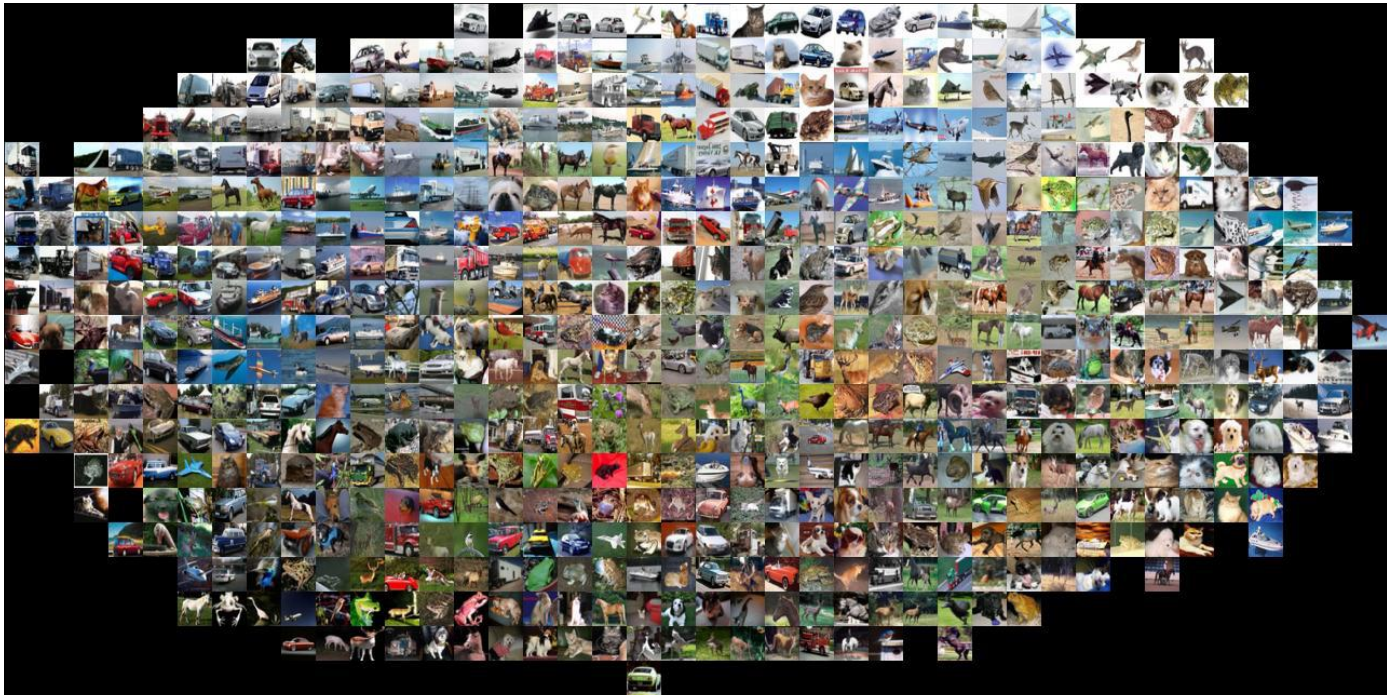
CS5670: Computer Vision

Noah Snavely

Lecture 33: Recognition Basics



Slides from Andrej Karpathy and Fei-Fei Li
http://vision.stanford.edu/teaching/cs231n/

Announcements

- Quiz moved to Tuesday
- Project 4 due tomorrow, April 28, by 11:59pm
- Project 5 to be assigned soon
- Take-home final: Assigned Tuesday May 9 (in class), due May 11 (by 5pm)

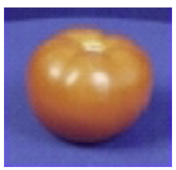
Today

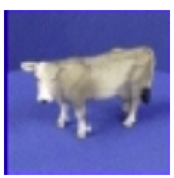
- Image classification pipeline
- Training, validation, testing
- Nearest neighbor classification
- Score function and loss function

- Building up to CNNs for learning
  - Next 2-3 lectures on deep learning

Categorization

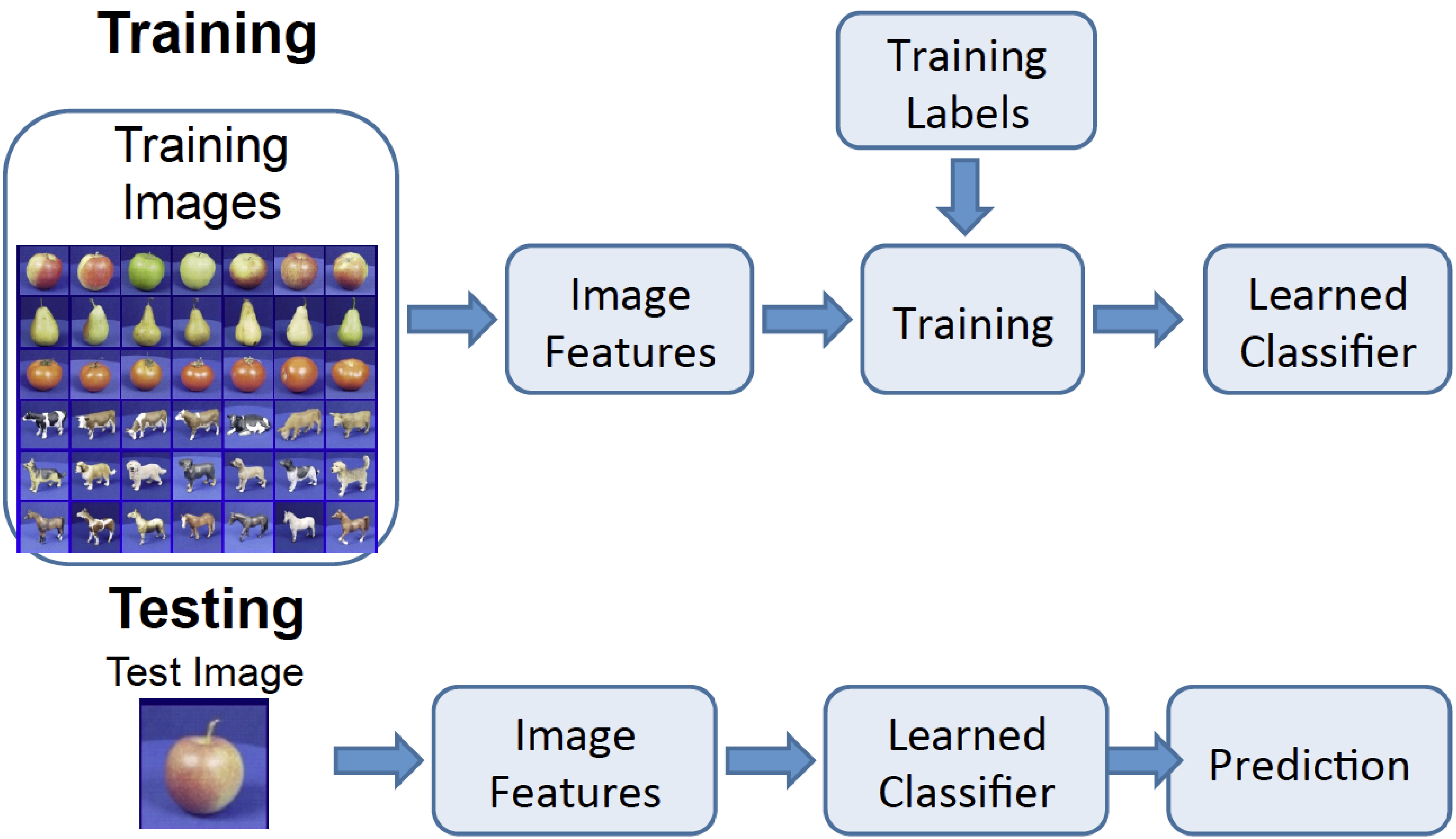
f() = “apple”

f() = “tomato”

f() = “cow”

