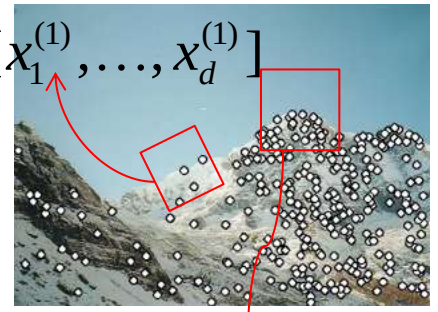
特征提取和匹配

基于特征匹配的识别

1) 检测 Detection: 找到图中的关键点

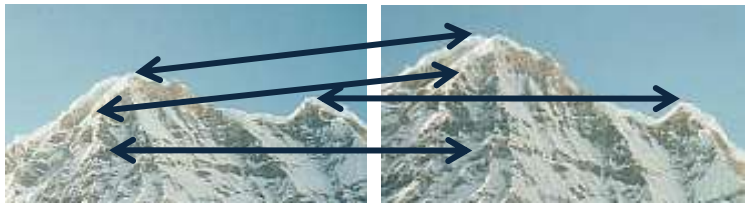
2) **解释 Description: 在关键点周围提取特征**

$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$



$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

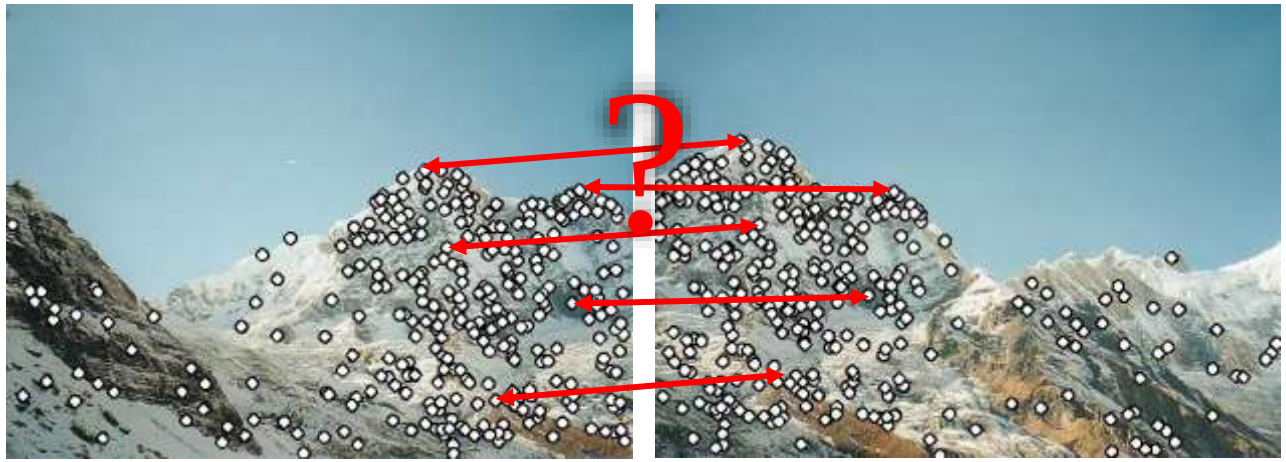
3) 匹配 Matching: 根据两个视角下的特征进行匹配



Kristen Grauman

特征提取器

我们已经知道怎么检测关键点了
下一个问题: **怎么提取关键点信息?**



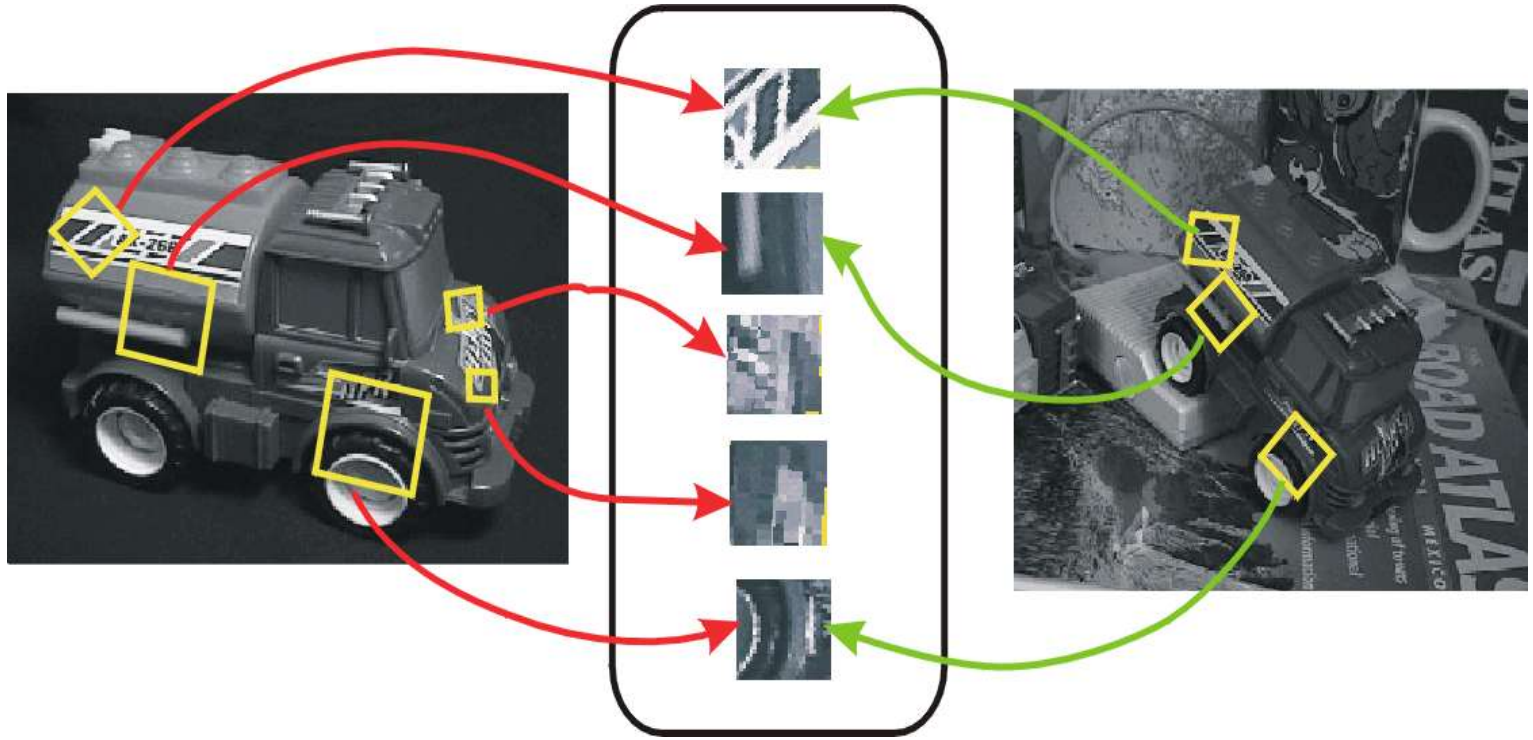
Answer: 使用特征提取器（特征描述子， descriptor）
怎么做?

1. **使用关键点周围的像素**
2. **使用如SIFT等保证不变性的特征提取器**

回顾：不变局部特征

找到在图像变换下仍然保持不变的图像特征

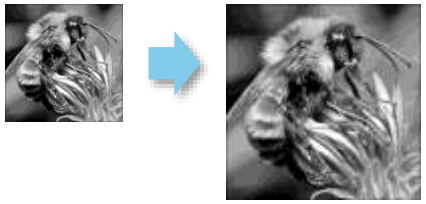
- 几何不变性：平移、旋转、缩放
- 光学不变性：亮度、对比度，…



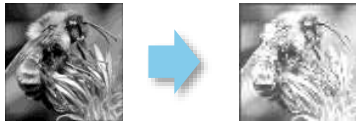
特征提取器

- 几何层面

旋转 

尺度 

- 光学层面

强度变换 

描述特征

Image – 40

1/2 size, rot. 45°
Lightened+40

全图

100x100 crop
at Glasses

特征提取器的旋转不变性

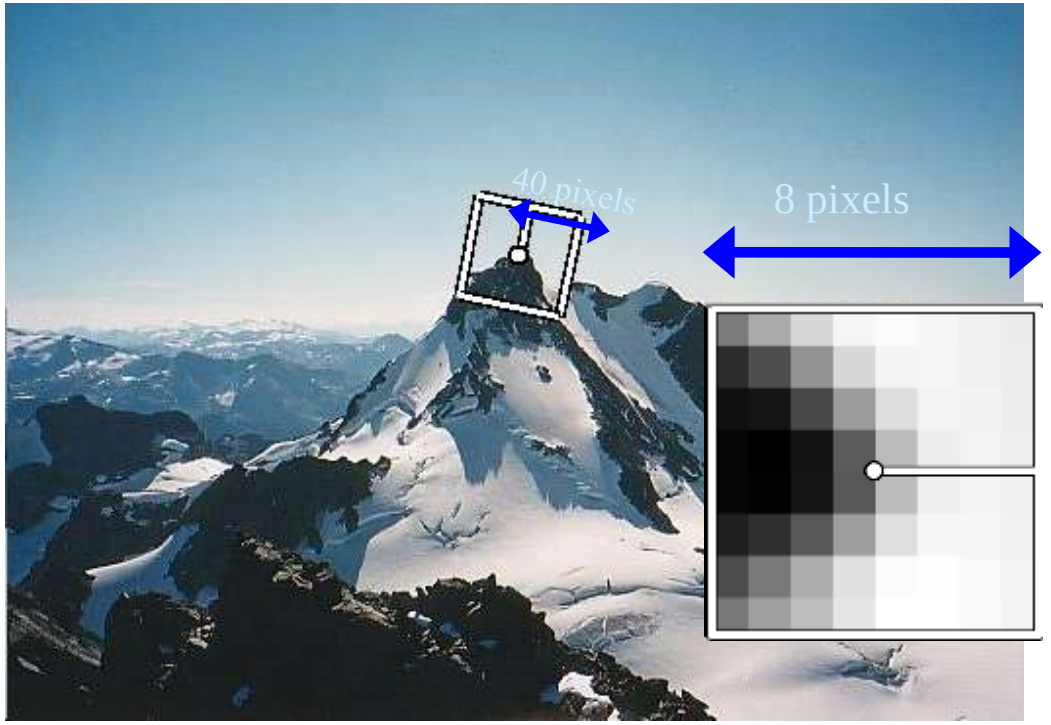
- 找到图像块的主要方向
  - 计算特征值 Eigenvalue
    - (最大的特征值)
  - 计算梯度方向
  - 旋转不同方向并提取特征

