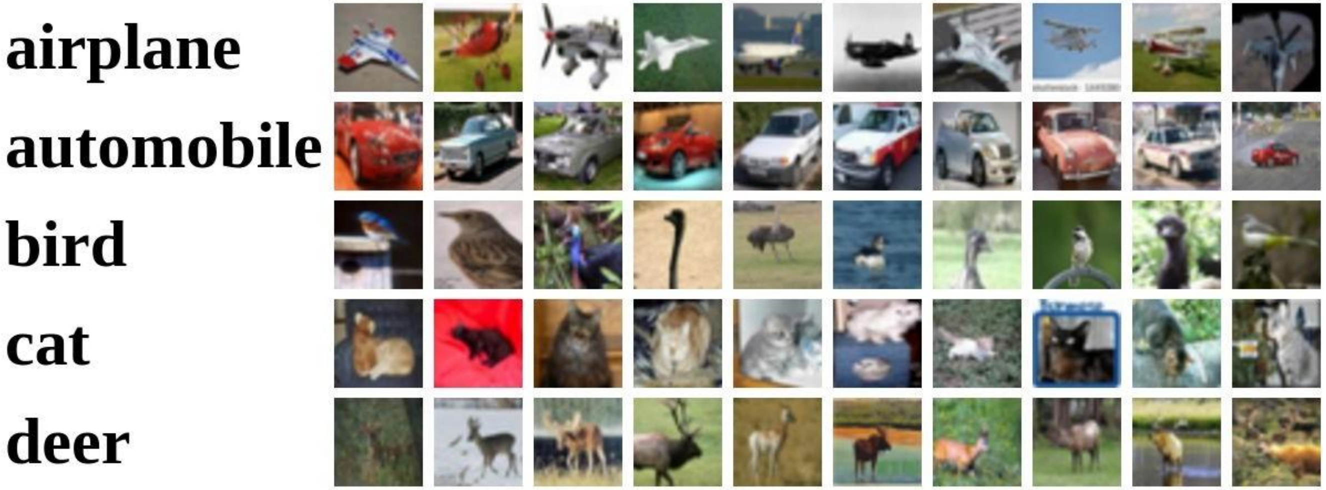
Machine Learning: Data-Driven Approach

1. Collect a dataset of images and labels
2. Use Machine Learning to train a classifier
3. Evaluate the classifier on new images

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

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

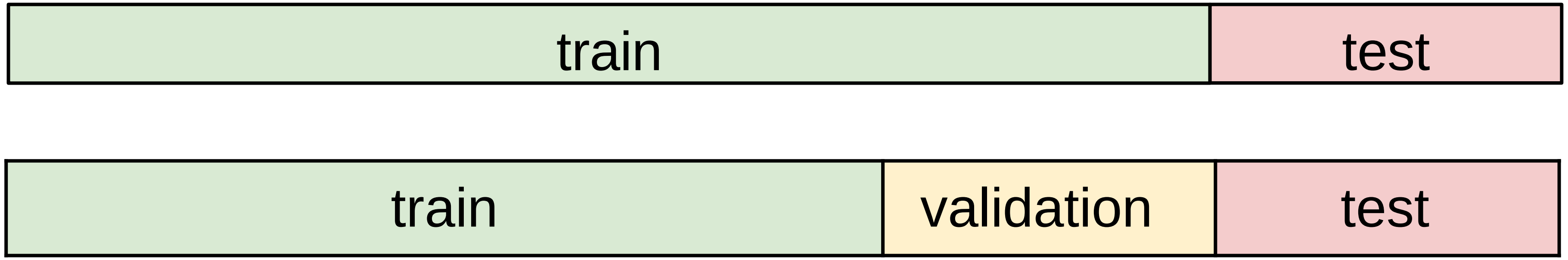
Example training set



Luigi Cinque, Fabio Galasso and Marco Raoul Marini

Machine Learning: Data-Driven Approach

1. Collect a dataset of images and labels
2. Use Machine Learning to train a classifier
3. Evaluate the classifier on new images



Recall CIFAR10

50,000 training images
each image is **32x32x3**

10,000 test images.