Dataset: ETH-80, by B. Leibe

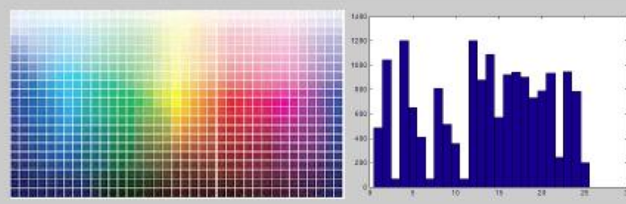
Slide credit: L. Lazebnik



Dataset: ETH-80, by B. Leibe Slide credit: D. Hoiem, L. Lazebnik

Input image

Color: Quantize RGB values

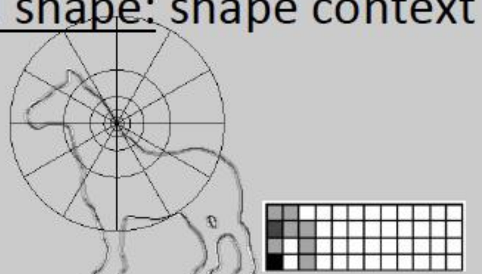


Invariance?
- Translation
- Scale
- Rotation
- Occlusion

Global shape: PCA space

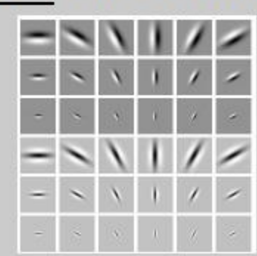
Invariance?
- Translation
- ? Scale
- Rotation
- Occlusion

Local shape: shape context



Invariance?
- Translation
- Scale
- ? Rotation (in-planar)
- Occlusion

Texture: Filter banks



Invariance?
- Translation
- ? Scale
- ? Rotation (in-planar)
- Occlusion

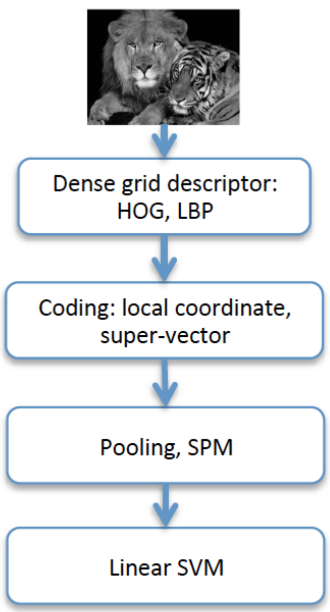
Results

	Color	$D_x D_y$	Mag-Lap	PCA Masks	PCA Gray	Cont. Greedy	Cont. DynProg	Avg.
apple	57.56%	**85.37%**	80.24%	78.78%	**88.29%**	77.07%	76.34%	77.66%
pear	66.10%	90.00%	85.37%	**99.51%**	**99.76%**	90.73%	91.71%	89.03%
tomato	**98.54%**	94.63%	**97.07%**	67.80%	76.59%	70.73%	70.24%	82.23%
cow	86.59%	82.68%	**94.39%**	75.12%	62.44%	86.83%	86.34%	82.06%
dog	34.63%	62.44%	74.39%	72.20%	66.34%	**81.95%**	**82.93%**	67.84%
horse	32.68%	58.78%	70.98%	77.80%	77.32%	**84.63%**	**84.63%**	69.55%
cup	79.76%	66.10%	77.80%	**96.10%**	**96.10%**	**99.76%**	**99.02%**	87.81%
car	62.93%	**98.29%**	77.56%	**100.0%**	**97.07%**	**99.51%**	**100.0%**	90.77%
total	64.85%	79.79%	82.23%	83.41%	82.99%	86.40%	86.40%	80.87%

IMAGENET Large Scale Visual Recognition Challenge

Year 2010

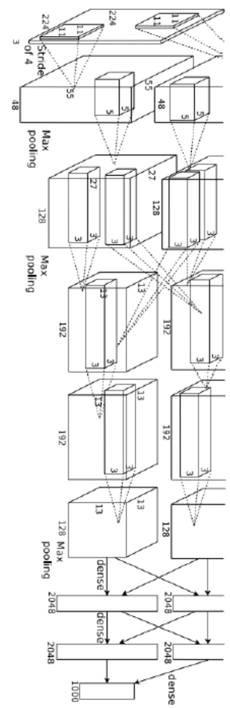
NEC-UIUC



[Lin CVPR 2011]

Year 2012

SuperVision



[Krizhevsky NIPS 2012]

Year 2014

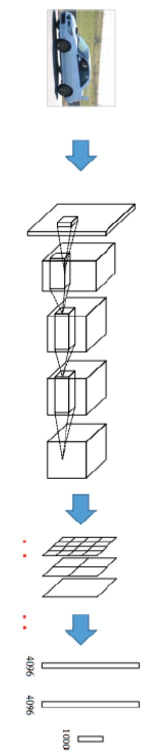
GoogLeNet

[Szegedy arxiv 2014]

VGG

[Simonyan arxiv 2014]

MSRA



[He arxiv 2014]

Deep Learning or CNNs

- Since 2012, huge impact…, best results
- Can soak up all the data for better prediction

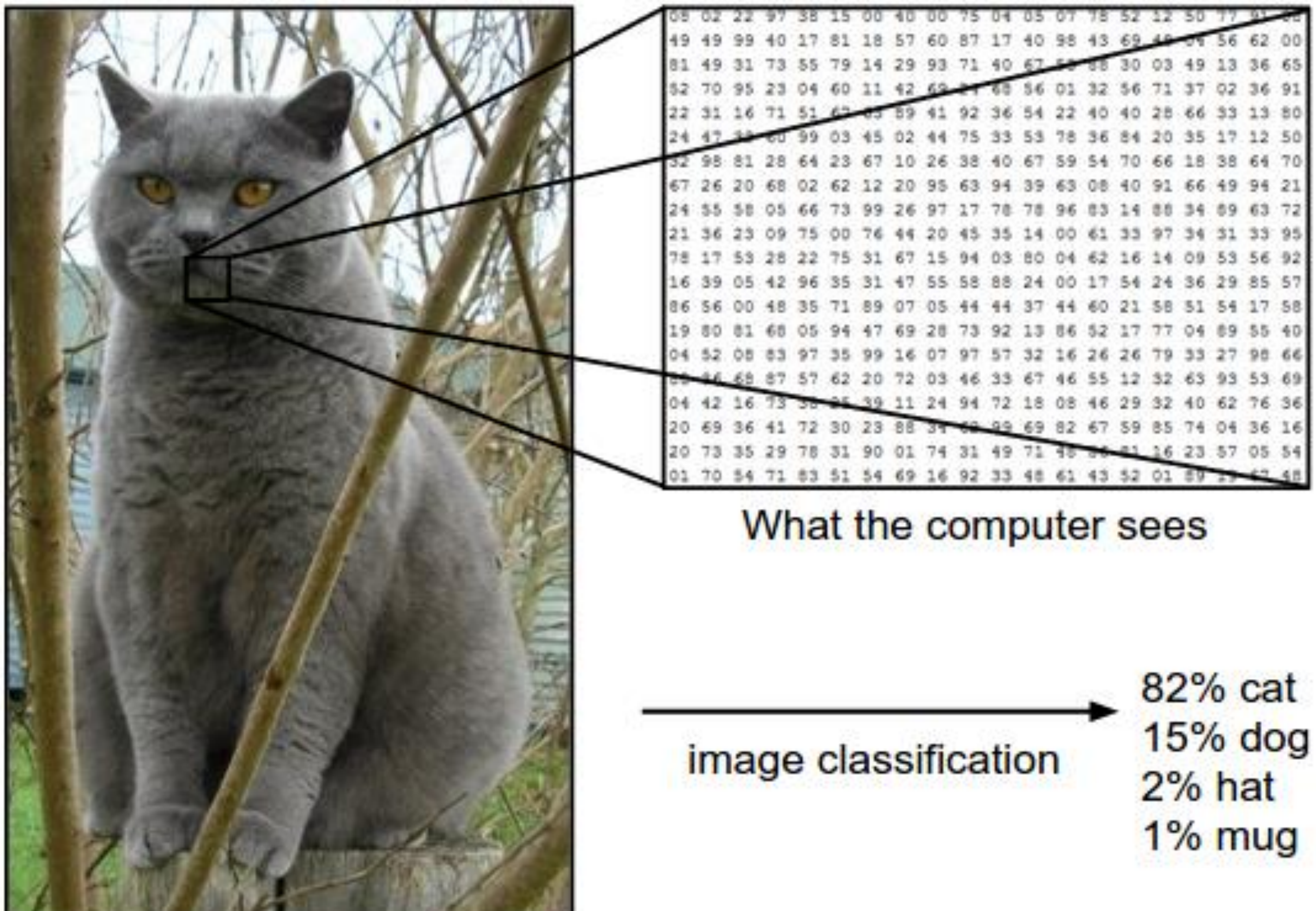
Image Classification

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

→ cat

Slides from Andrej Karpathy and Fei-Fei Li
http://vision.stanford.edu/teaching/cs231n/