Figure by Matthew Brown

Multiscale Oriented PatcheS descriptor

使用 40x40 方形窗口提取特征

- 缩放至1/5
- 旋转至水平
- 将 8x8 的方形窗口作为特征
- 减去均值和方差保证强度不变性



Adapted from slide by Matthew Brown

不同尺度下的特征提取

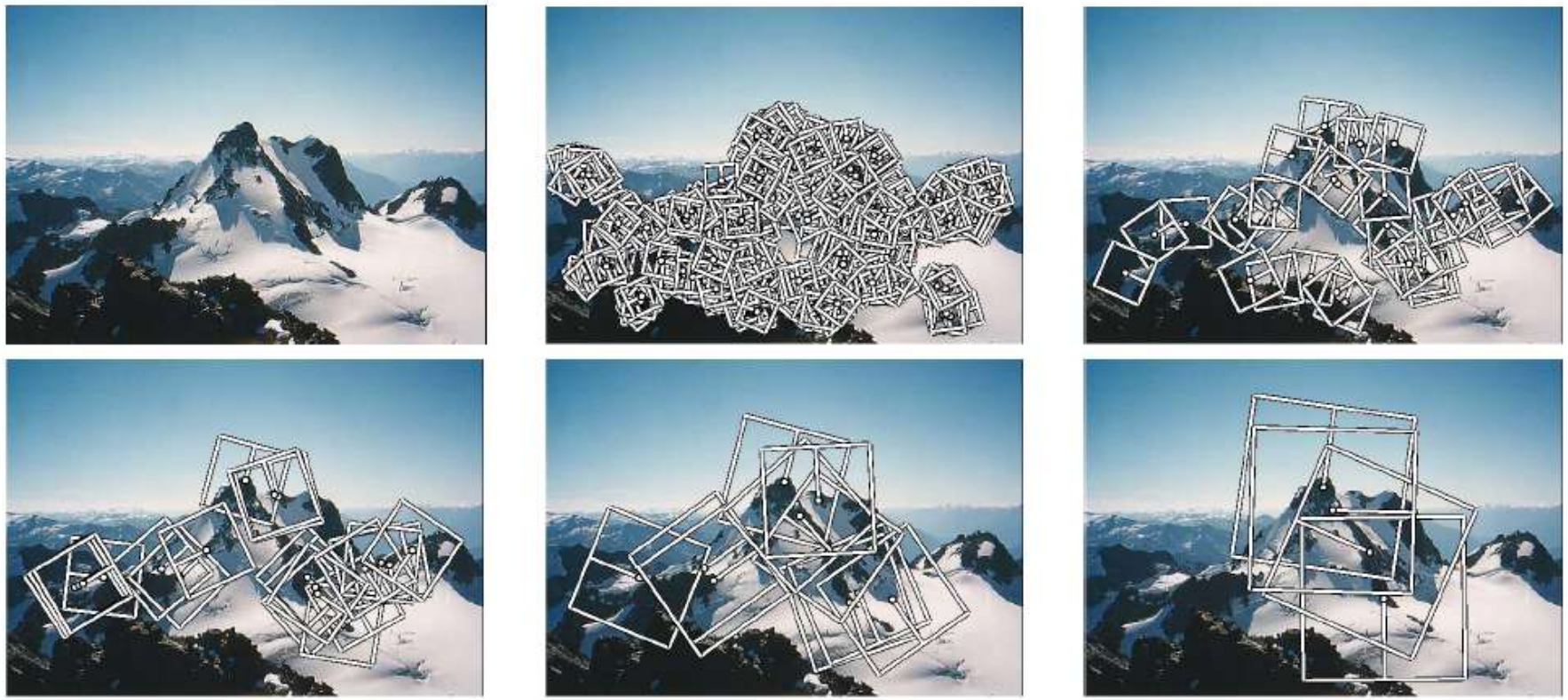
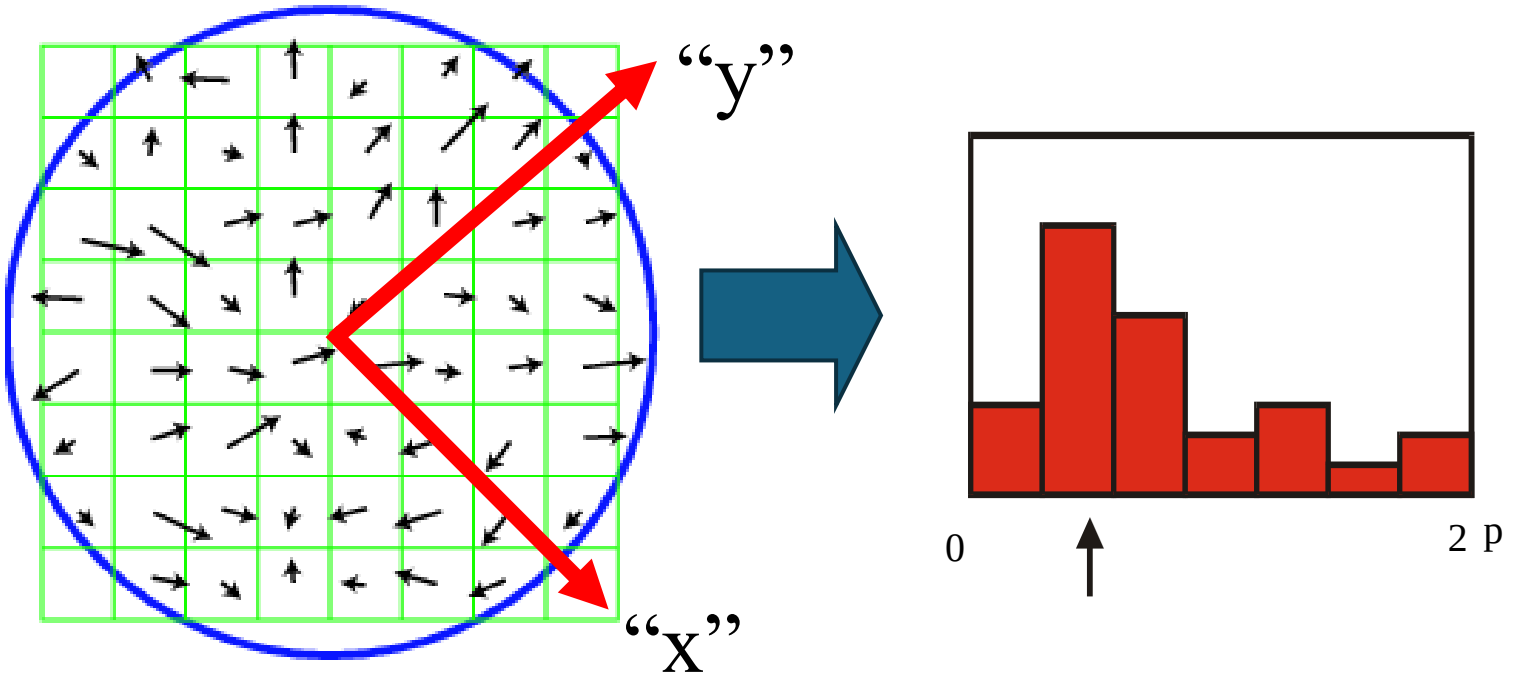


Figure 1. Multi-scale Oriented Patches (MOPS) extracted at five pyramid levels from one of the Matier images. The boxes show the feature orientation and the region from which the descriptor vector is sampled.