Luigi Cinque, Fabio Galasso and Marco Raoul Marini

Parametric Approach: Linear Classifier

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

Image

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

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

W

parameters
or weights

Luigi Cinque, Fabio Galasso and Marco Raoul Marini

Parametric Approach: Linear Classifier

3072x1

$$\boxed{f(x,W) = \boxed{W}\boxed{x} + }\boxed{b}$$ 10x1

10x1 10x3072

Image

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

$f(x,W)$ → **10** numbers giving class scores

W

parameters
or weights

Neural Network

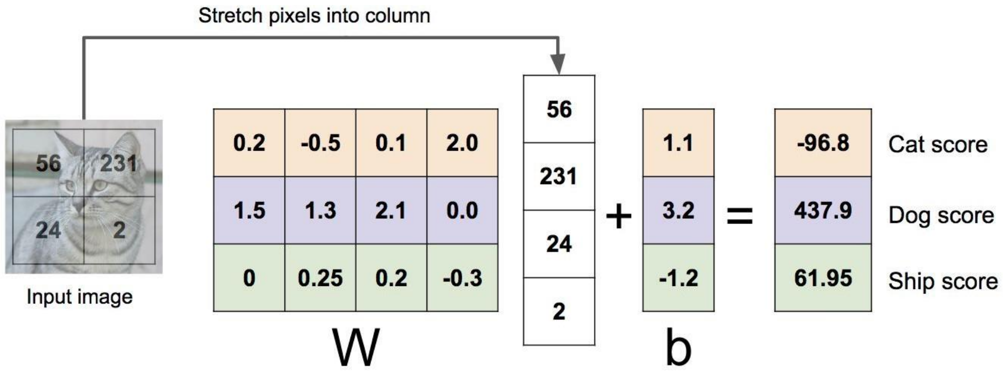
Linear
classifiers

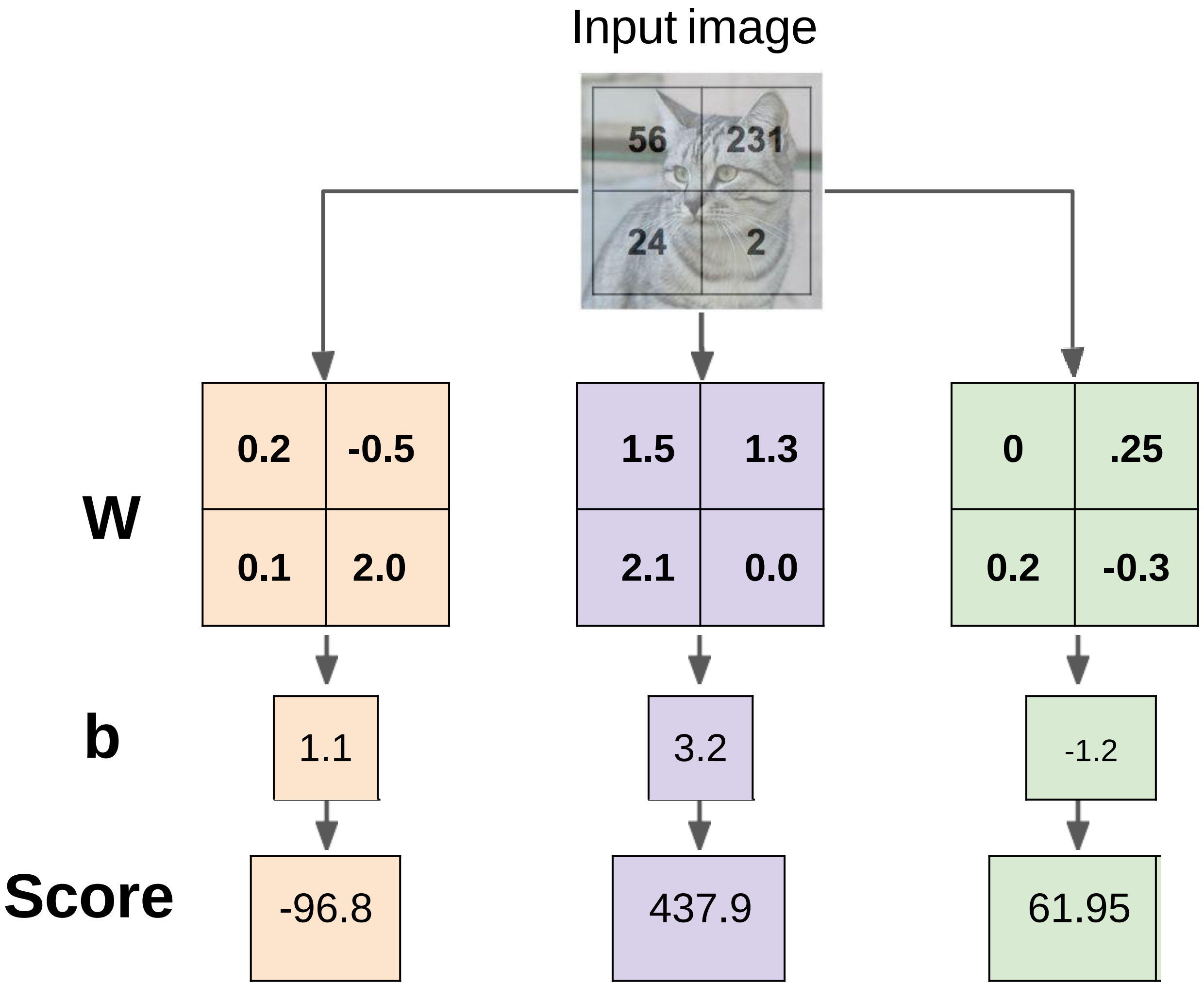
This image is CC0 1.0 public domain

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

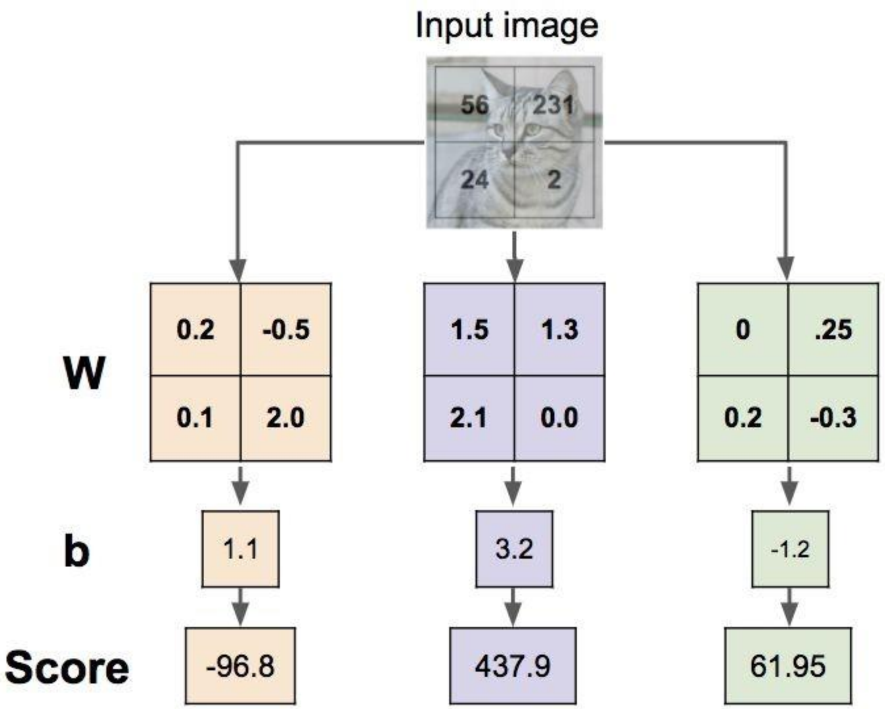
Algebraic Viewpoint

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

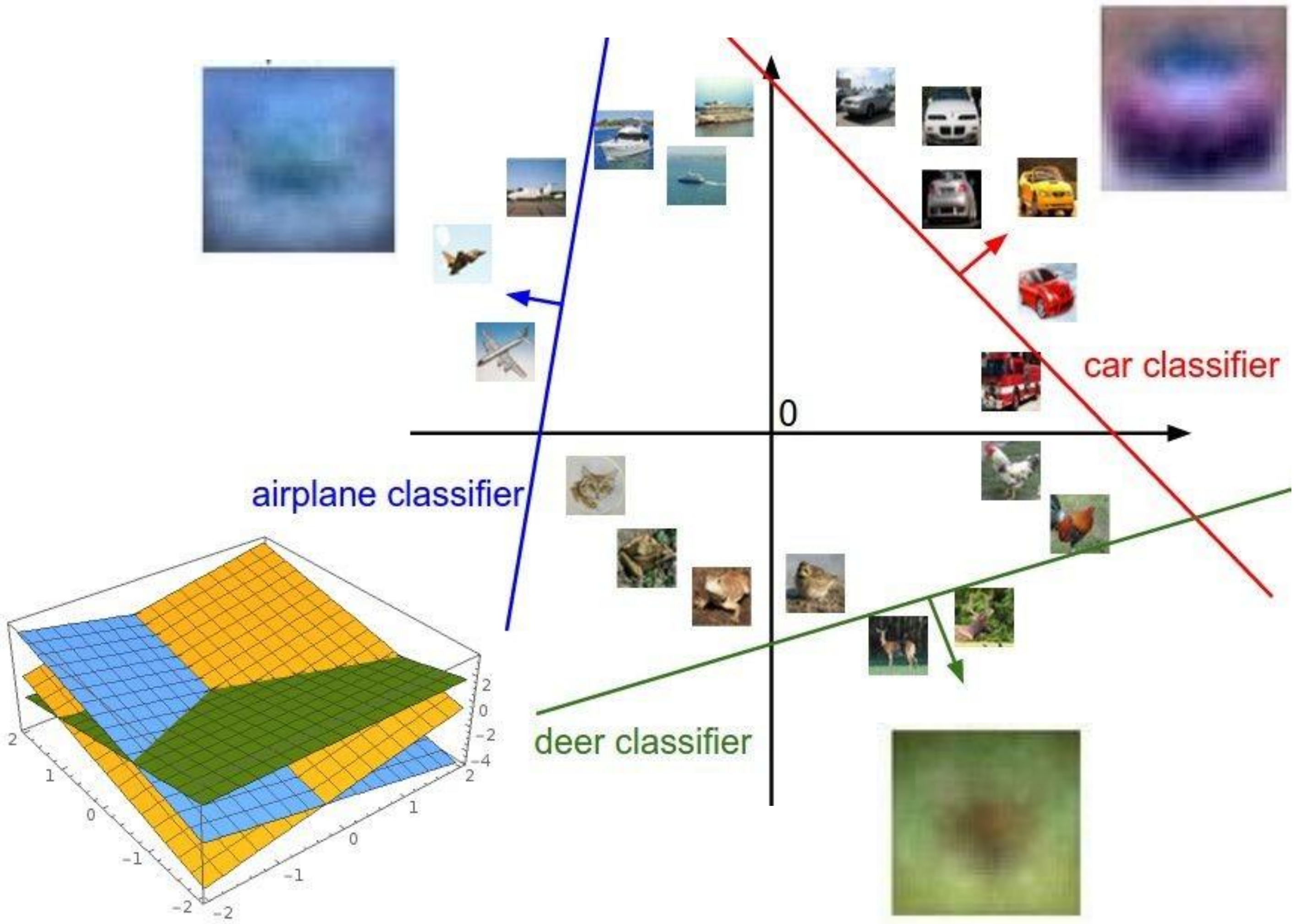




Interpreting a Linear Classifier: Visual Viewpoint



Interpreting a Linear Classifier: Geometric Viewpoint



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

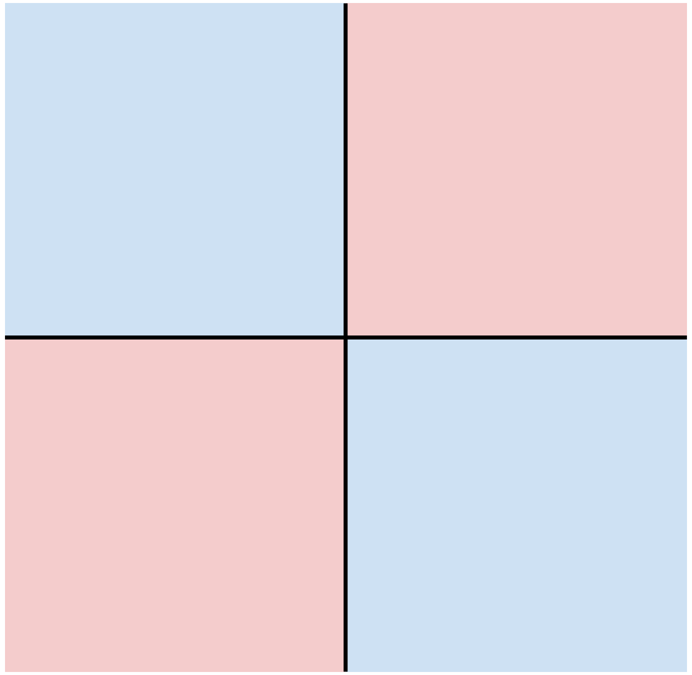
Array of **32x32x3** numbers
(3072 numbers total)

Plot created using [Wolfram Cloud]