Image Classification: Problem



Data-driven approach

- Collect a database of images with labels
- Use ML to train an image classifier
- Evaluate the classifier on test images

Example training set

Slides from Andrej Karpathy and Fei-Fei Li
http://vision.stanford.edu/teaching/cs231n/

Data-driven approach

- Collect a database of images with labels
- Use ML to train an image classifier
- Evaluate the classifier on test images

```
def train(train_images, train_labels):
    # build a model of images -> labels

def predict(image):
    # evaluate the model on the image
    return class_label
```

Slides from Andrej Karpathy and Fei-Fei Li
http://vision.stanford.edu/teaching/cs231n/

Train and Test

- Split dataset between training images and test images
- Be careful about inflation of results

Classifiers

- Nearest Neighbor
- kNN

- SVM
- …

First: Nearest Neighbor (NN) Classifier

- Train
  - Remember all training images and their labels
- Predict
  - Find the closest (most similar) training image
  - Predict its label as the true label

How to find the most similar training image? What is the distance metric?

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

Where I_1 denotes image 1, and p denotes each pixel

$$\left| \overset{\text{test image}}{\begin{bmatrix} 56 & 32 & 10 & 18 \\ 90 & 23 & 128 & 133 \\ 24 & 26 & 178 & 200 \\ 2 & 0 & 255 & 220 \end{bmatrix}} - \overset{\text{training image}}{\begin{bmatrix} 10 & 20 & 24 & 17 \\ 8 & 10 & 89 & 100 \\ 12 & 16 & 178 & 170 \\ 4 & 32 & 233 & 112 \end{bmatrix}} \right| = \overset{\text{pixel-wise absolute value differences}}{\begin{bmatrix} 46 & 12 & 14 & 1 \\ 82 & 13 & 39 & 33 \\ 12 & 10 & 0 & 30 \\ 2 & 32 & 22 & 108 \end{bmatrix}} \rightarrow 456$$

Slides from Andrej Karpathy and Fei-Fei Li
http://vision.stanford.edu/teaching/cs231n/

Choice of distance metric

- Hyperparameter

L1 (Manhattan) distance

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

L2 (Euclidean) distance

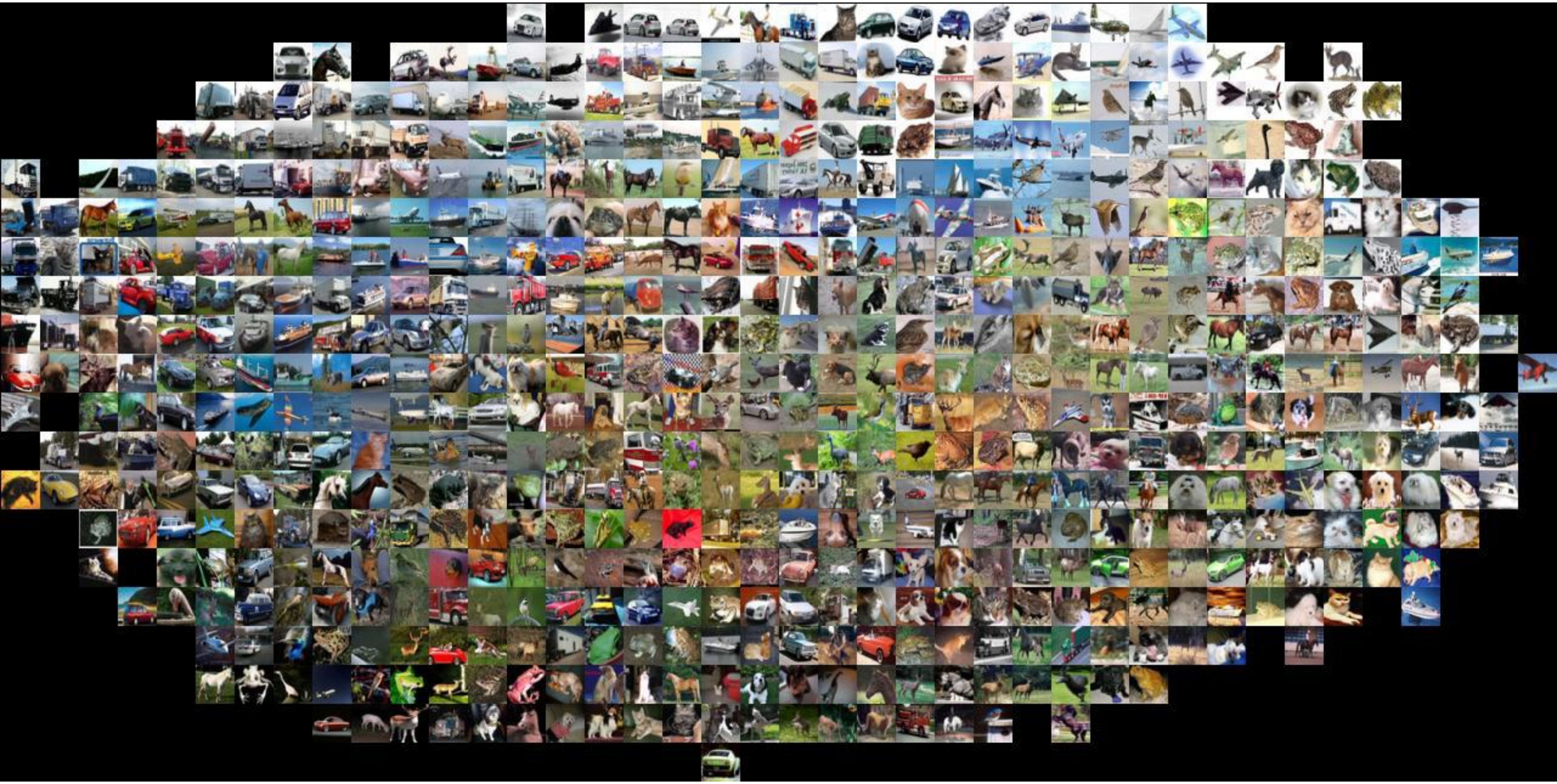
$$d_2(I_1, I_2) = \sqrt{\sum_p \left(I_1^p - I_2^p\right)^2}$$

- Two most commonly used special cases of p-norm

$$||x||_p = \left(|x_1|^p + \cdots + |x_n|^p\right)^{\frac{1}{p}} \qquad p \geq 1, x \in \mathbb{R}^n$$

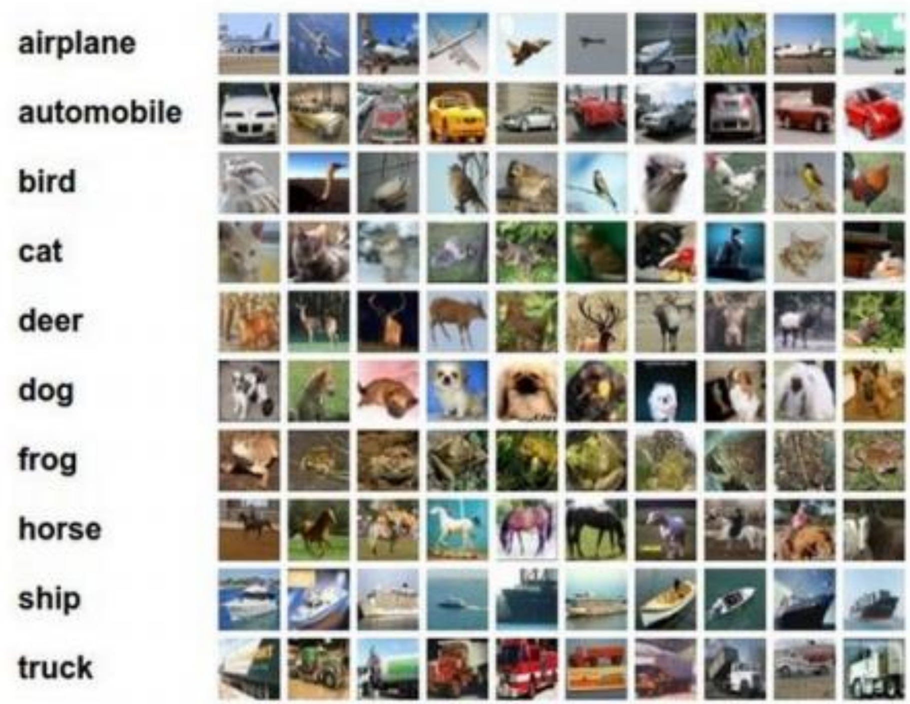
Slides from Andrej Karpathy and Fei-Fei Li
http://vision.stanford.edu/teaching/cs231n/