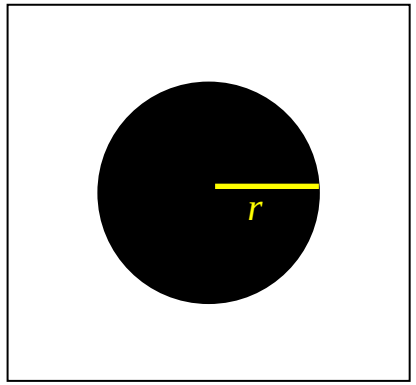
旋转不变：另一种方式

给定窗口，找到像素点梯度方向最多的方向

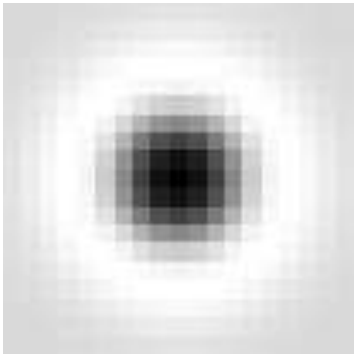


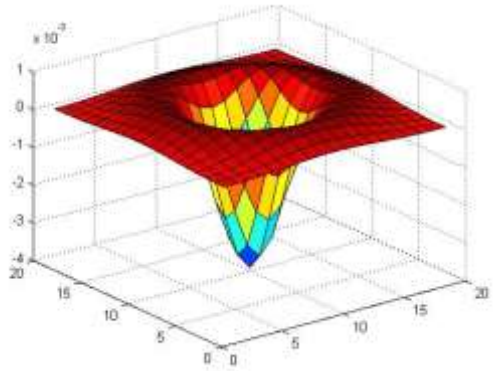
Slide credit: S. Lazebnik

特征尺度



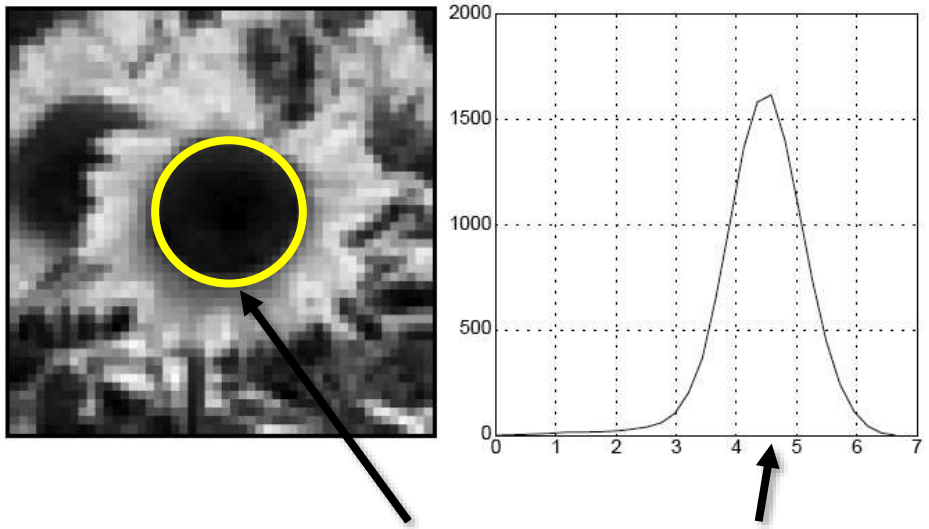
image





Laplacian

与角点相同，计算响应最大的尺度

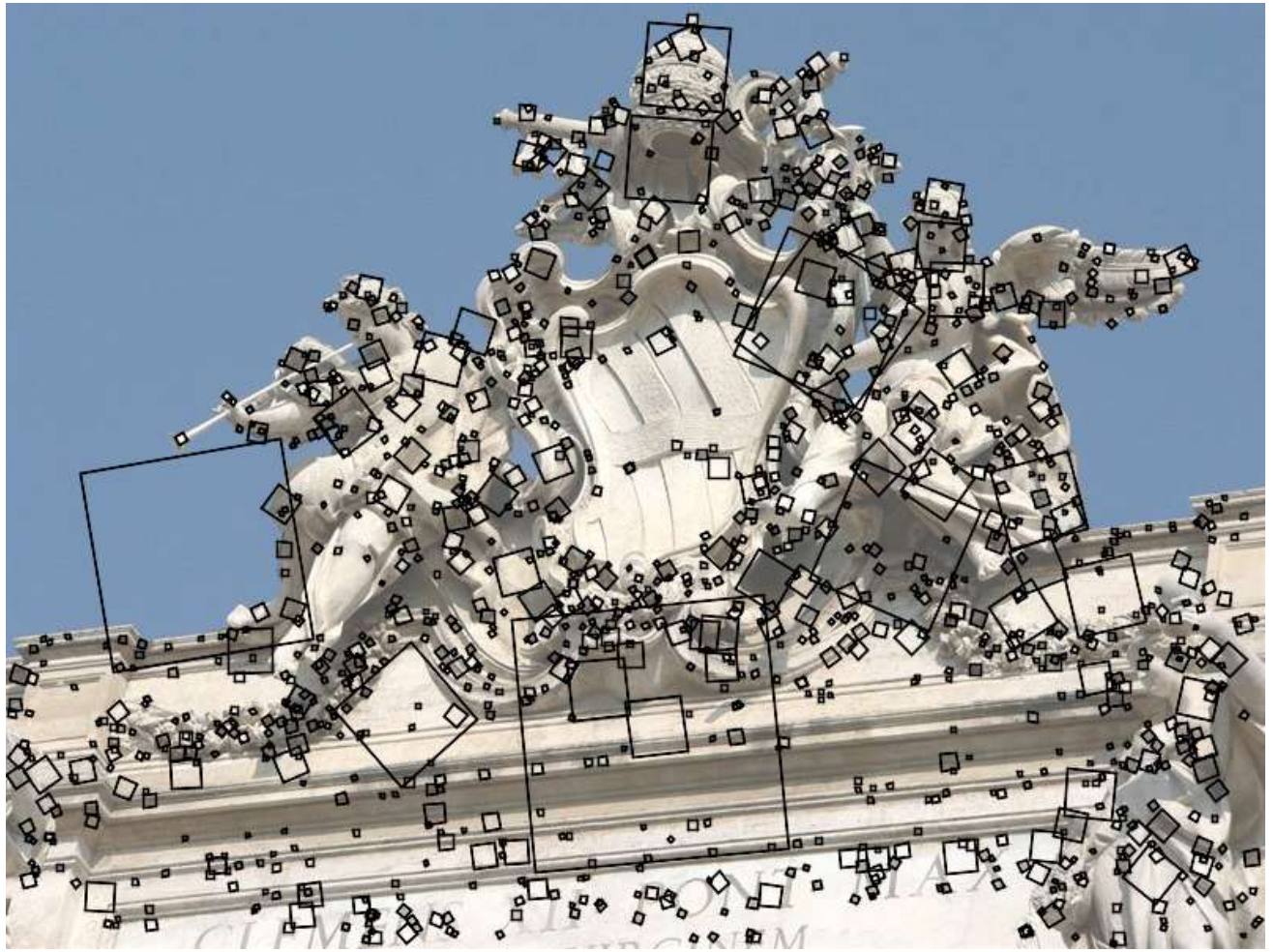


characteristic scale

- 首先找到特征尺度确定窗口大小

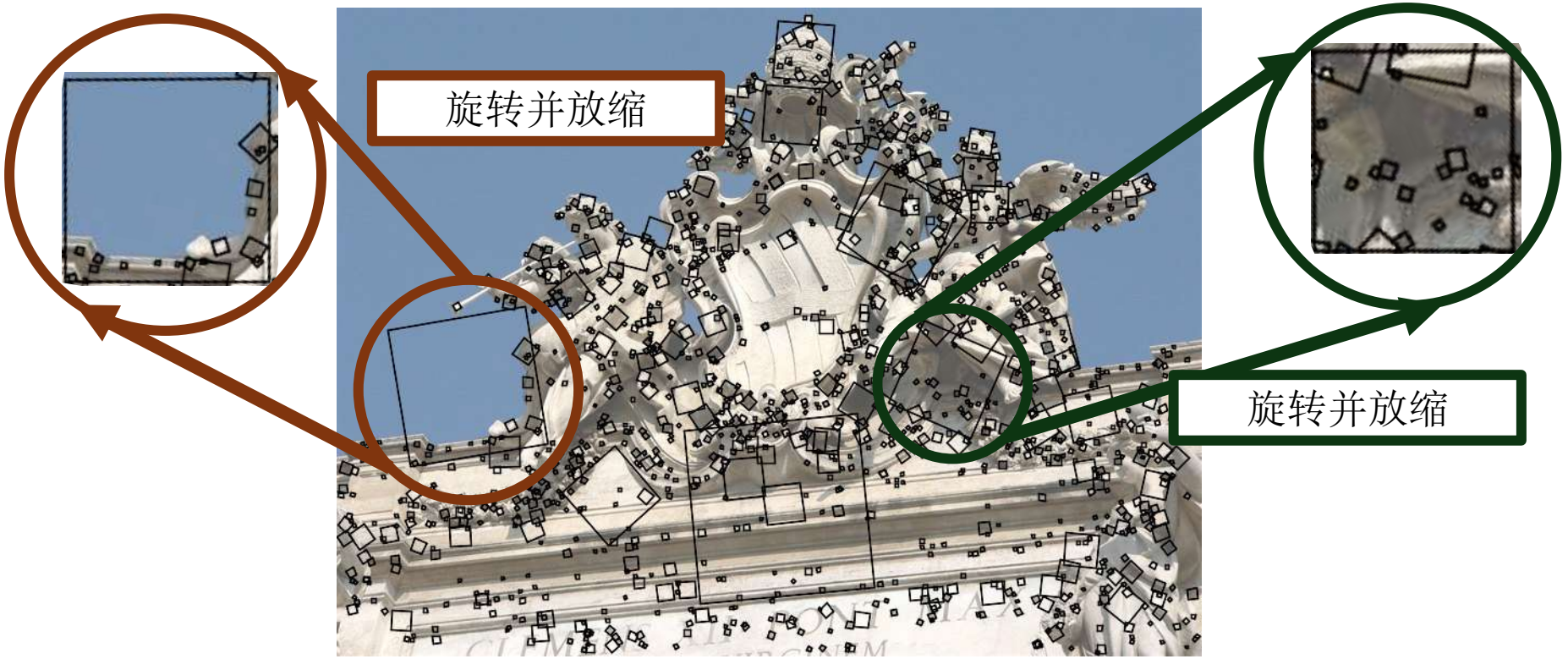
T. Lindeberg (1998). "Feature detection with automatic scale selection." *International Journal of Computer Vision* **30** (2): pp 77--116.

尺度与旋转



Picture credit: S. Lazebnik. Paper: David G. Lowe. **"Distinctive image features from scale-invariant keypoints."** *IJCV* 60 (2), pp. 91-110, 2004.

尺度与旋转



Picture credit: S. Lazebnik. Paper: David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* 60 (2), pp. 91-110, 2004.

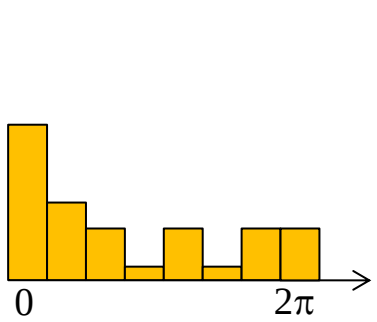
Scale Invariant Feature Transform

想法:

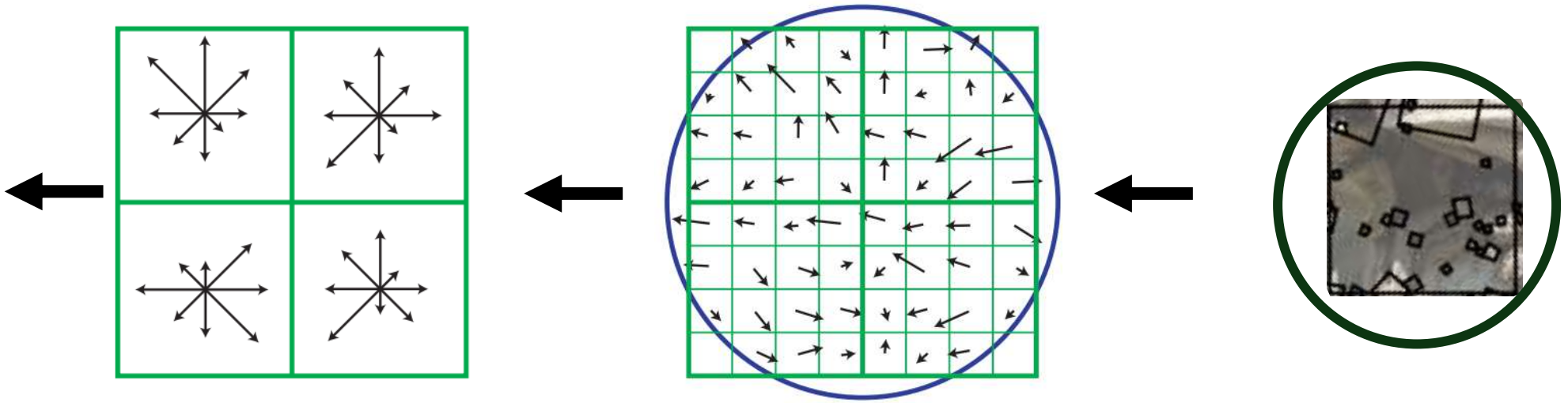
- 在检测到的特征周围取16x16的方形窗口
- 为每个像素计算边缘方向（梯度的角度 - 90度）
  - 减少亮度影响
- 排除弱边缘（梯度幅值低于阈值）
- 为剩余边缘方向创建直方图

做法：

- 把16x16的窗口分成4x4 的格点(在这里画的是2x2)
- 计算每个点的方向直方图
- 16 个格点 * 8 个方向 = 128 维的特征



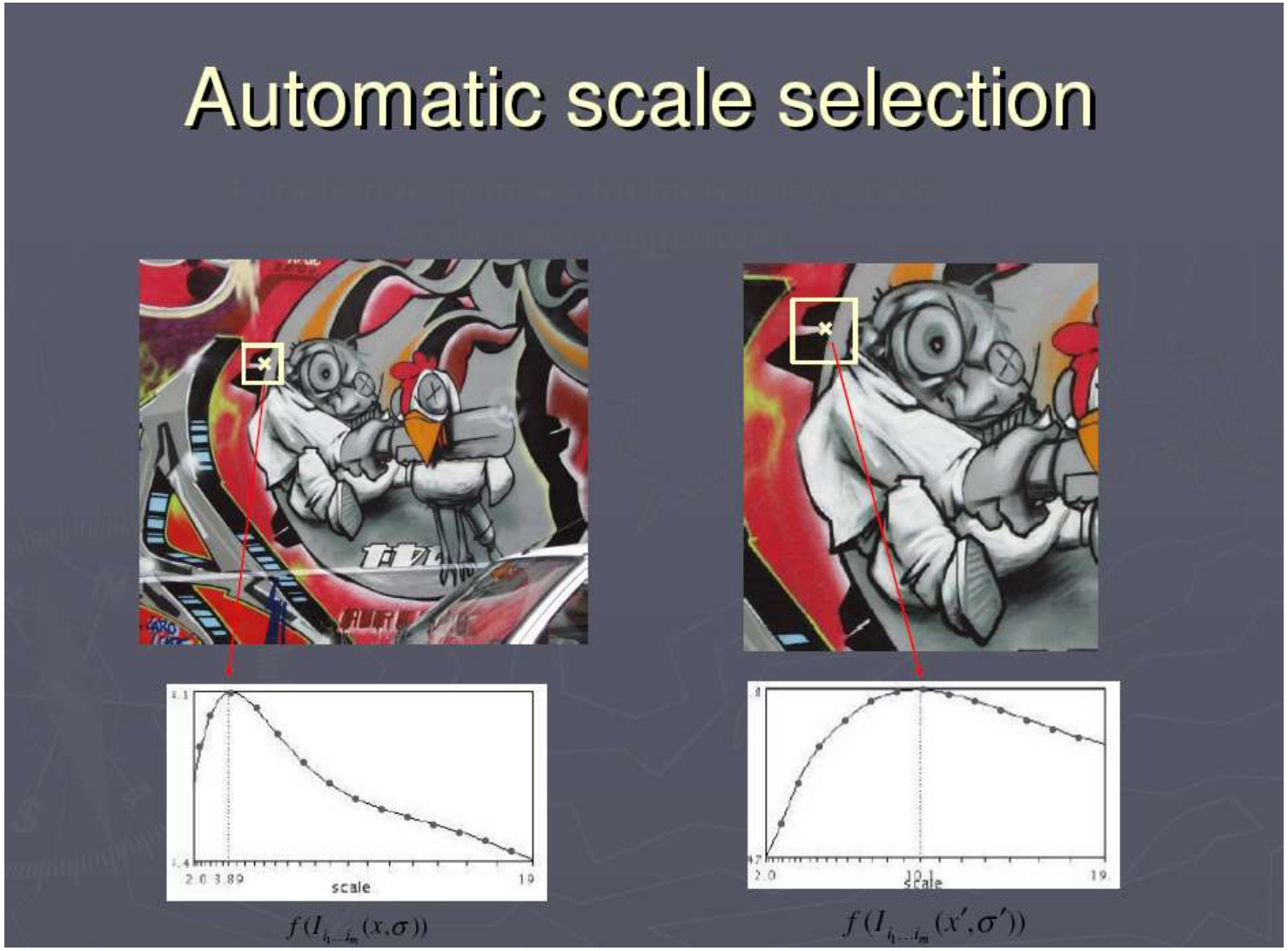
梯度角度直方图



SIFT Descriptors

- In principle: build a histogram of the gradients
- In reality: quite complicated
  - Gaussian weighting: smooth response
  - Normalization: reduces illumination effects
  - Clamping
  - Tons of more stuff

尺度不变性？特征尺度

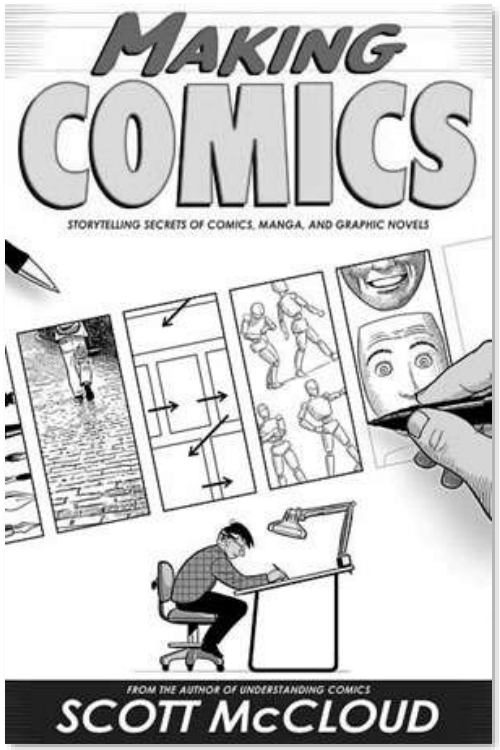

Automatic scale selection
scale
2.0
3.89
19
scale
2.0
10.1
19
$f(I_{i_1\ldots i_m}(x,\sigma))$
$f(I_{i_1\ldots i_m}(x',\sigma'))$

SIFT

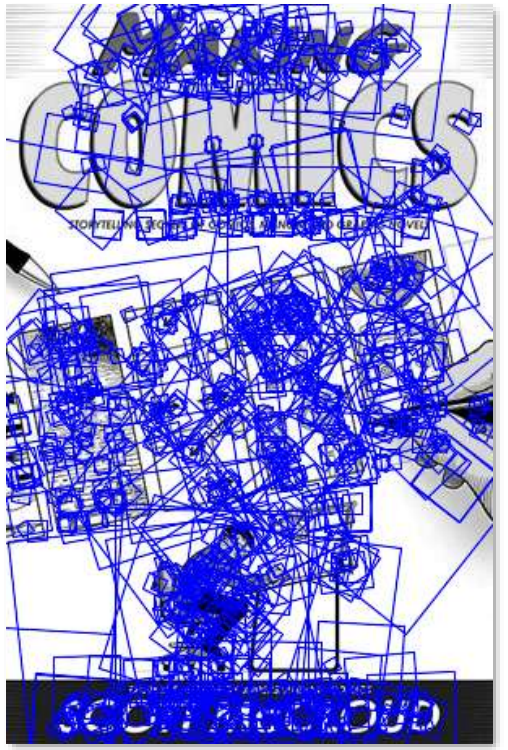
十分稳定的特征提取技术

- 视角变化(60 度以内的平面旋转)
- 光照变化(日景、夜景)
- 速度快!