Hard cases for a linear classifier

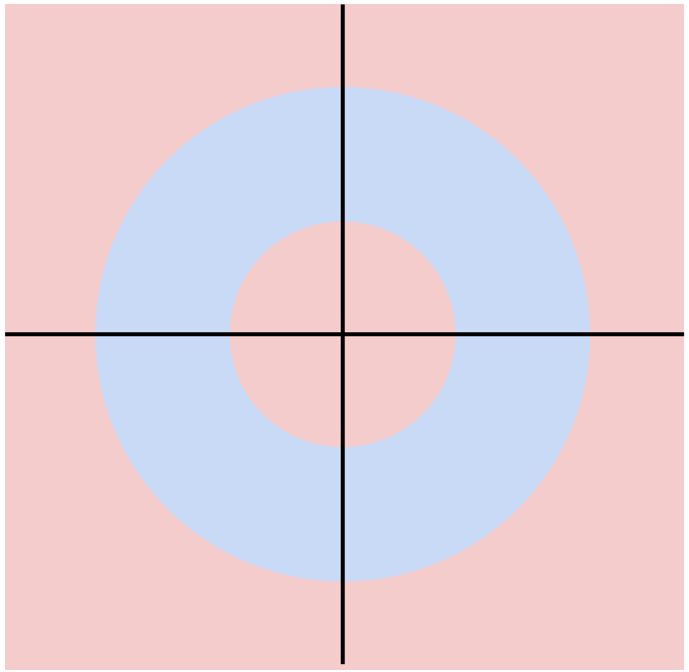
Class 1:
First and third quadrants

Class 2:
Second and fourth quadrants



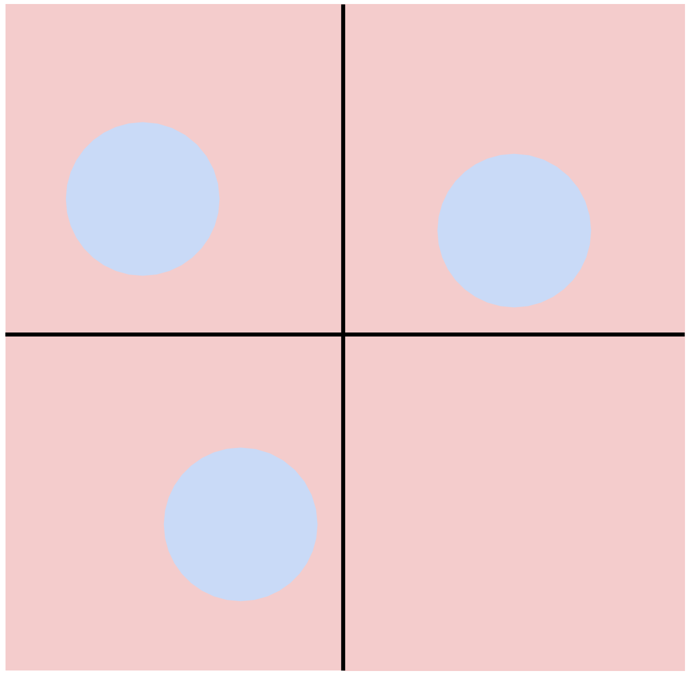
Class 1:
1 <= L2 norm <= 2

Class 2:
Everything else



Class 1:
Three modes

Class 2:
Everything else

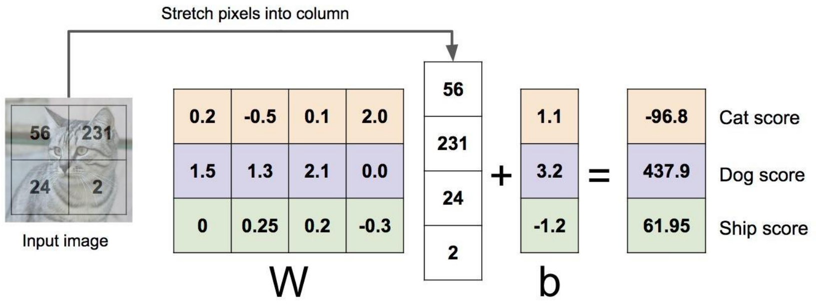


Luigi Cinque, Fabio Galasso and Marco Raoul Marini

Linear Classifier: Three Viewpoints

Algebraic Viewpoint

$f(x,W) = Wx$

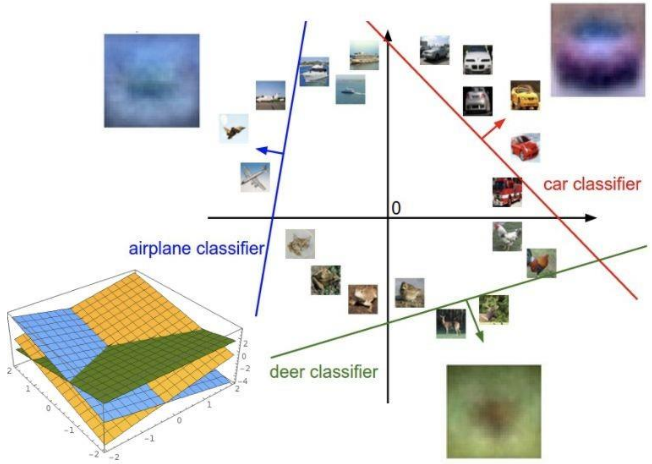


Visual Viewpoint

One template
per class

Geometric Viewpoint

Hyperplanes
cutting up space



So far: Defined a (linear) <u>score function</u> $f(x,W) = Wx + b$

Example class scores for 3 images for some W:

How can we tell whether this W is good or bad?

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

Loss Functions and Optimization

Suppose: 3 training examples, 3 classes.
With 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**

Luigi Cinque, Fabio Galasso and Marco Raoul Marini

Suppose: 3 training examples, 3 classes.
With 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**

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

Given a dataset of examples

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

Where x_i is image and
y_i is (integer) label

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

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

Luigi Cinque, Fabio Galasso and Marco Raoul Marini

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

cat	**3.2**
car	5.1
frog	-1.7

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

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

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

cat	**3.2**
car	5.1
frog	-1.7

Luigi Cinque, Fabio Galasso and Marco Raoul Marini

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

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

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

Probabilities must be >= 0

Probabilities must sum to 1

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

	Unnormalized log-probabilities / logits		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

→ $L_i = -\log(0.13)$
$= \mathbf{2.04}$

Maximum Likelihood Estimation
Choose weights to maximize the likelihood of the observed data

Luigi Cinque, Fabio Galasso and Marco Raoul Marini

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

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

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

Probabilities must be >= 0

Probabilities must sum to 1

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

	Unnormalized log-probabilities / logits	$\xrightarrow{\text{exp}}$	unnormalized probabilities	$\xrightarrow{\text{normalize}}$	probabilities		Correct probs
cat	**3.2**		**24.5**		**0.13**	$\rightarrow$ compare $\leftarrow$	**1.00**
car	5.1		164.0		0.87		0.00
frog	-1.7		0.18		0.00		0.00

Cross Entropy

$$H(\underline{P}, \underline{Q}) = H(p) + D_{KL}(P \| Q)$$

Luigi Cinque, Fabio Galasso and Marco Raoul Marini

Let's double check...