Visualization: L2 distance



CIFAR-10 and NN results

Example dataset: **CIFAR-10**
10 labels
50,000 training images, each image is tiny: 32x32
10,000 test images.



CIFAR-10 and NN results

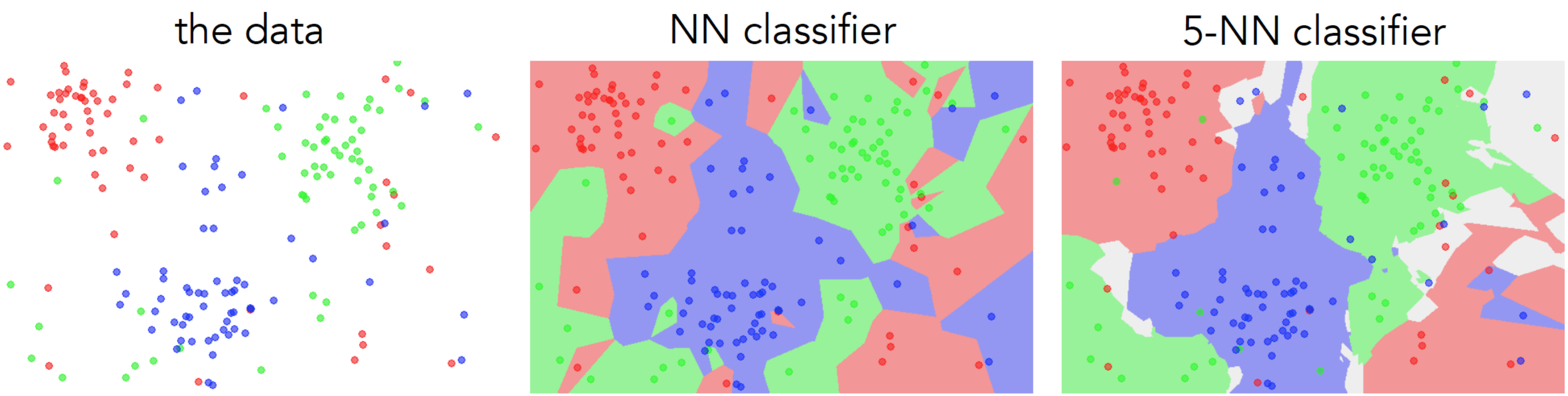
Example dataset: **CIFAR-10**
10 labels
50,000 training images
10,000 test images.

For every test image (first column), examples of nearest neighbors in rows

Slides from Andrej Karpathy and Fei-Fei Li
http://vision.stanford.edu/teaching/cs231n/

k-nearest neighbor

- Find the k closest points from training data
- Labels of the k points "vote" to classify



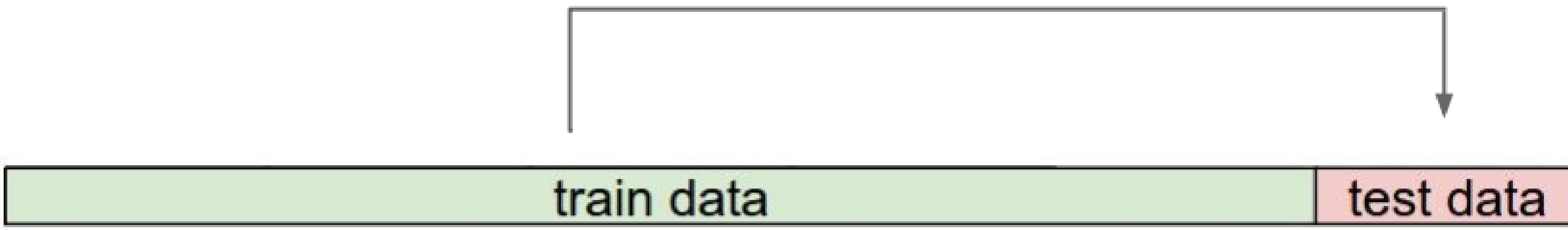
Hyperparameters

- What is the best distance to use?
- What is the best value of k to use?

- i.e., how do we set the hyperparameters?

- Very problem-dependent
- Must try them all and see what works best

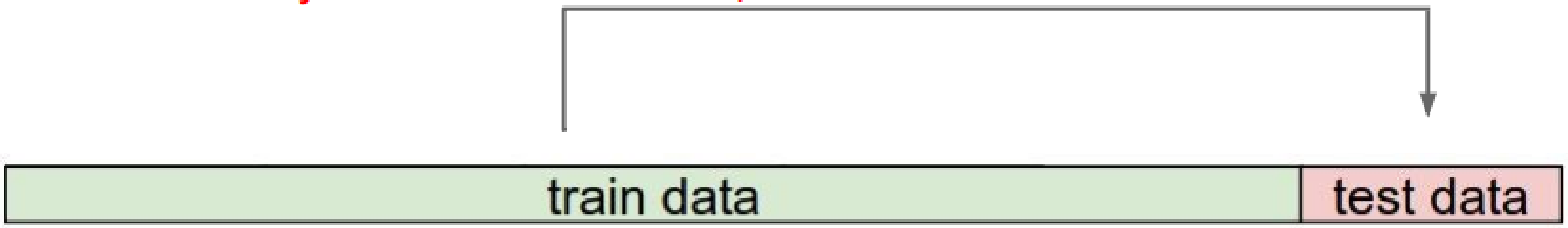
Try out what hyperparameters work best on test set.



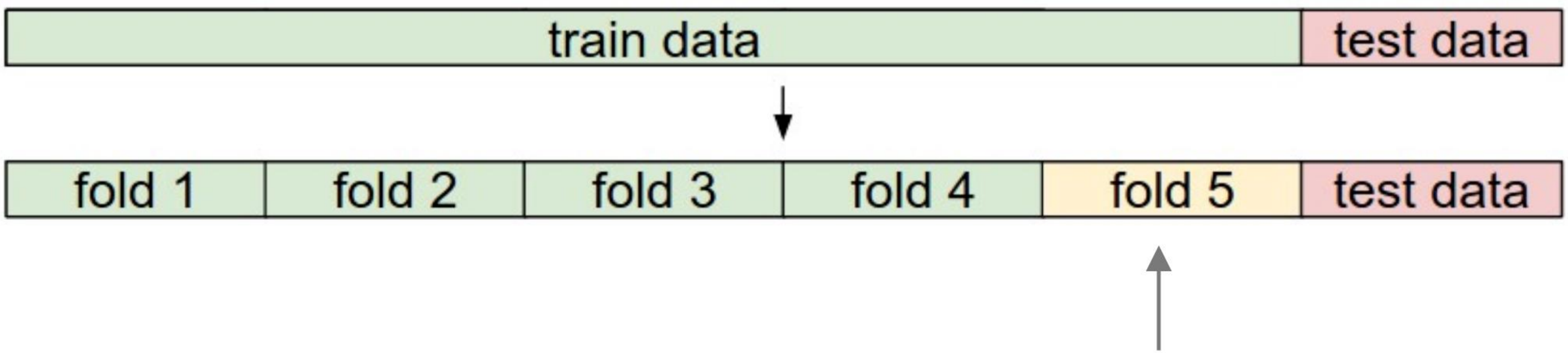
Trying out what hyperparameters work best on test set:

Very bad idea. The test set is a proxy for the generalization performance!