SIFT



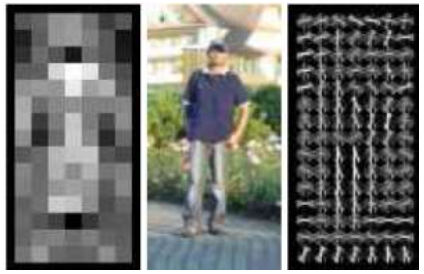
sift



868 SIFT 特征

其他特征提取器

- HOG: Histogram of Gradients (HOG)
  - 滑动窗口, 行人检测



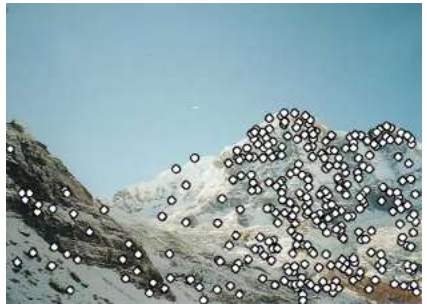
- FREAK: Fast Retina Keypoint
  - 用在SLAM 实时提取

- LIFT: Learned Invariant Feature Transform
  - 与深度学习结合，提取各种特征

    https://arxiv.org/abs/1603.09114

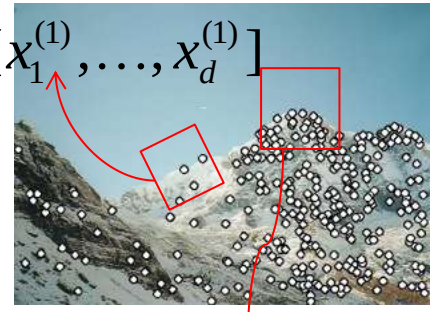
基于特征匹配的识别

1) 检测 Detection: 找到图中的关键点



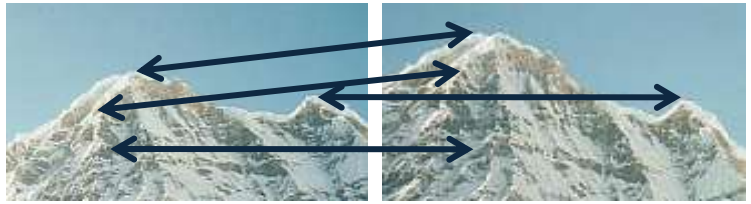
2) 解释 Description: 在关键点周围提取特征

$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$



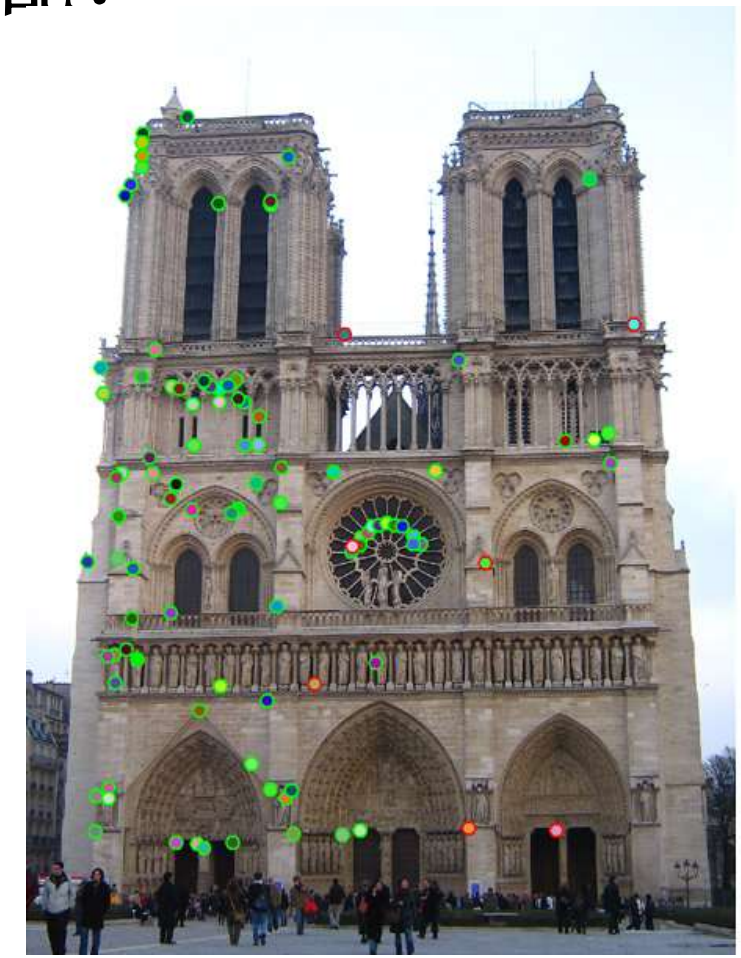
$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$