- Entropy & KL-divergence:

$$H(P^t) = -\sum_y P^t(y) \log P^t(y)$$

$$D_{KL}(P^t||Q) = \sum_y P^t(y) \log \frac{P^t(y)}{Q(y)}$$

- Cross Entropy the sum of both:

$$\begin{aligned} H(P^t, Q) &= H(P^t) + D_{KL}(P^t||Q) \\ &= \sum_y P^t(y) \left(\log \frac{P^t(y)}{Q(y)} - \log P^t(y) \right) \\ &= -\sum_y P^t(y) \log Q(y) \end{aligned}$$

Luigi Cinque, Fabio Galasso and Marco Raoul Marini

Let's double check...

- Cross Entropy in our classification case:
  - Target "distribution" / output:

$$P^t(y) = \begin{cases} 1 & y = y_i \\ 0 & y \neq y_i \end{cases}$$

  - Output of the network:

$$Q(y|x_i) = P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

- Then Cross Entropy Loss for image x_i

$$\begin{aligned} L_i = L(x_i) &= -\sum_y P^t(y) \log Q(y|x_i) \\ &= -1 \cdot \log Q(y_i|x_i) \\ &= -\log P(Y = y_i|X = x_i) \end{aligned}$$

Regularization

λ = regularization strength (hyperparameter)

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)} + \underbrace{\lambda R(W)}$$

Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too* well on training data

Simple examples

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

More complex:

Dropout

Batch normalization

Stochastic depth, fractional pooling, etc

Regularization

λ = regularization strength (hyperparameter)

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}_{} + \underbrace{\lambda R(W)}_{}$$

Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too* well on training data

Why regularize?

- Express preferences over weights
- Make the model *simple* so it works on test data
- Improve optimization by adding curvature

Luigi Cinque, Fabio Galasso and Marco Raoul Marini

Regularization: Expressing Preferences

L2 Regularization

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

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

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

$$w_2 = \boxed{[0.25, 0.25, 0.25, 0.25]}$$

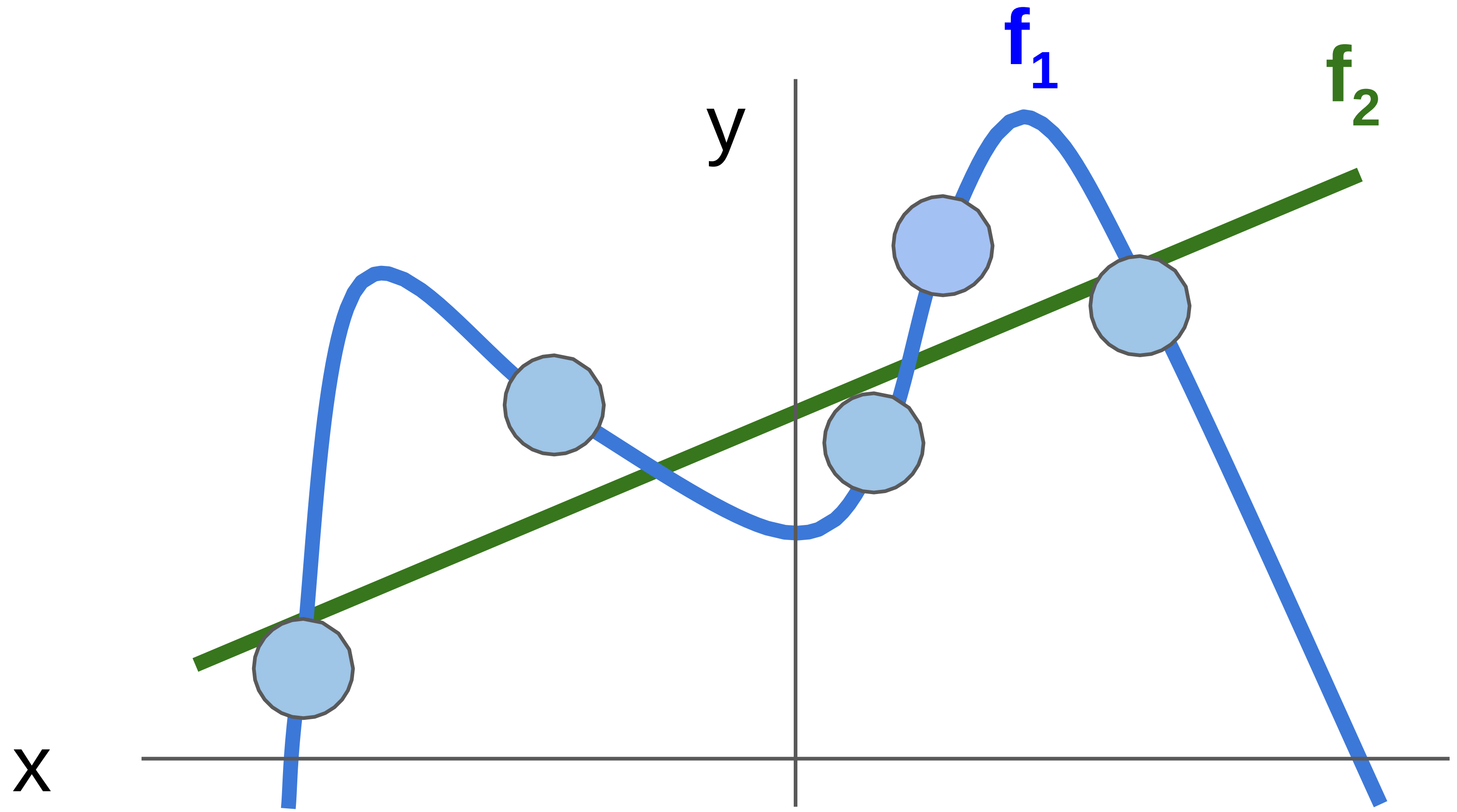
L2 regularization likes to “spread out” the weights