Use only **VERY SPARINGLY,** at the end.



Validation



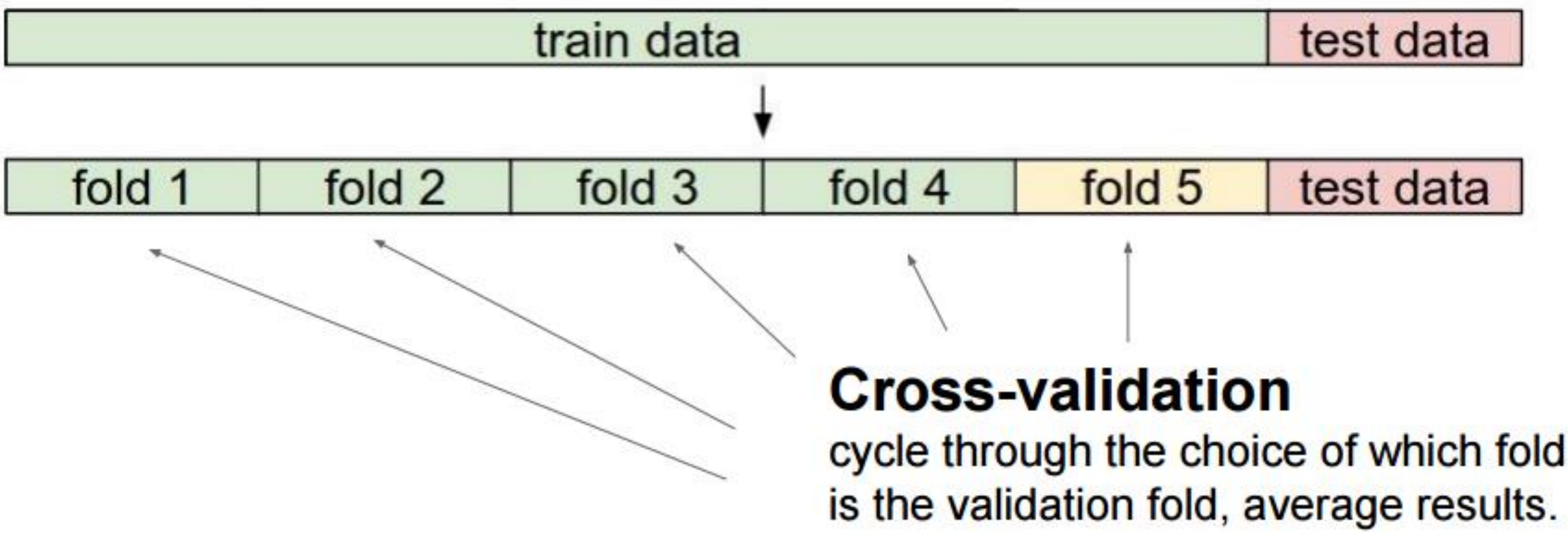
Validation data

use to tune hyperparameters

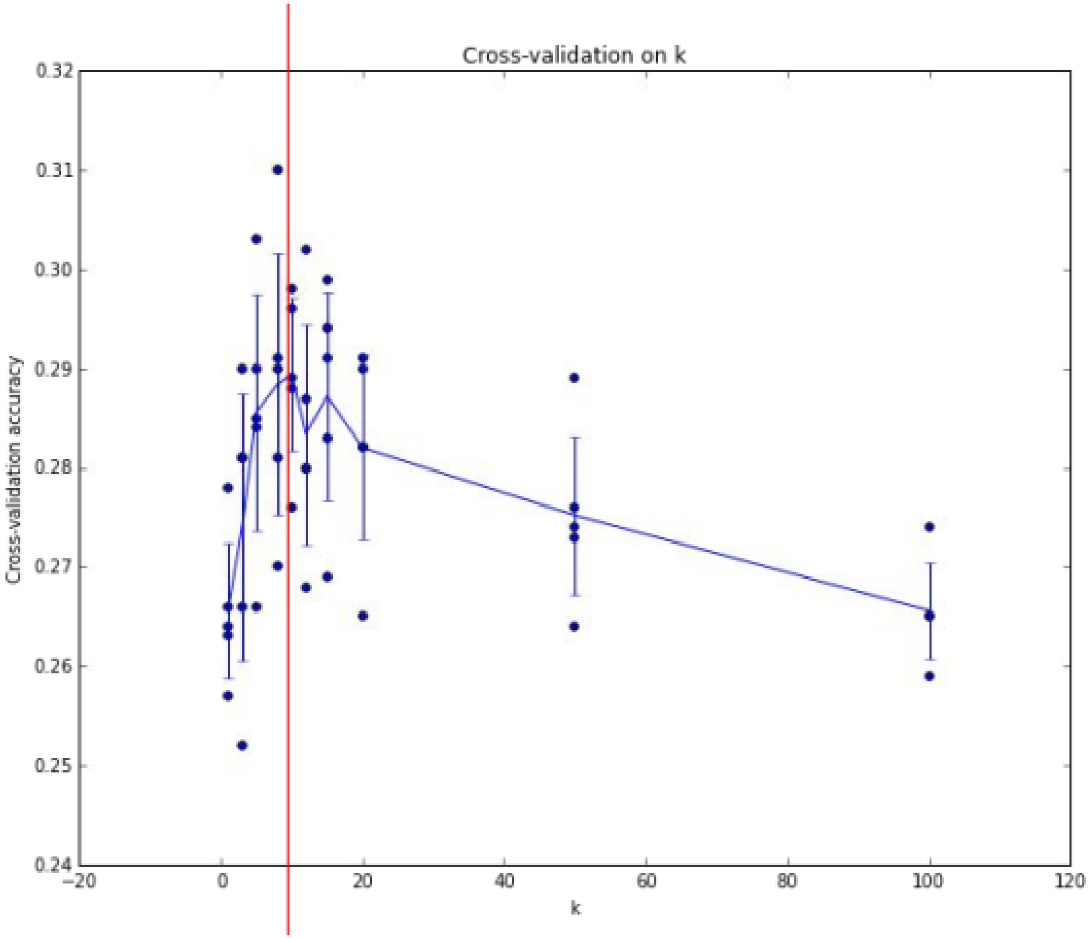
evaluate on test set ONCE at the end

Slides from Andrej Karpathy and Fei-Fei Li
http://vision.stanford.edu/teaching/cs231n/

Cross-validation



Slides from Andrej Karpathy and Fei-Fei Li
http://vision.stanford.edu/teaching/cs231n/



Example of
5-fold cross-validation
for the value of **k.**

Each point: single
outcome.

The line goes
through the mean, bars
indicated standard
deviation

(Seems that k ~= 7 works best
for this data)

How to pick hyperparameters?

- Methodology
  - Train and test
  - Train, validate, test

- Train for original model
- Validate to find hyperparameters
- Test to understand generalizability

kNN -- Complexity and Storage

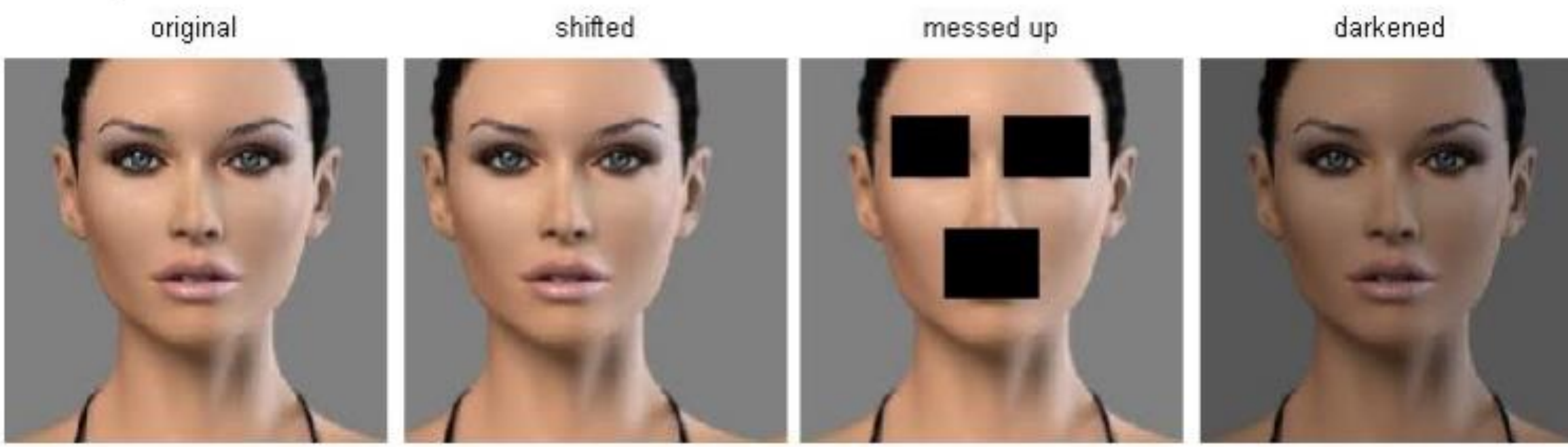
- N training images, M test images

- Training: O(1)
- Testing: O(MN)