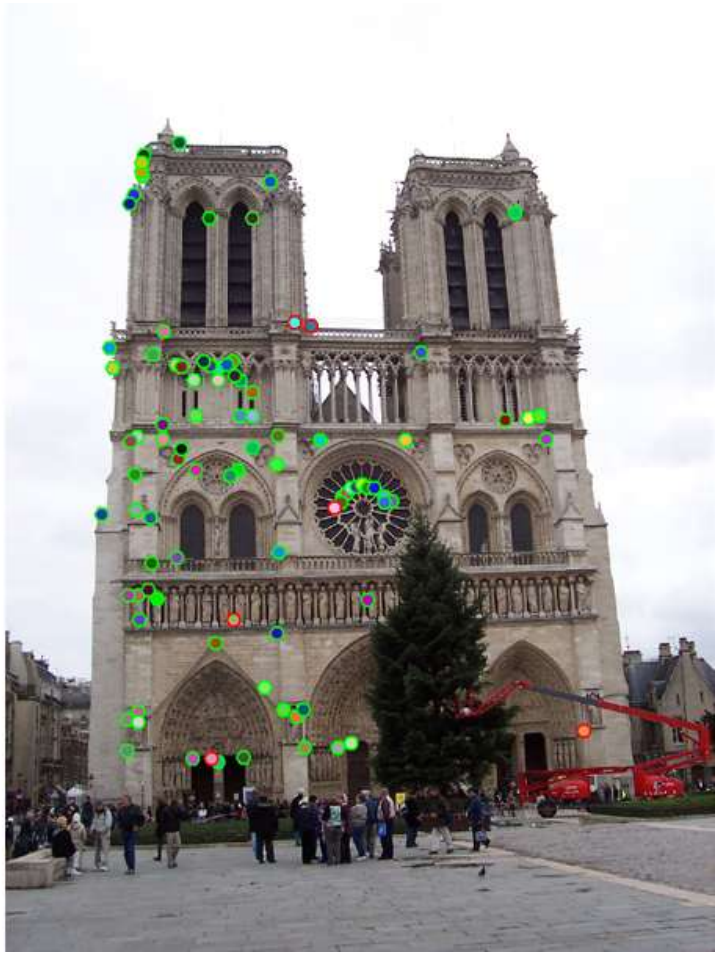
3) **匹配 Matching: 根据两个视角下的特征进行匹配**



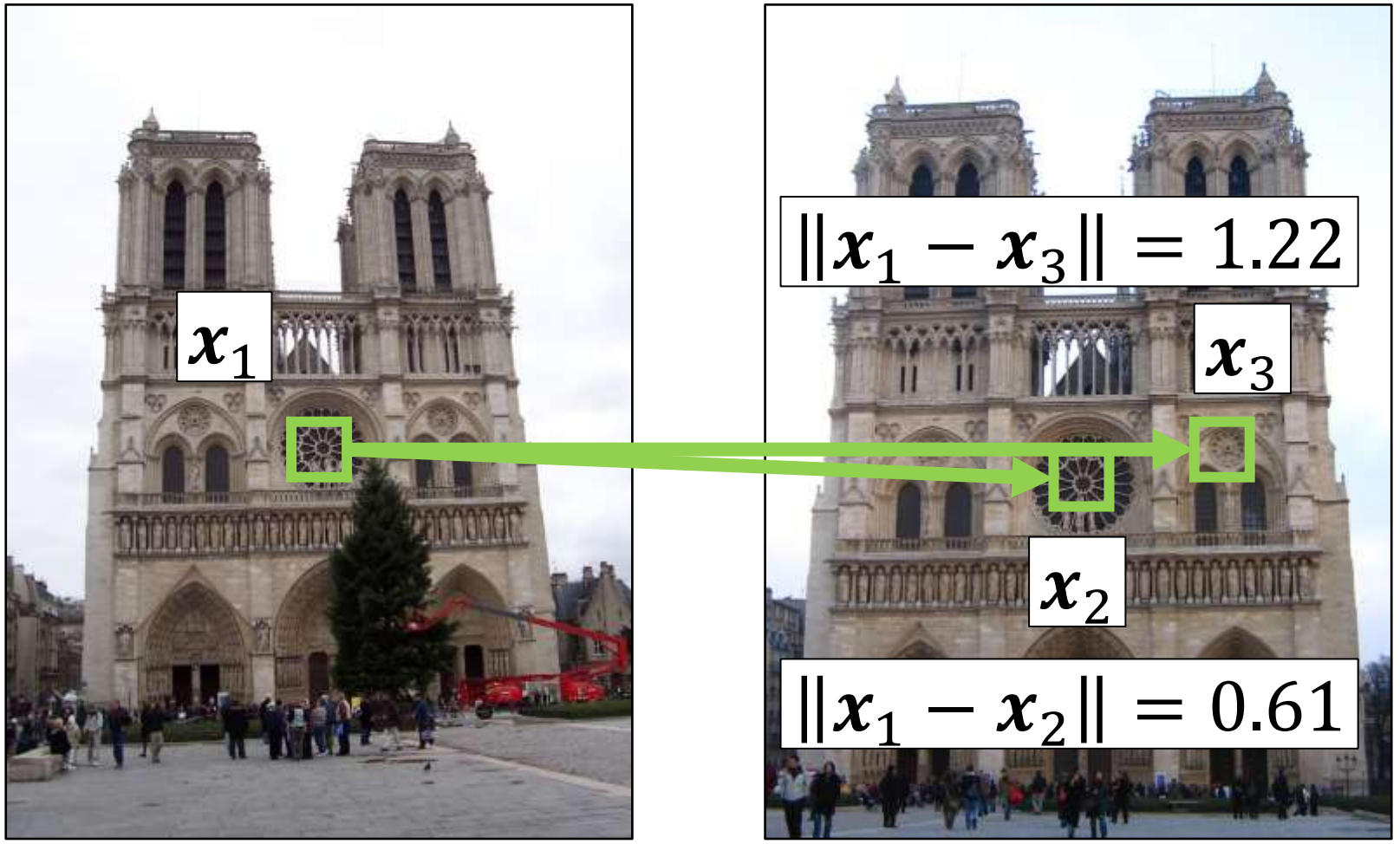
Kristen Grauman

怎么匹配？



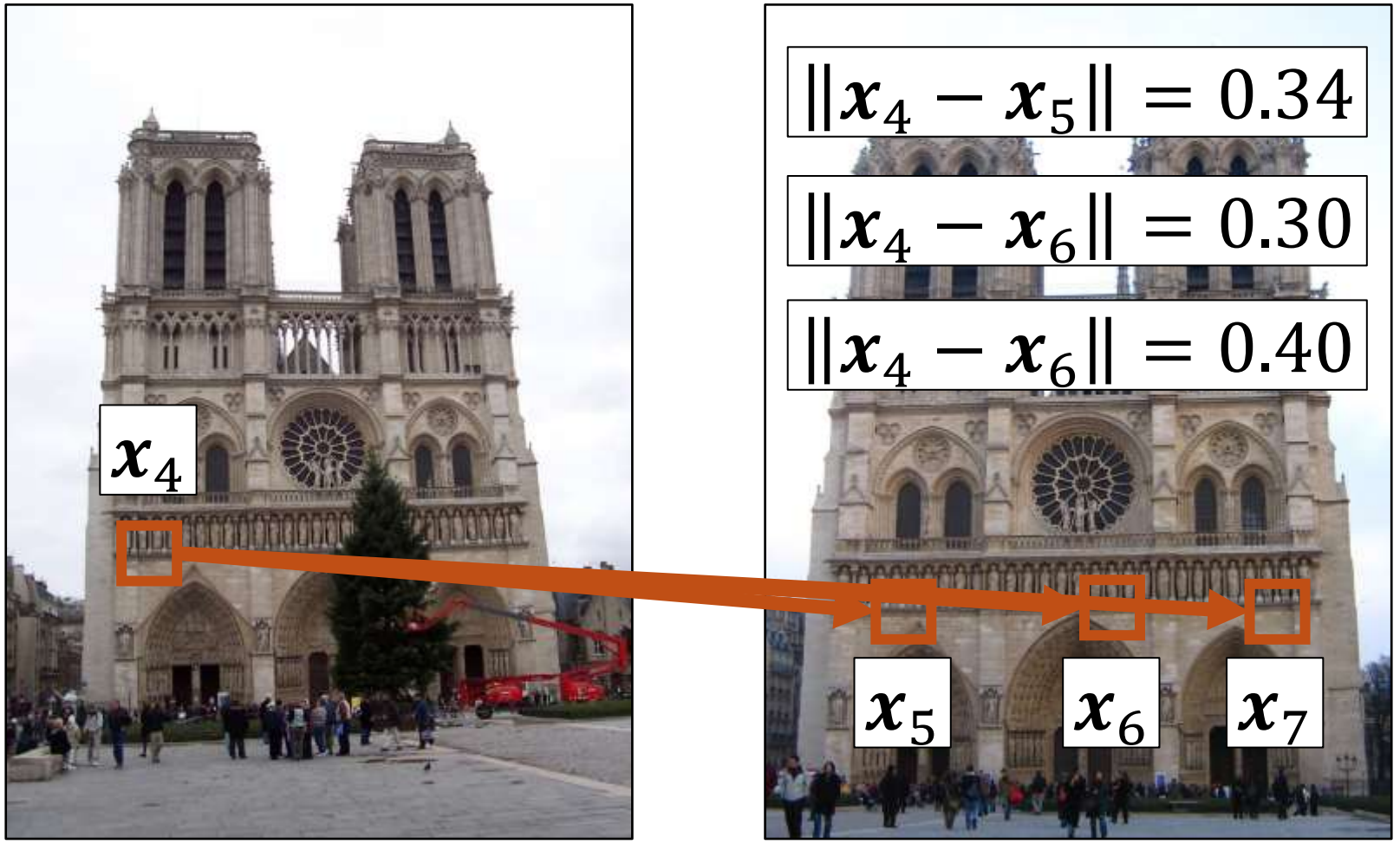


怎么匹配？



Example credit: J. Hays

怎么匹配？

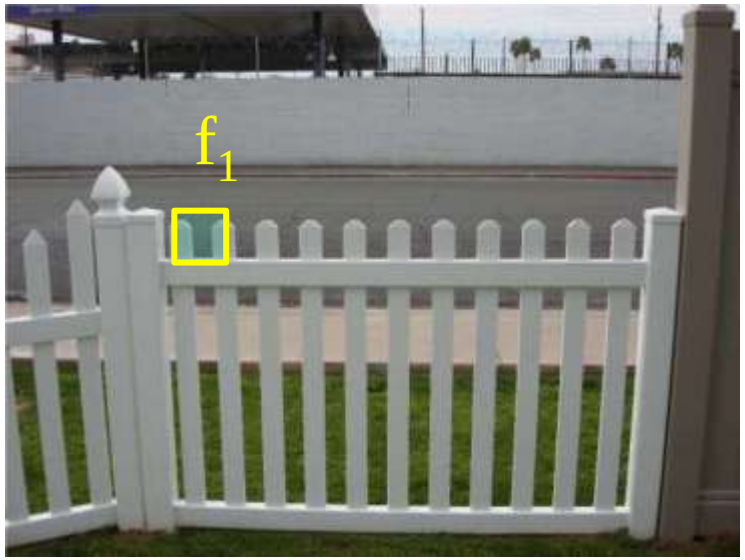


Example credit: J. Hays

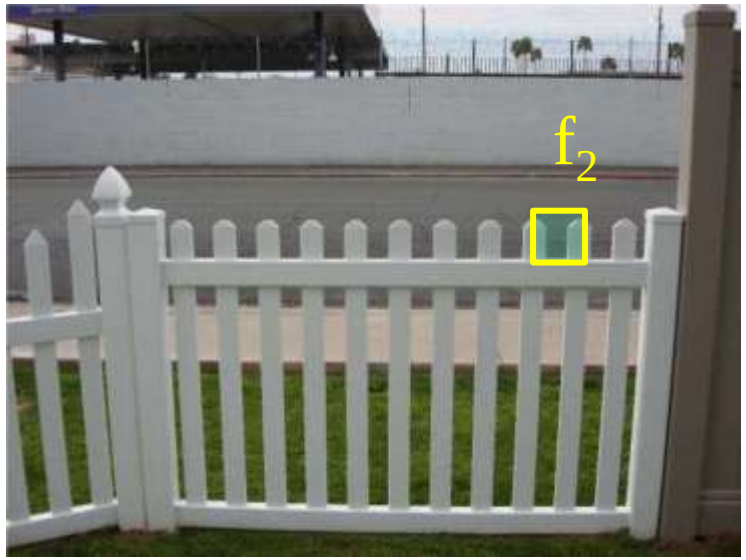
特征“距离”

怎么根据相似度匹配特征 f_1, f_2?

- 简单方法: L_2 distance, $\| f_1 - f_2 \|$
- 模糊匹配效果不好 (阈值法)



I_1

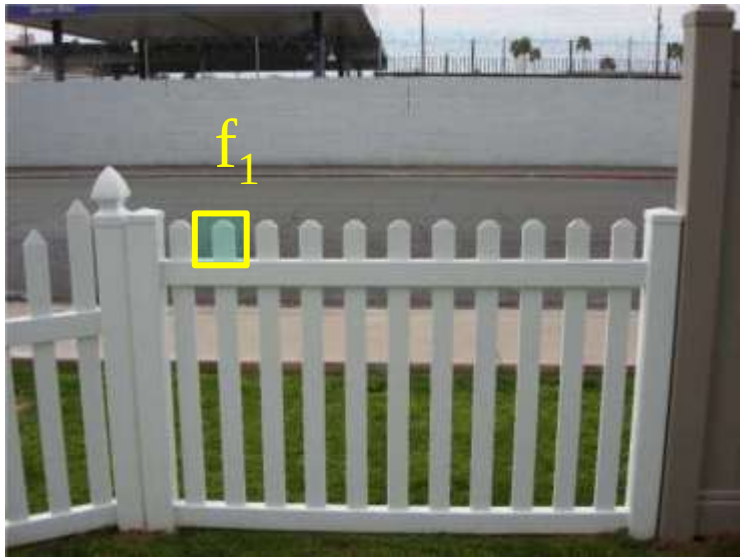


I_2

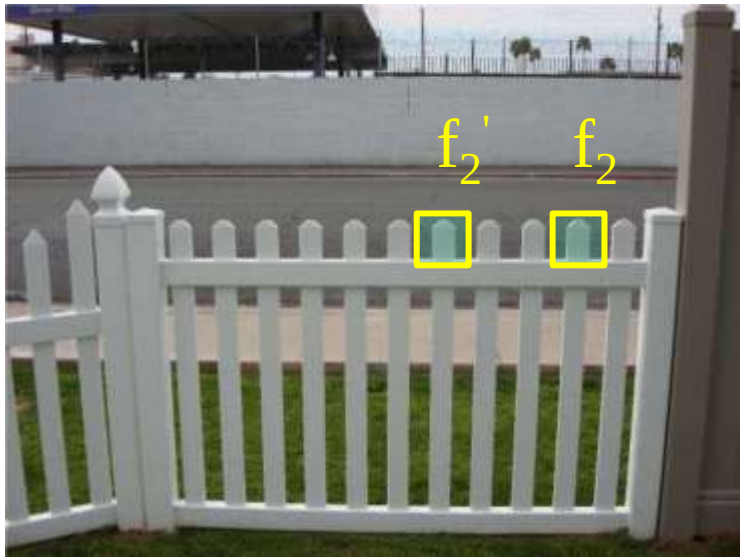
特征“距离”

怎么根据相似度匹配特征 f_1, f_2?

- 高级方法 2nd Nearest Neighbor Trick： $= \| f_1 - f_2 \| / \| f_1 - f_2' \|$
  - f_2 是 I_2 最匹配 f_1 的像素点
  - f_2' 是 I_2 第二匹配 f_1 的像素点
  - 模糊匹配效果较好:如果一个特征与其最近邻的距离与其与第二近邻的距离相差很大，那么这个匹配很可能是正确的。相反，如果这两个距离相差不大，那么这可能是一个误匹配



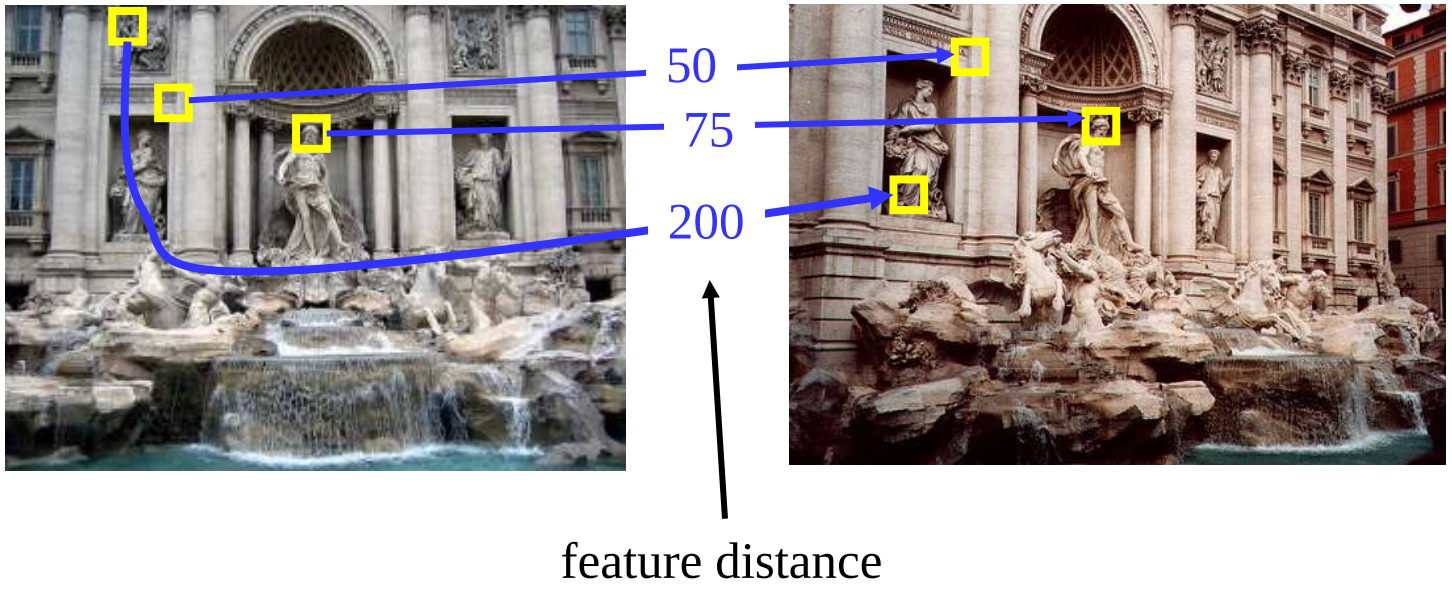
I_1



I_2

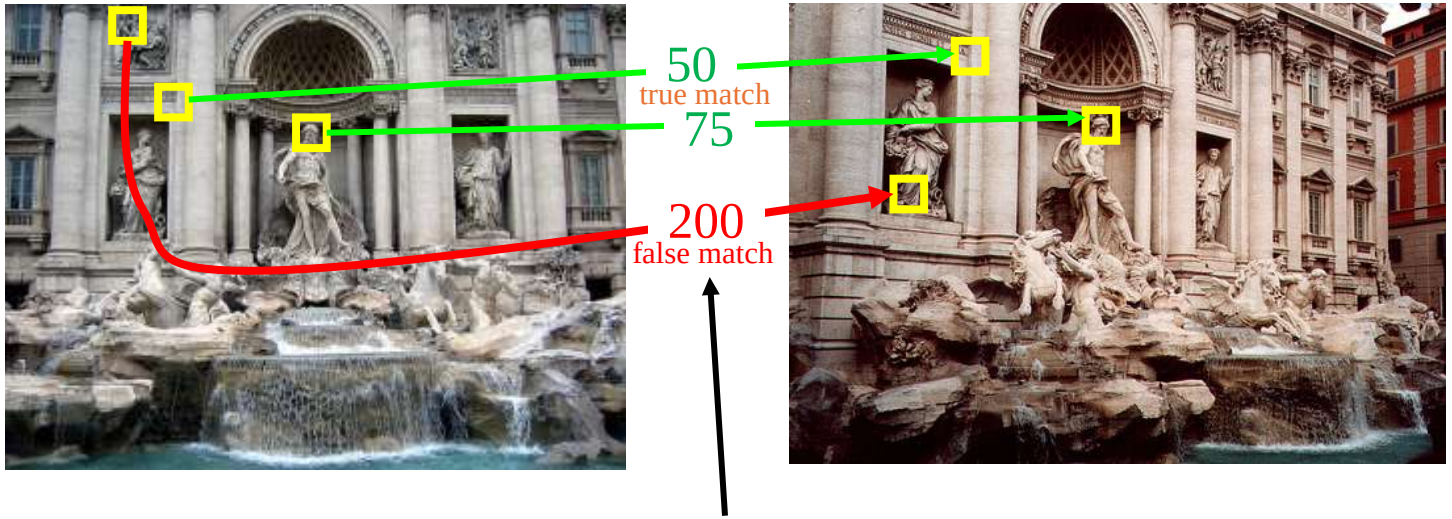
Evaluating the results

How can we measure the performance of a feature matcher?



True/false positives

How can we measure the performance of a feature matcher?