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

$$R(w_1) = 1^2 + 0^2 + \cdots = 1^2$$

$$R(w_2) = 0.25^2 + 0.25^2 + \cdots = 4 * 0.25^2 = 0.25$$

Regularization: Prefer Simpler Models

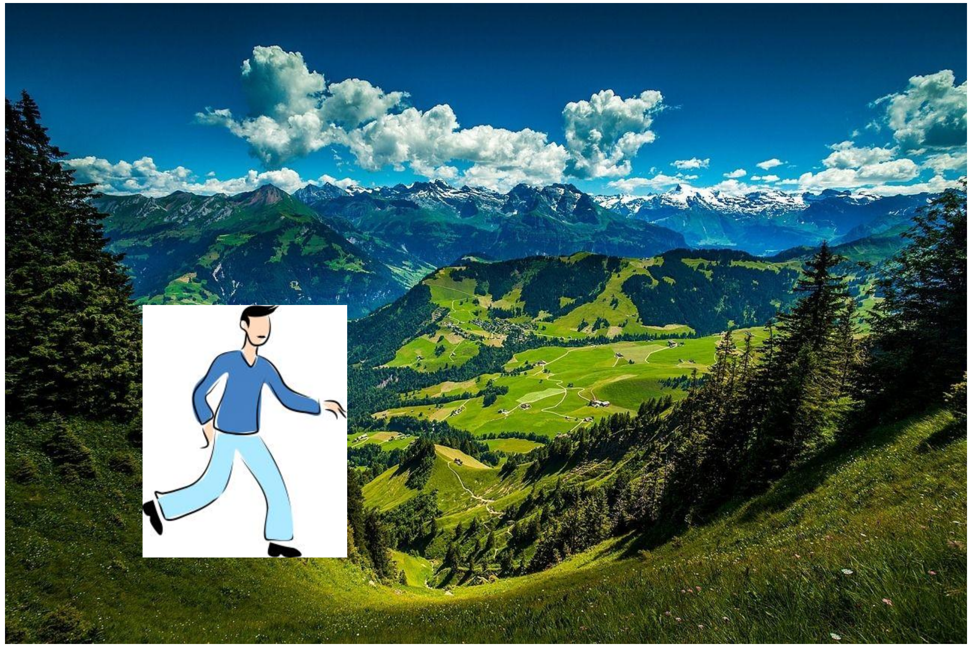


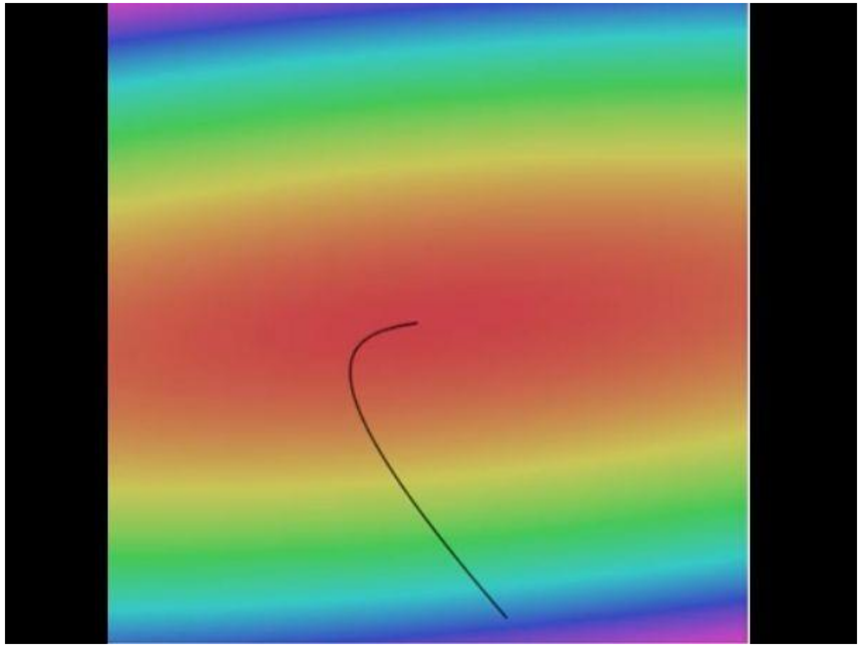
Regularization pushes against fitting the data *too* well so we don't fit noise in the data

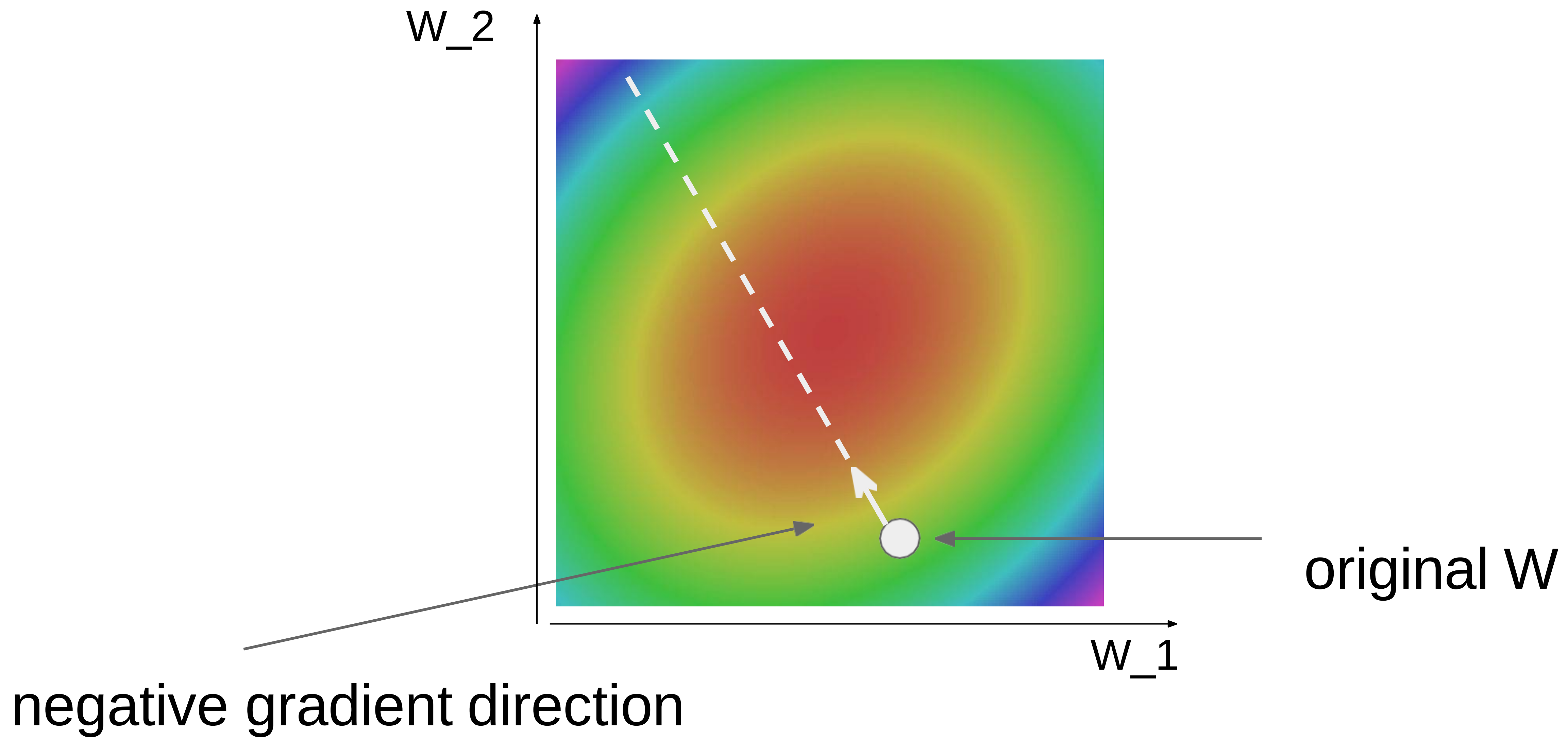
Luigi Cinque, Fabio Galasso and Marco Raoul Marini