- Hmm…
  - Normally need the opposite
  - Slow training (ok), fast testing (necessary)

k-Nearest Neighbor on images **never used.**

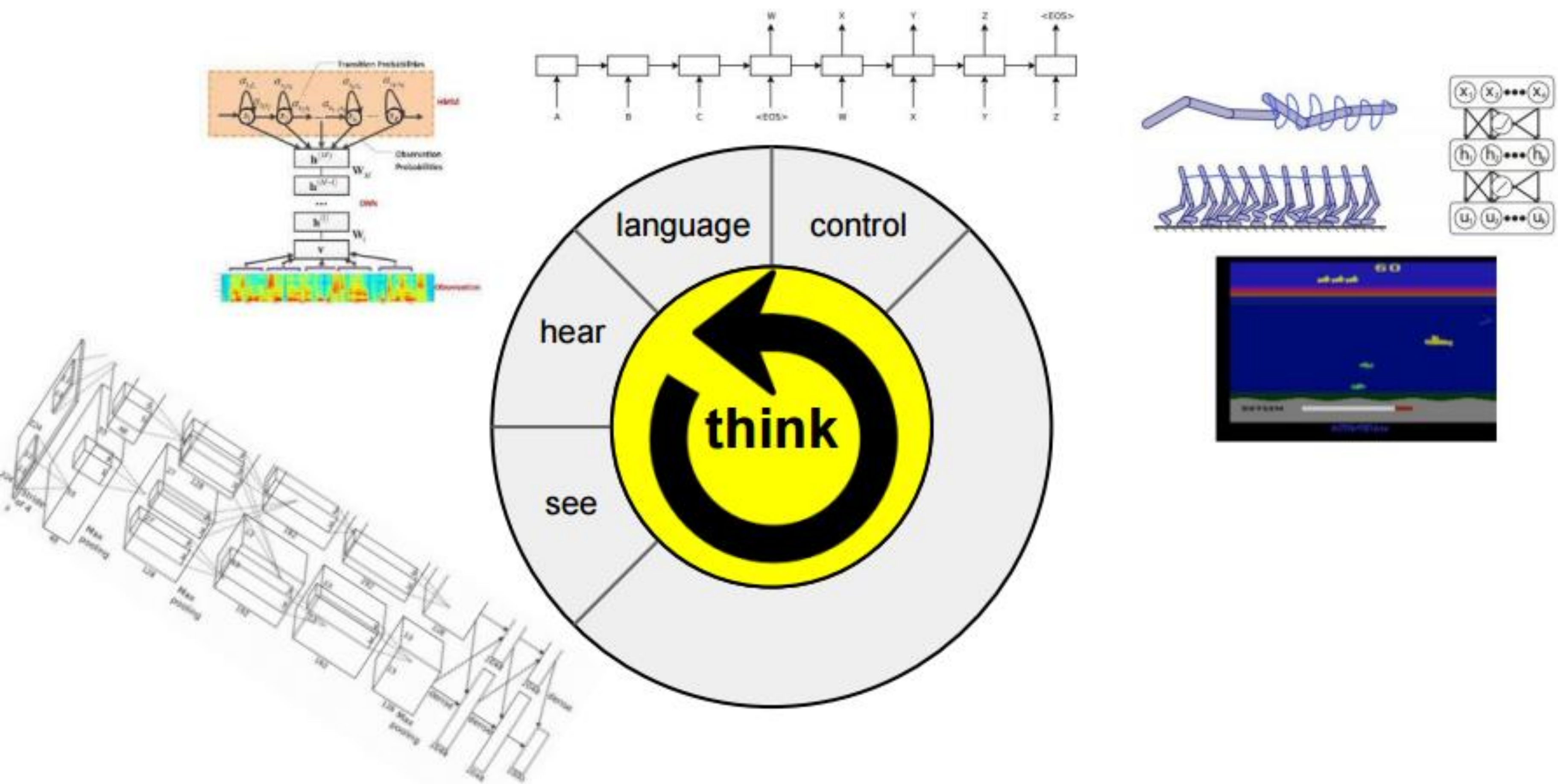
- terrible performance at test time
- distance metrics on level of whole images can be very unintuitive



(all 3 images have same L2 distance to the one on the left)

Instead

- Image classification using linear classifiers



Instead

- Image classification using linear classifiers

- Need
  - Score function: raw data to class scores
  - Loss function: agreement between predicted scores and ground truth labels

Score function

class scores

Slides from Andrej Karpathy and Fei-Fei Li
http://vision.stanford.edu/teaching/cs231n/

Score function: f

Parametric approach

image parameters

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

10 numbers, indicating class scores

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

Parametric approach: Linear classifier

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

3072x1

10x1

10x3072

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

10 numbers, indicating class scores

parameters, or “weights”

Parametric approach: Linear classifier

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

3072x1

(+b) 10x1

10x1

10x3072

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

10 numbers,
indicating class
scores

parameters, or “weights”

Linear Classifier

define a **score function**

data (image)

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

class scores

"weights"

"bias vector"

"parameters"