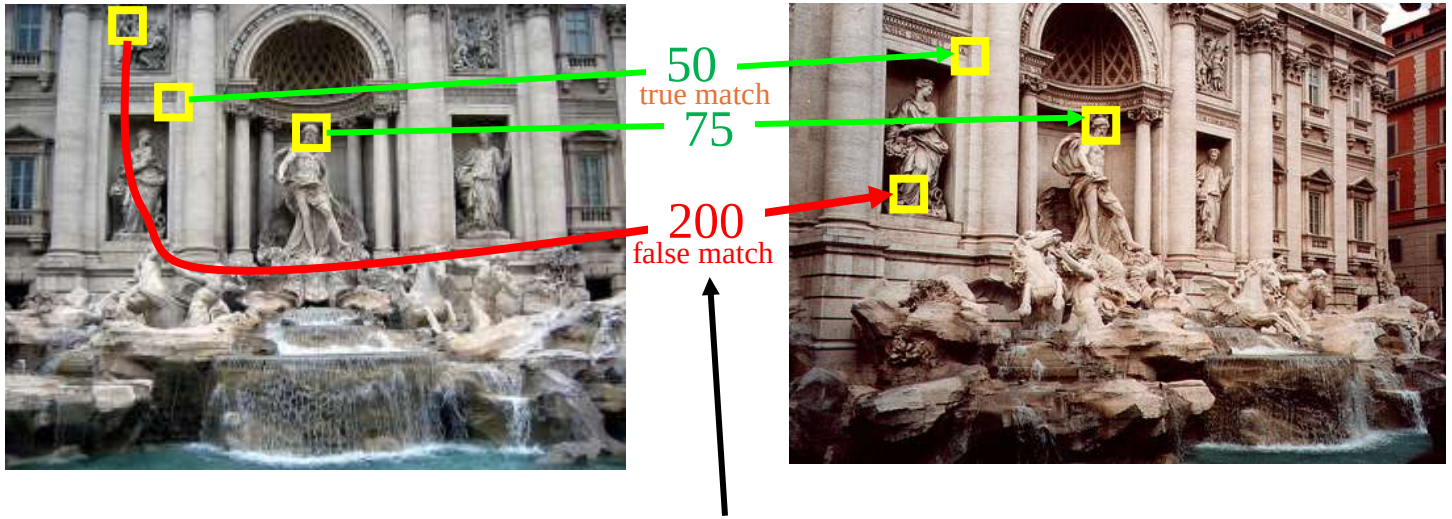


The distance threshold affects performance

- True positives = # of detected matches that survive the threshold that are correct
- False positives = # of detected matches that survive the threshold that are incorrect

True/false positives

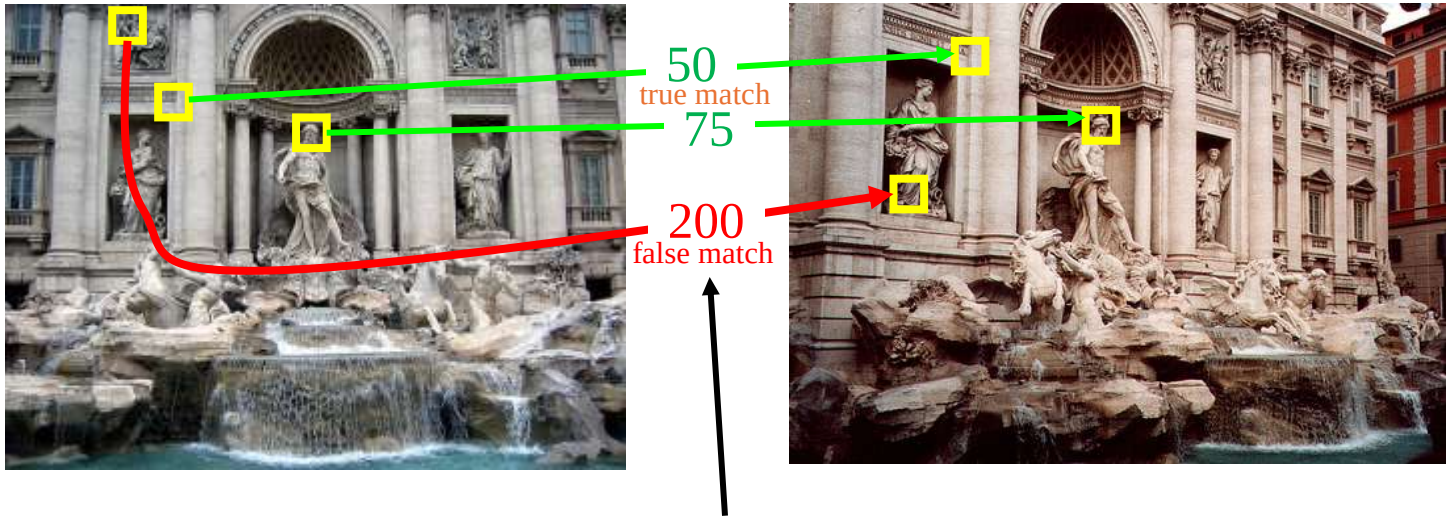
How can we measure the performance of a feature matcher?



Suppose we want to maximize true positives. How do we set the threshold? (We keep all matches with distance below the threshold.)

True/false positives

How can we measure the performance of a feature matcher?



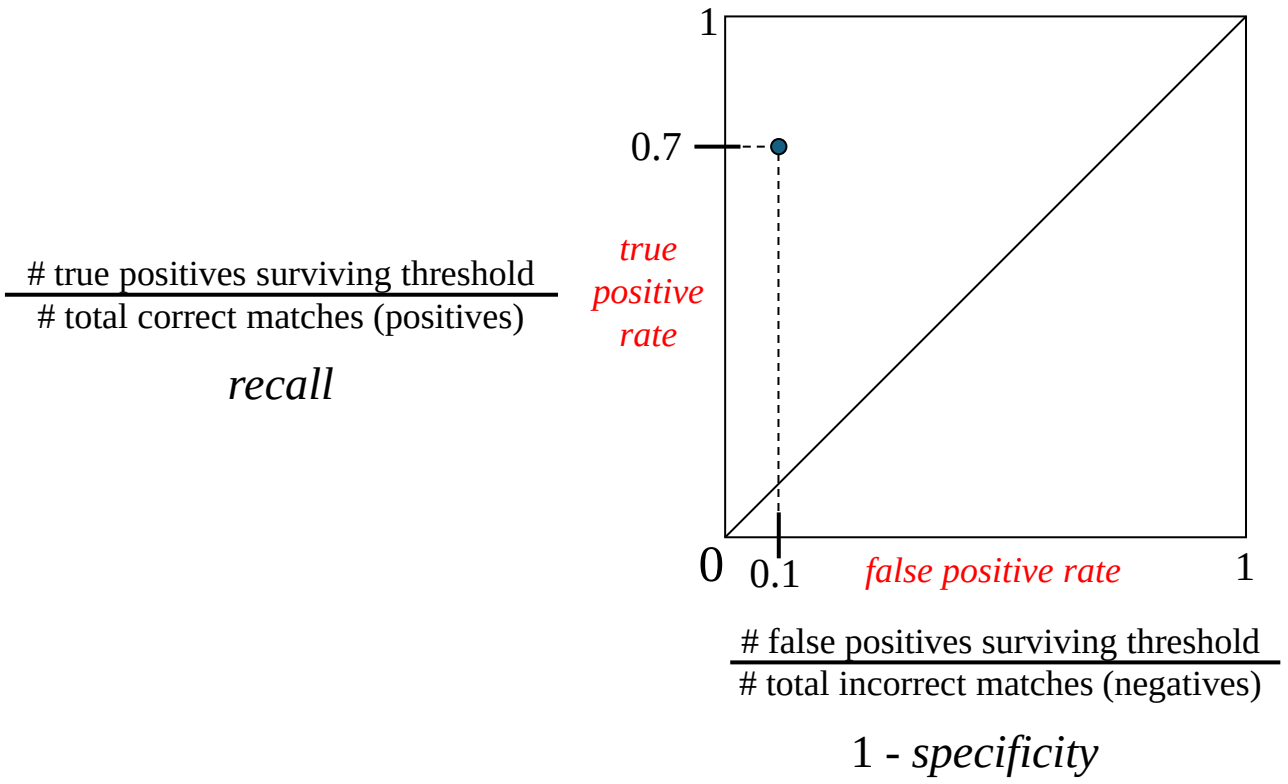
Suppose we want to minimize false positives. How do we set the threshold? (We keep all matches with distance below the threshold.)

Example

- Suppose our matcher computes 1,000 matches between two images
  - 800 are correct matches, 200 are incorrect (according to an oracle that gives us ground truth matches)
  - A given threshold (e.g., ratio distance = 0.6) gives us 600 correct matches and 100 incorrect matches that survive the threshold
  - True positive rate = 600 / 800 = ¾
  - False positive rate = 100 / 200 = ½

Evaluating the results

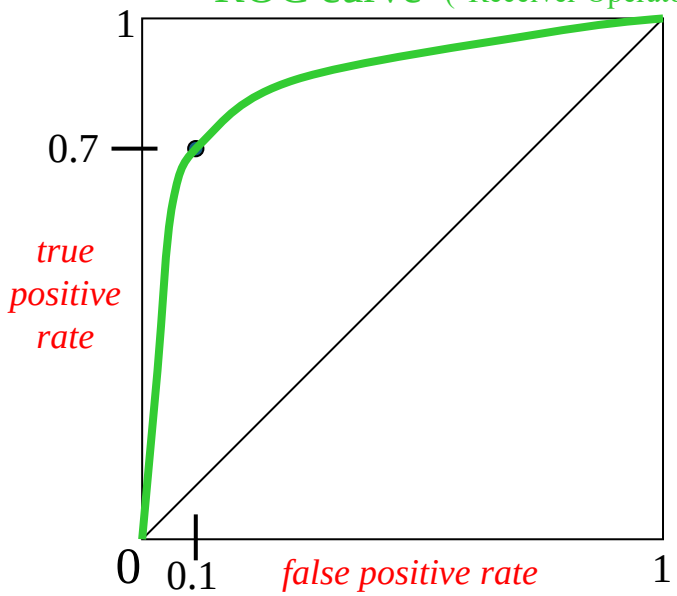
How can we measure the performance of a feature matcher?



Evaluating the results

How can we measure the performance of a feature matcher?

ROC curve ("Receiver Operator Characteristic")



$$\frac{\text{\# true positives surviving threshold}}{\text{\# total correct matches (positives)}}$$

recall

$$\frac{\text{\# false positives surviving threshold}}{\text{\# total incorrect matches (negatives)}}$$

1 - *specificity*

Single number: Area Under the Curve (AUC)

E.g. AUC = 0.87

1 is the best

ROC curves – summary

- By thresholding the match distances at different thresholds, we can generate sets of matches with different true/false positive rates
- ROC curve is generated by computing rates at a set of threshold values swept through the full range of possible threshold
- Area under the ROC curve (AUC) summarizes the performance of a feature pipeline (higher AUC is better)

More on feature detection/description

http://www.robots.ox.ac.uk/~vgg/research/affine/
http://www.cs.ubc.ca/~lowe/keypoints/
http://www.vision.ee.ethz.ch/~surf/

Publications

Region detectors

- *Harris-Affine & Hessian Affine*: K. Mikolajczyk and C. Schmid, Scale and Affine invariant interest point detectors. In IJC V 60(1):63-86, 2004. PDF
- *MSER*: J.Matas, O. Chum, M. Urban, and T. Pajdla, Robust wide baseline stereo from maximally stable extremal regions. In BMVC p. 384-393, 2002. PDF
- *IBR & EBR*: T.Tuytelaars and L. Van Gool, Matching widely separated views based on affine invariant regions . In IJCV 59(1):61-85, 2004. PDF
- *Salient regions*: T. Kadir, A. Zisserman, and M. Brady, An affine invariant salient region detector. In ECCV p. 404-416, 2004. PDF
- *All Detectors - Survey*: T. Tuytelaars and K. Mikolajczyk , Local Invariant Feature Detectors - Survey. In CVG, 3(1):1-110, 2008. PDF

Region descriptors

- *SIFT*: D. Lowe, Distinctive image features from scale invariant keypoints. In IJCV 60(2):91-110, 2004. PDF