Optimization

Finding the best W: Optimize with Gradient Descent






W_2
W_1
original W
negative gradient direction

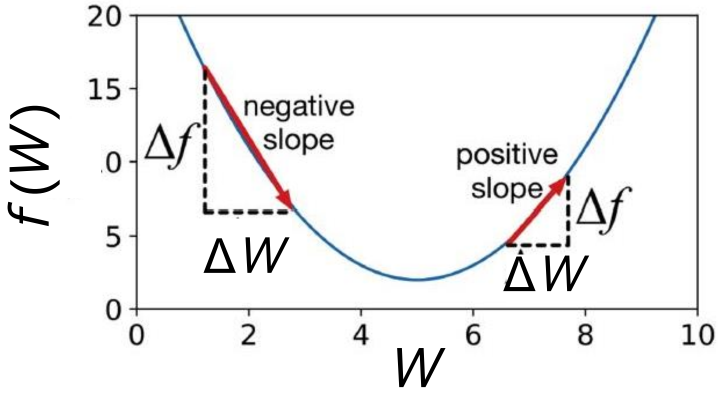
Gradient Descent

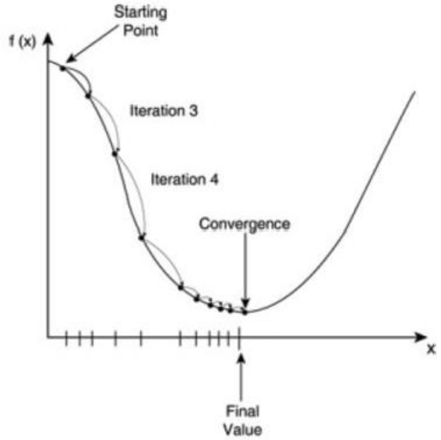
General gradient descent: Start with initial point W_0

Sequence: $W_{t+1} = W_t + \alpha_t d_t$

Steepest Descent:

$d_t = -\nabla f(W_t)$ (we move in the opposite direction of the gradient).






```
# Vanilla Gradient Descent

while True:
  weights_grad = evaluate_gradient(loss_fun, data, weights)
  weights += - step_size * weights_grad # perform parameter update
```


Luigi Cinque, Fabio Galasso and Marco Raoul Marini

Gradient Descent Variants

- Assume Loss to be: $\mathrm{L}(W) = \frac{1}{n}\sum_i^n L_i(W)$
  - with n the number of training samples
  - L_i the loss for training sample x_i
- Stochastic Gradient Descent:
  - randomly choose one training sample x_i
  - update weights based on loss $L_i(W)$
- Mini-batch training:
  - process a subset of training samples $M \subset \{1, \dots, n\}$
  - update weights based on $L_M(W) = \frac{1}{|M|}\sum_{i \in M} L_i(W)$
- Batch training:
  - process all training samples
  - update weights based on $\mathrm{L}(W) = \frac{1}{n}\sum_i^n L_i(W)$

Stochastic Gradient Descent (SGD)

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

$$\nabla_W L(W) = \frac{1}{N} \sum_{i=1}^{N} \nabla_W L_i(x_i, y_i, W) + \lambda \nabla_W R(W)$$

Full sum expensive when N is large!

Approximate sum using a **minibatch** of examples
32 / 64 / 128 common

```
# Vanilla Minibatch Gradient Descent

while True:
  data_batch = sample_training_data(data, 256) # sample 256 examples
  weights_grad = evaluate_gradient(loss_fun, data_batch, weights)
  weights += - step_size * weights_grad # perform parameter update
```


Luigi Cinque, Fabio Galasso and Marco Raoul Marini

Gradient Descent

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Numerical gradient: slow :(, approximate :(, easy to write :)
Analytic gradient: fast :), exact :), error-prone :(

In practice: Derive analytic gradient, check your implementation with numerical gradient