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

stretch pixels into single column

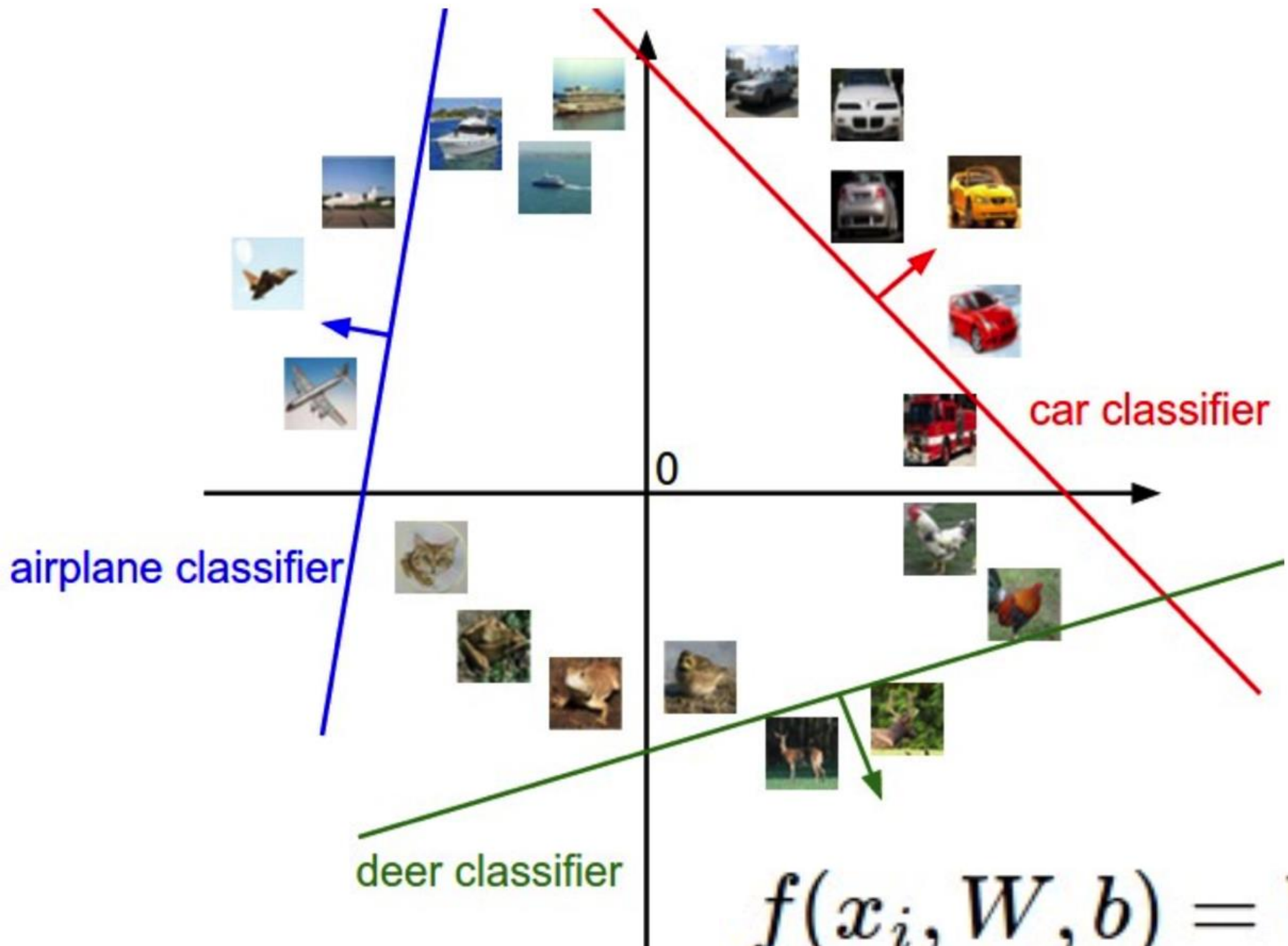
input image

$$\underbrace{\begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0 & 0.25 & 0.2 & -0.3 \end{bmatrix}}_{W} \underbrace{\begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix}}_{x_i} + \underbrace{\begin{bmatrix} 1.1 \\ 3.2 \\ -1.2 \end{bmatrix}}_{b} \longrightarrow \begin{array}{cl} \underbrace{\begin{bmatrix} -96.8 \\ 437.9 \\ 61.95 \end{bmatrix}}_{f(x_i;W,b)} & \begin{array}{l} \text{cat score} \\ \text{dog score} \\ \text{ship score} \end{array} \end{array}$$

Interpretation: Template matching

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

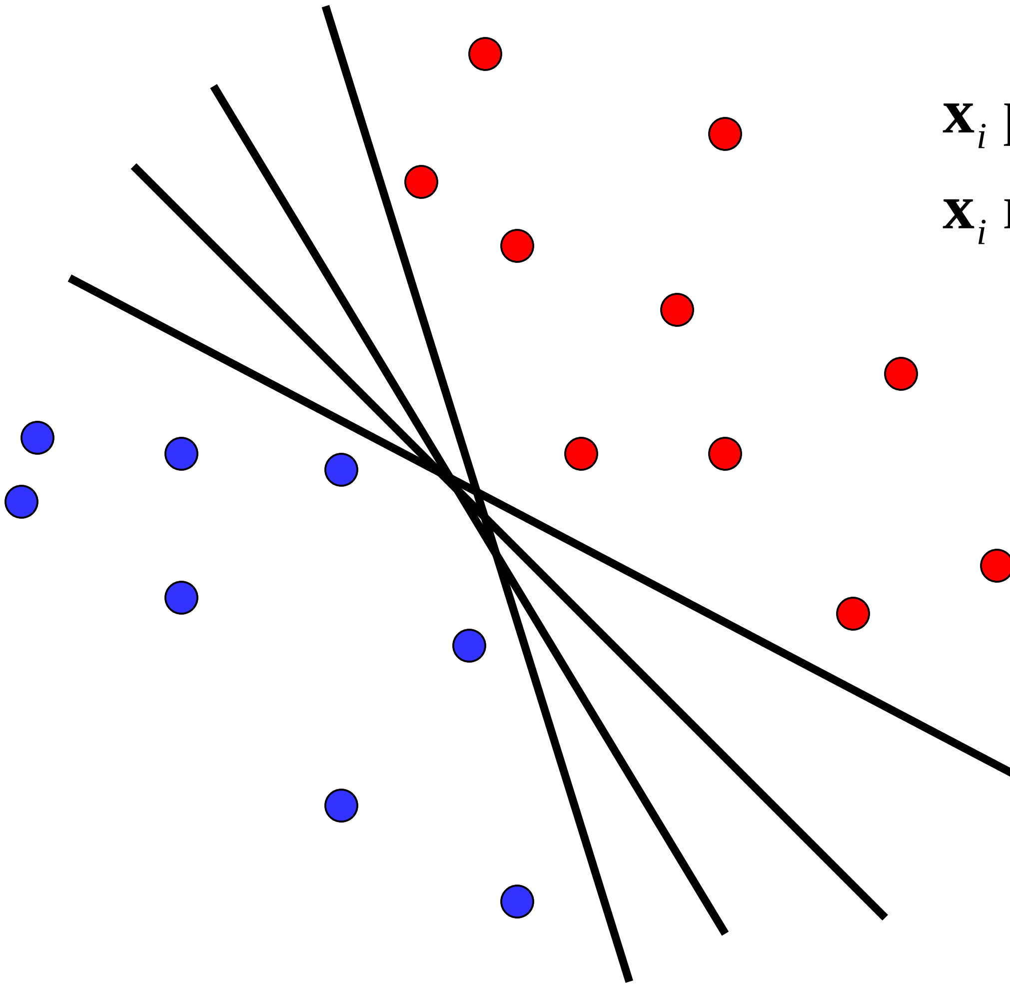
Geometric Interpretation



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

Linear classifiers

- Find linear function (*hyperplane*) to separate positive and negative examples



$\mathbf{x}_i$ positive : $\quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 0$

$\mathbf{x}_i$ negative : $\quad \mathbf{x}_i \cdot \mathbf{w} + b < 0$

Which hyperplane
is best?

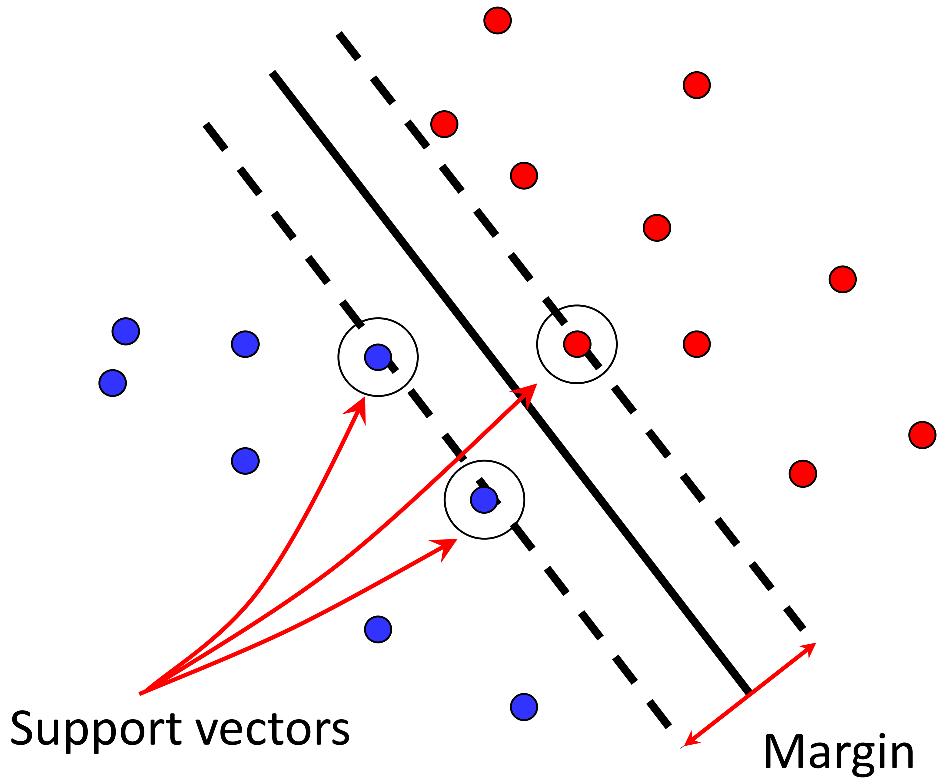
Support vector machines

- Find hyperplane that maximizes the *margin* between the positive and negative examples