Performance evaluation

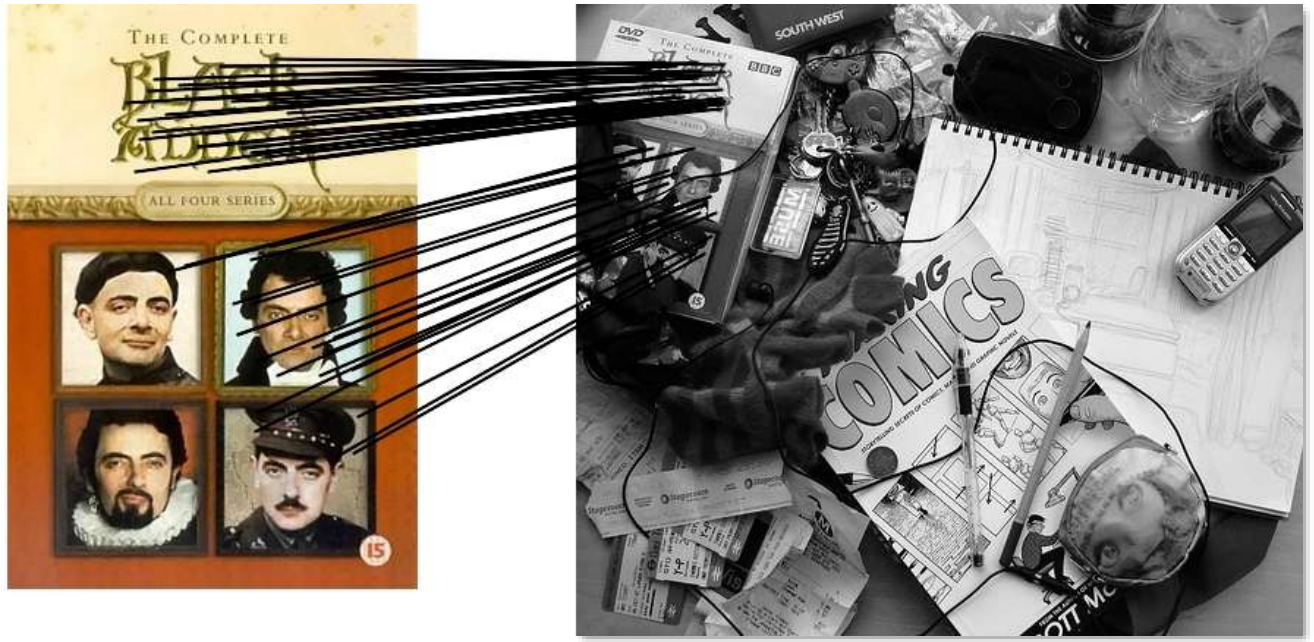
- K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir and L. Van Gool, A comparison of affine region detectors. In IJCV 65(1/2):43-72, 2005. PDF
- K. Mikolajczyk, C. Schmid, A performance evaluation of local descriptors. In PAMI 27(10):1615-1630 . PDF

Lots of applications

Features are used for:

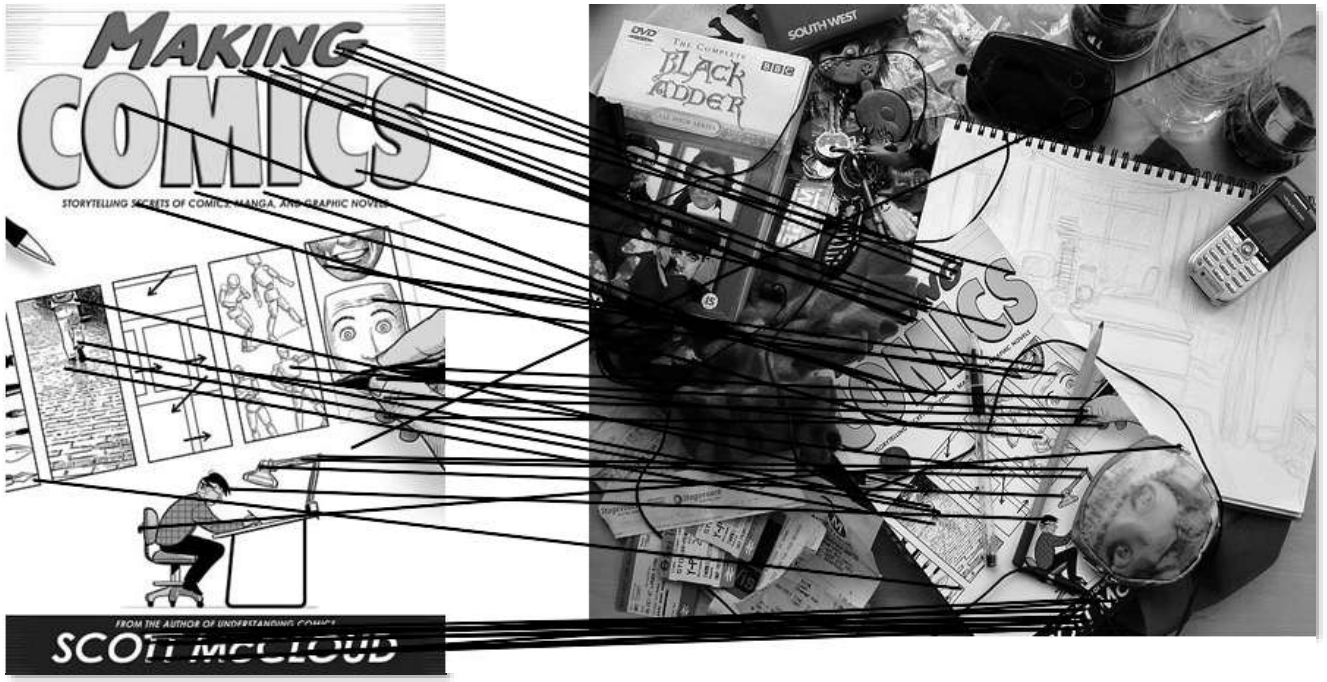
- Image alignment (e.g., mosaics)
- 3D reconstruction
- Motion tracking
- Object recognition
- Indexing and database retrieval
- Robot navigation
- … other

特征匹配



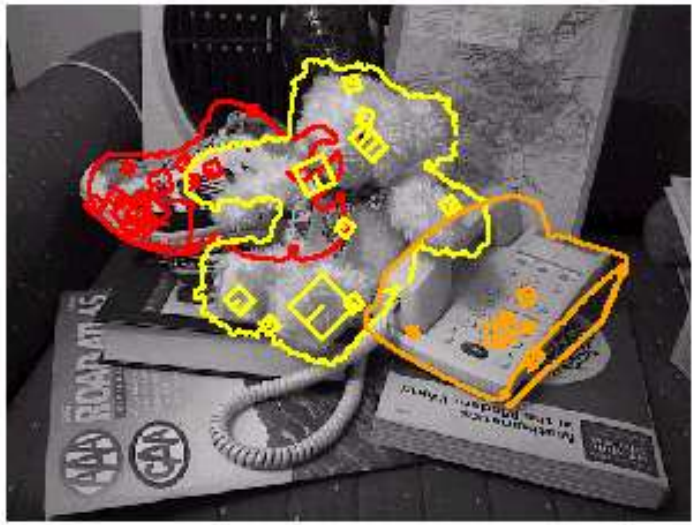
58 处匹配 (thresholded by ratio score)

特征匹配——Outliers

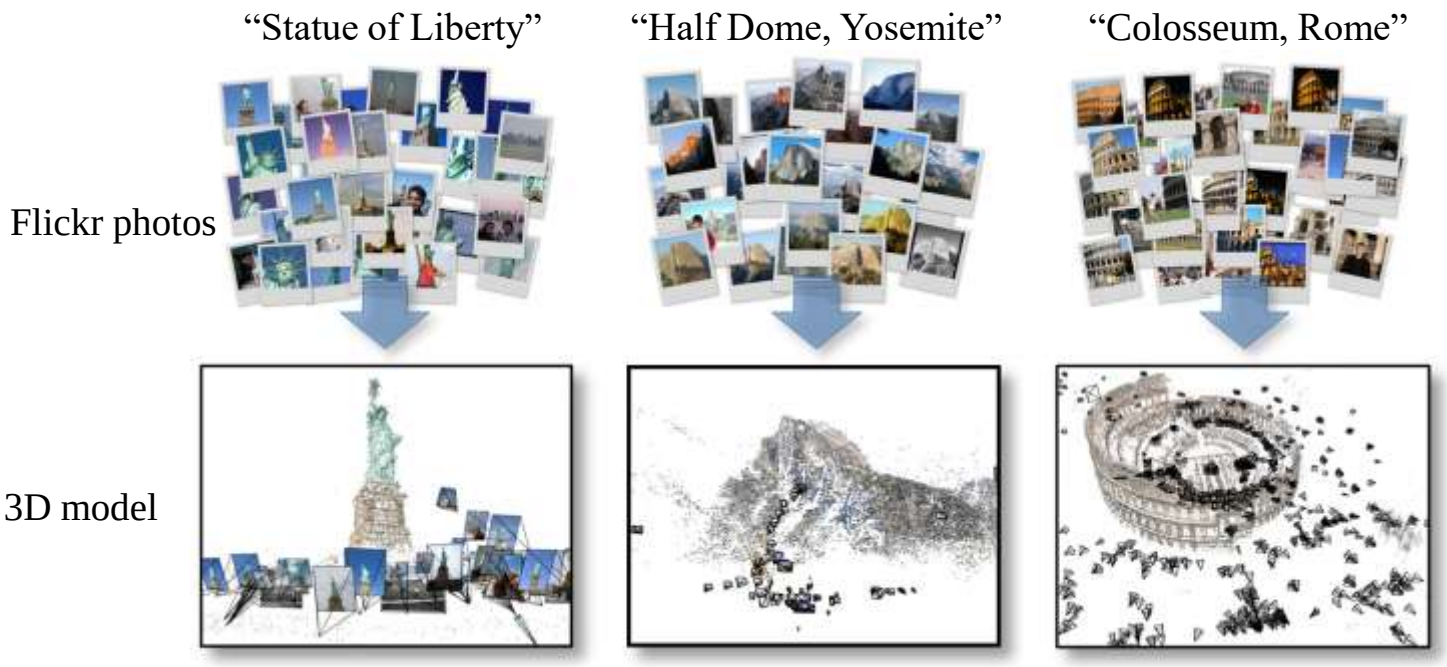


51 处匹配(thresholded by ratio score)

物体识别 (David Lowe)

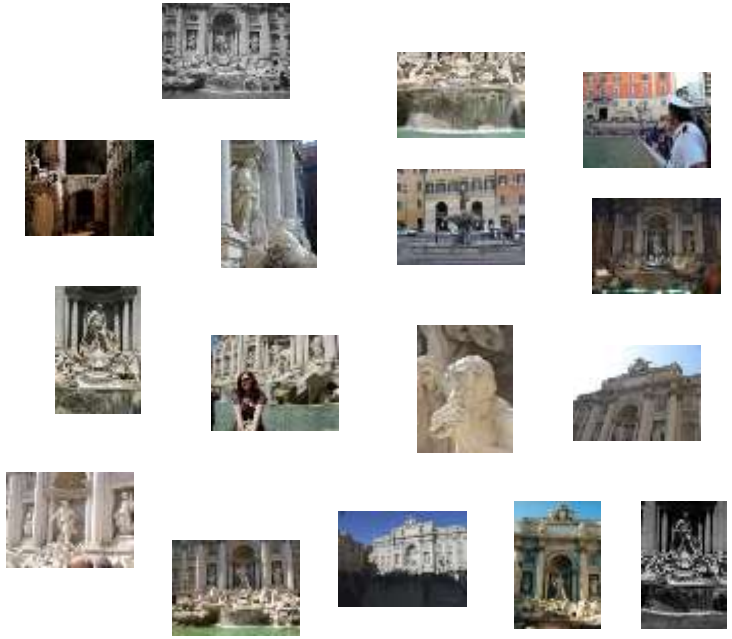


从网络照片重构一个场景



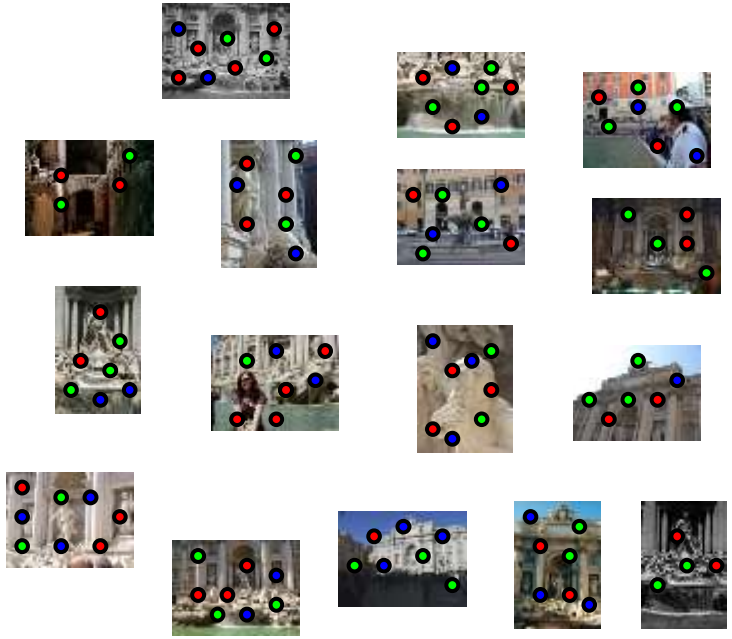
特征检测

使用SIFT检测特征 [Lowe, IJCV 2004]



特征检测

使用SIFT检测特征 [Lowe, IJCV 2004]