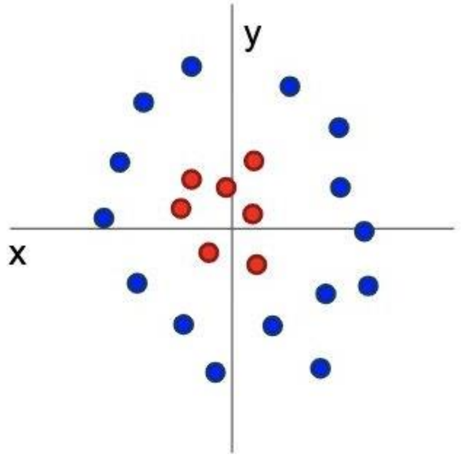
Image Features

Problem: Linear Classifiers are not very powerful

Visual Viewpoint

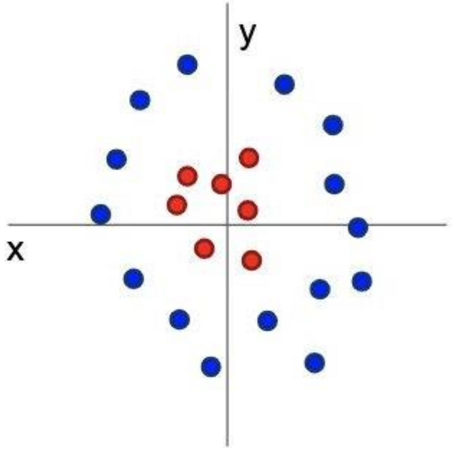
Linear classifiers learn one template per class

Geometric Viewpoint



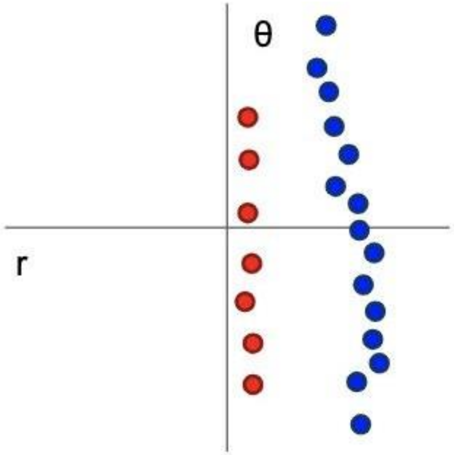
Linear classifiers can only draw linear decision boundaries

One Solution: Feature Transformation

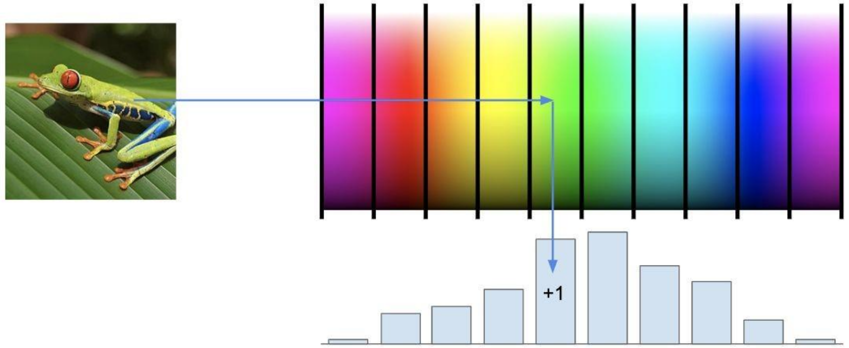


$f(x, y) = (r(x, y), \theta(x, y))$

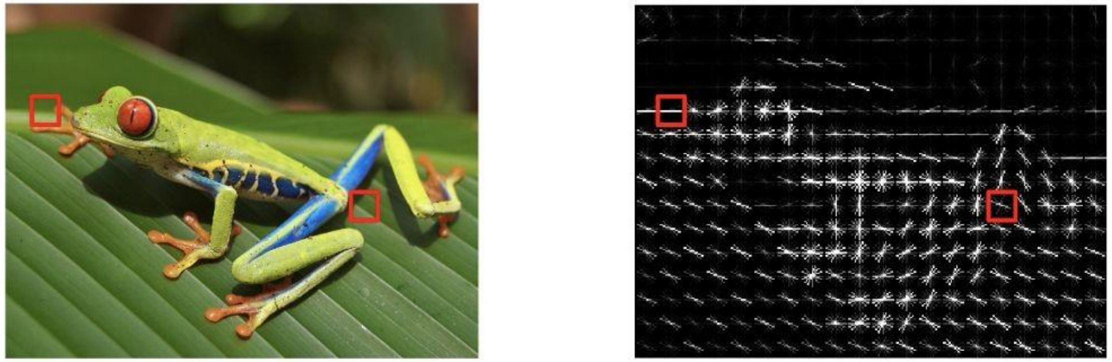
Transform data with a cleverly chosen **feature transform** f, then apply linear classifier



Color Histogram



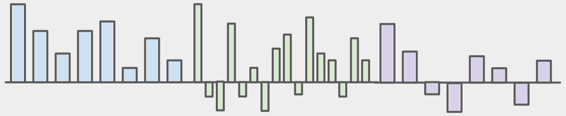
Histogram of Oriented Gradients (HoG)



Luigi Cinque, Fabio Galasso and Marco Raoul Marini

Image features vs ConvNets

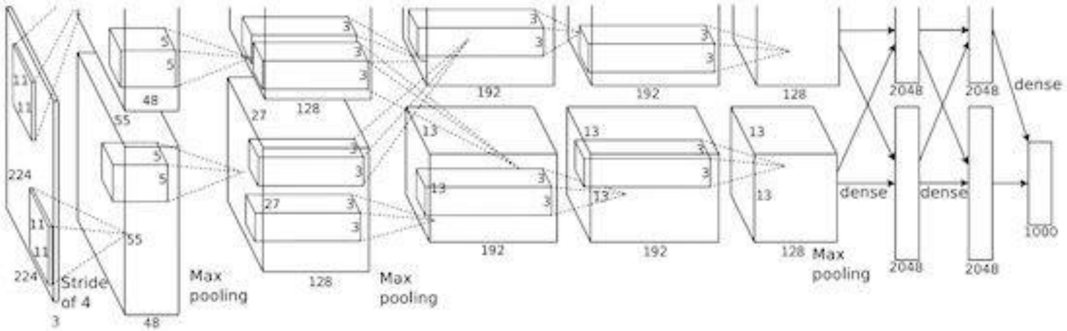
Feature Extraction



f

10 numbers giving scores for classes

training



Krizhevsky, Sutskever, and Hinton, “Imagenet classification with deep convolutional neural networks”, NIPS 2012.

10 numbers giving scores for classes

training

Thank you

Acknowledges: some slides and material from Bernt Schiele, Mario Fritz, Fei-Fei, Justin Johnson, Serena Yeung

Sapienza Università di Roma