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Support vector machines

- Find hyperplane that maximizes the *margin* between the positive and negative examples



$\mathbf{x}_i$ positive $(y_i = 1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$

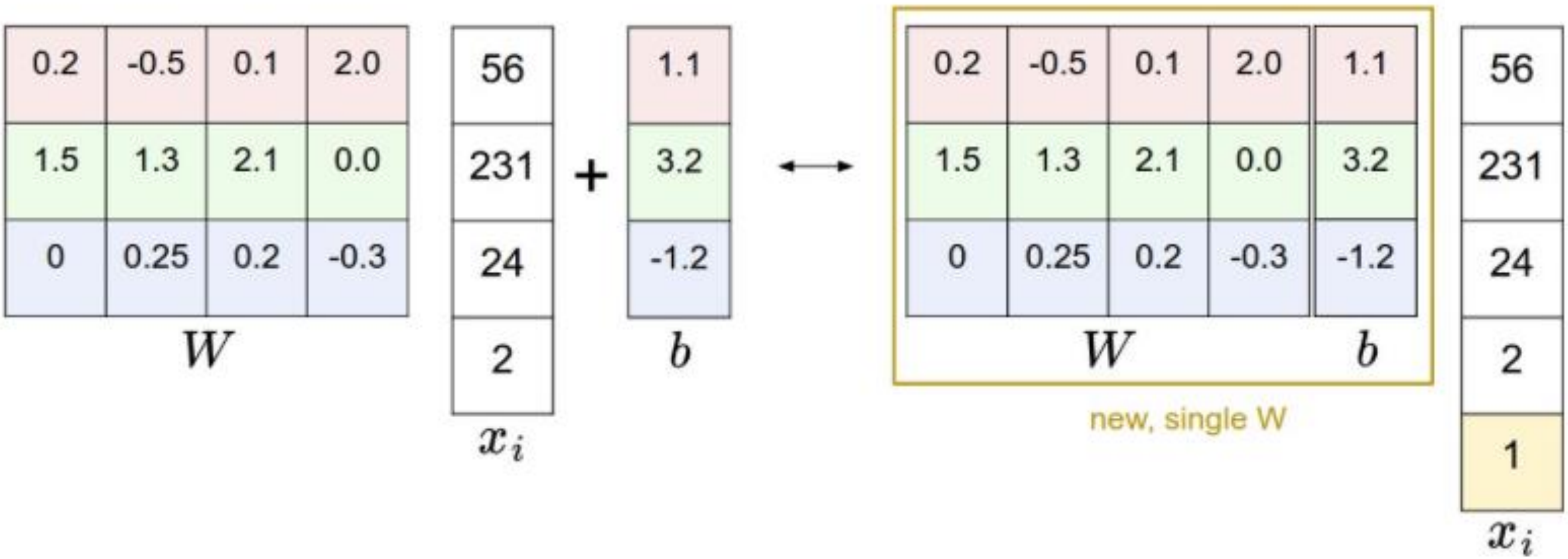
$\mathbf{x}_i$ negative $(y_i = -1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

For support, vectors, $\quad \mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

Distance between point and hyperplane: $\quad \dfrac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{\|\mathbf{w}\|}$

Therefore, the margin is $2 / \|\mathbf{w}\|$

Bias Trick



0.2
-0.5
0.1
2.0
1.5
1.3
2.1
0.0
0
0.25
0.2
-0.3
W
56
231
24
2
x_i
+
1.1
3.2
-1.2
b
0.2
-0.5
0.1
2.0
1.1
1.5
1.3
2.1
0.0
3.2
0
0.25
0.2
-0.3
-1.2
W
b
new, single W
56
231
24
2
1
x_i

Summary

- Data-driven: Train, validate, test
  - Need labeled data
- Classifier
  - Nearest neighbor, kNN

Loss function, cost/objective function

- Given ground truth labels (yi), scores f (xi, W)
  - how unhappy are you with the scores?
- Loss function or objective/cost function
- Want to minimize the loss function

Loss function, cost/objective function

- Given ground truth labels (yi) and scores f(xi, W)
  - how unhappy are you with the scores?

$$f(x_i, W) = [\boxed{13}, -7, 11]$$

$$y_i = 0$$

Intuition

delta
score
scores for other classes
score for correct class

Loss fn: Multi-class SVM loss

- Given ground truth labels (yi) and scores f(xi, W)
  - how unhappy are you with the scores?

$$L_i = \sum_{j \neq y_i} \max(0, f(x_i, W)_j - f(x_i, W)_{y_i} + \Delta)$$

(One possible generalization of Binary Support Vector Machine to multiple classes)

$$L_i = C \max(0, 1 - y_i w^T x_i) + R(W)$$

Loss fn: interpretation

$$\boxed{L_i} = \boxed{\sum_{j \neq y_i}} \max(0, \boxed{f(x_i, W)_j - f(x_i, W)_{y_i}} + \Delta)$$

loss due to example i

sum over all incorrect labels

difference between the correct class score and incorrect class score

Example: loss

Example: $$L_i = \sum_{j \neq y_i} \max(0, f(x_i, W)_j - f(x_i, W)_{y_i} + \Delta)$$

e.g. 10

$$f(x_i, W) = [13, -7, 11]$$

$$y_i = 0$$

---

loss = ?

Example: loss

Example: $$L_i = \sum_{j \neq y_i} \max(0, f(x_i, W)_j - f(x_i, W)_{y_i} + \Delta)$$

e.g. 10

$$f(x_i, W) = [13, -7, 11]$$

$$y_i = 0$$

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

Need more..

- Regularization
  - Ambiguity: W is not unique
  - If loss is 0, k W also has 0 loss

  - Add a regularization penalty
    - Try to keep the weights low
    - Also, weights can blow up if you don't have it

Important: regularization

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

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

Regularization strength

Determine by cross-validation

Final loss function

$$L = \frac{1}{N} \sum_i \sum_{j \neq y_i} \left[\max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + \Delta) \right] + \lambda R(W)$$

Can set delta to 1

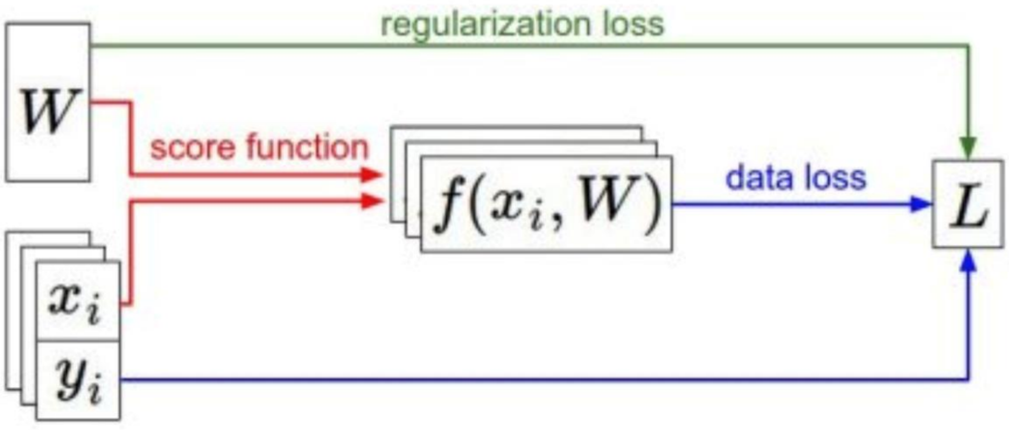
Summary

1. Score function

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

2. Loss function

$$L = \frac{1}{N} \sum_i \sum_{j \neq y_i} \left[\max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + \Delta) \right] + \lambda R(W)$$



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

Summary

- Have score function and loss function
  - Will generalize the score function
- Find W and b to minimize loss
  - Minimize loss using gradient descent
- Now to CNNs