相似度关联

- 把不同图像的关键点通过相似程度联系起来

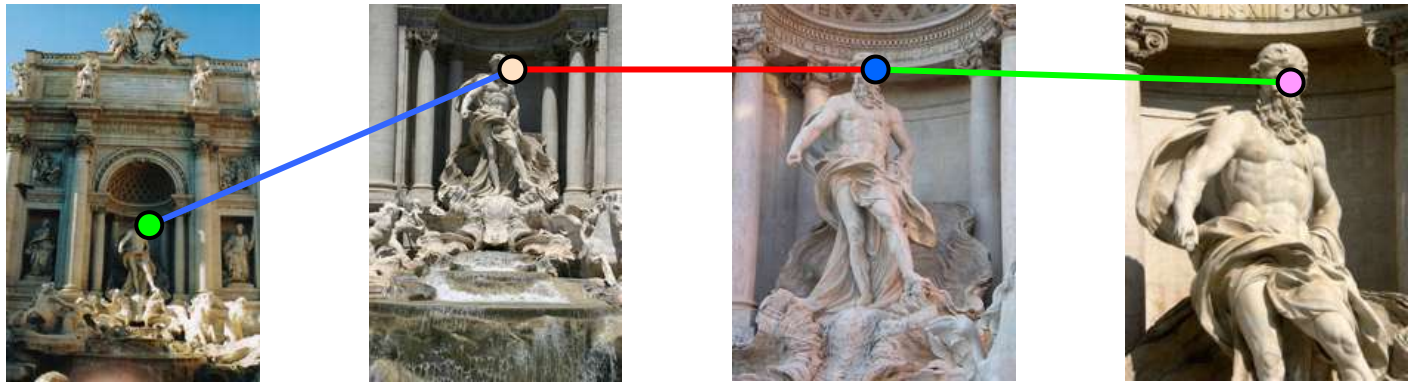
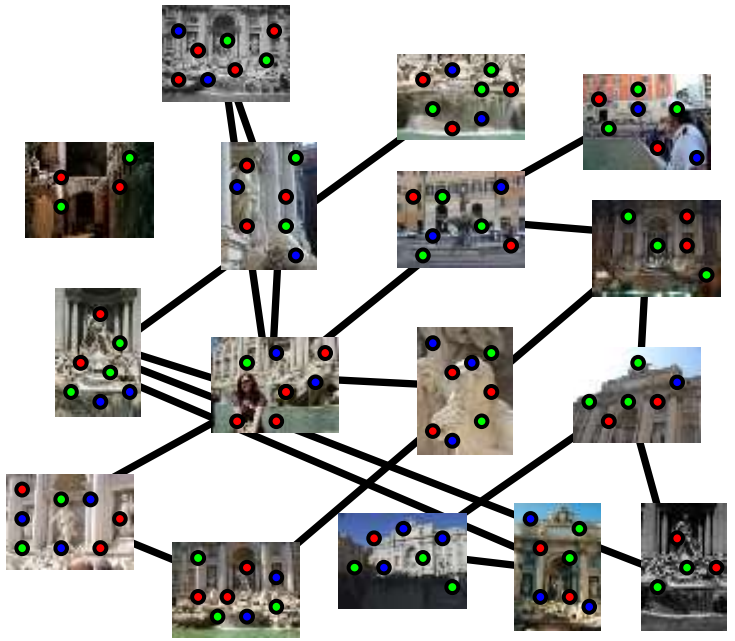


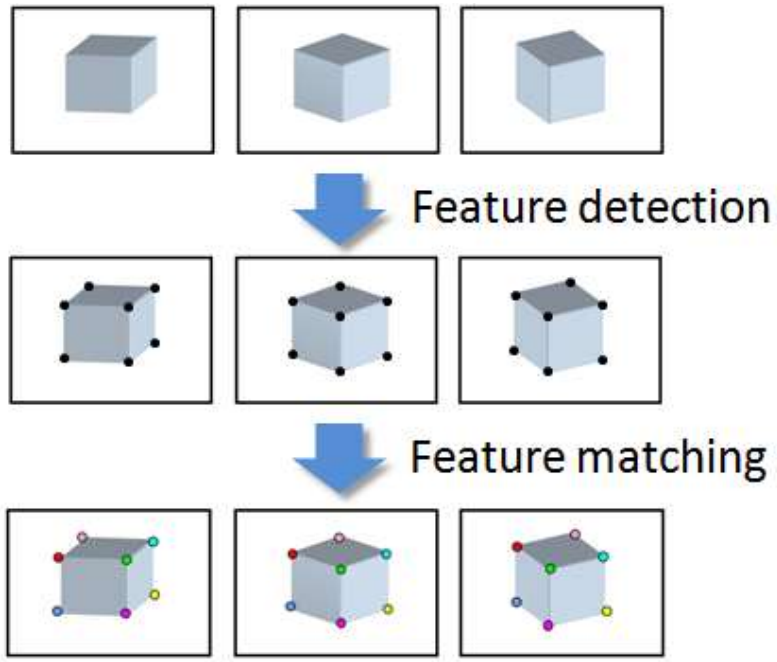
Image 1 Image 2 Image 3 Image 4

特征匹配

基于每对做特征匹配



计算场景结构



与二维图像整合方法一致

城镇级的三维重建

Reconstruction of Dubrovnik, Croatia, from ~40,000 images

场景与光照结合





“感知”

计算机系统对数字图像和视频进行分析和理解的能力，以获得有关物体、场景、特征和动作的信息。感知的目标是模拟和模仿人类视觉系统，使计算机能够理解和解释图像中的内容。

特征信息提取、对象识别、场景理解、运动分析、三维深度感知、上下文感知 ……

什么是 “识别”?

Next few slides adapted from Li, Fergus, & Torralba’s excellent short course on category and object recognition

什么是“识别”?

- Verification: is that a lamp?
  - 验证：那是一盏路灯吗？

什么是 “识别”?

- Verification: is that a lamp?
- Detection: where are the people?
  - 检测：人们在哪里？

什么是 “识别”?

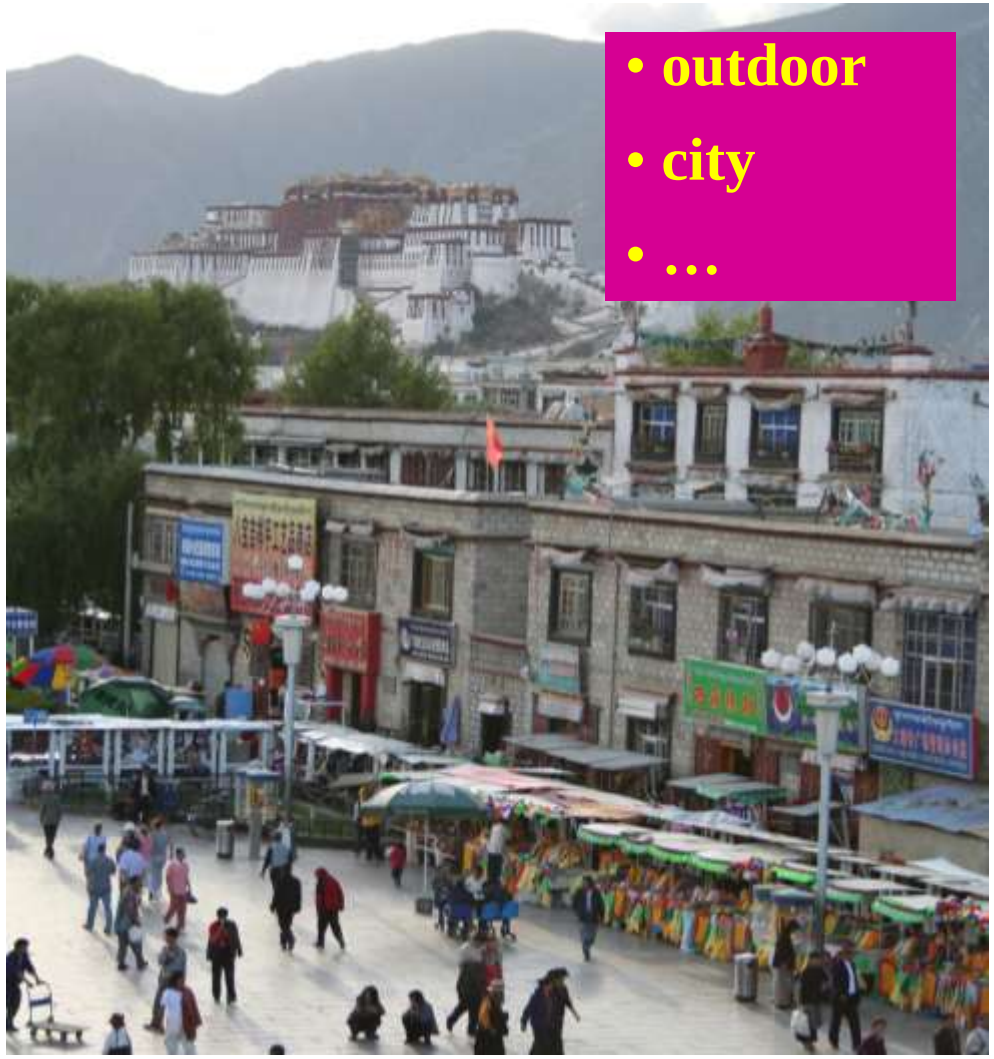
- Verification: is that a lamp?
- Detection: where are the people?
- Identification: is that Potala Palace?
  - 辨认：那是布达拉宫吗？

什么是 “识别”?

- Verification: is that a lamp?
- Detection: where are the people?
- Identification: is that Potala Palace?
- Object categorization
  - 对象分类

什么是“识别”?

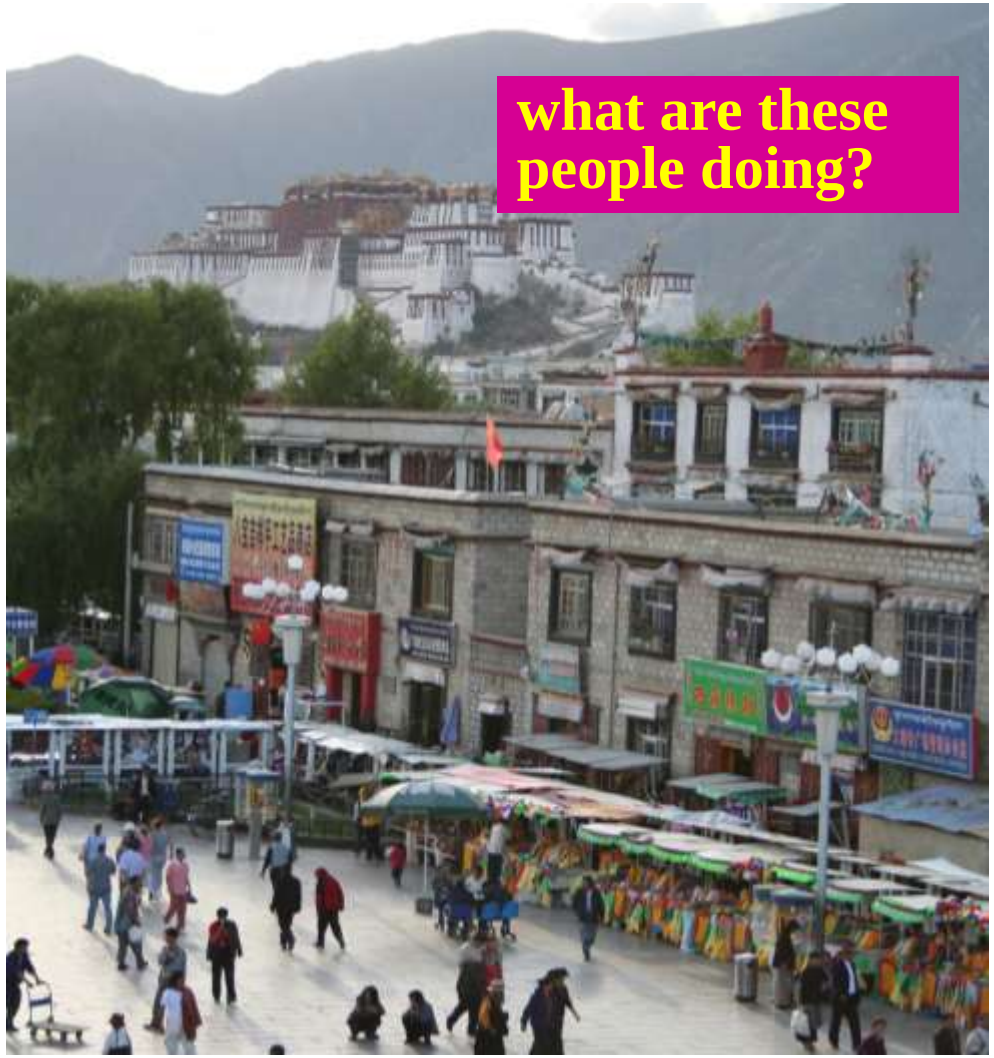
- Verification: is that a lamp?
- Detection: where are the people?
- Identification: is that Potala Palace?
- Object categorization
- Scene and context categorization
  - 场景识别与理解



什么是 “识别”?

- Verification: is that a lamp?
- Detection: where are the people?
- Identification: is that Potala Palace?
- Object categorization
- Scene and context categorization

- Activity / Event Recognition
  - 动作、事件识别



难点：不同视角

Michelangelo 1475-1564

难点：不同光照条件

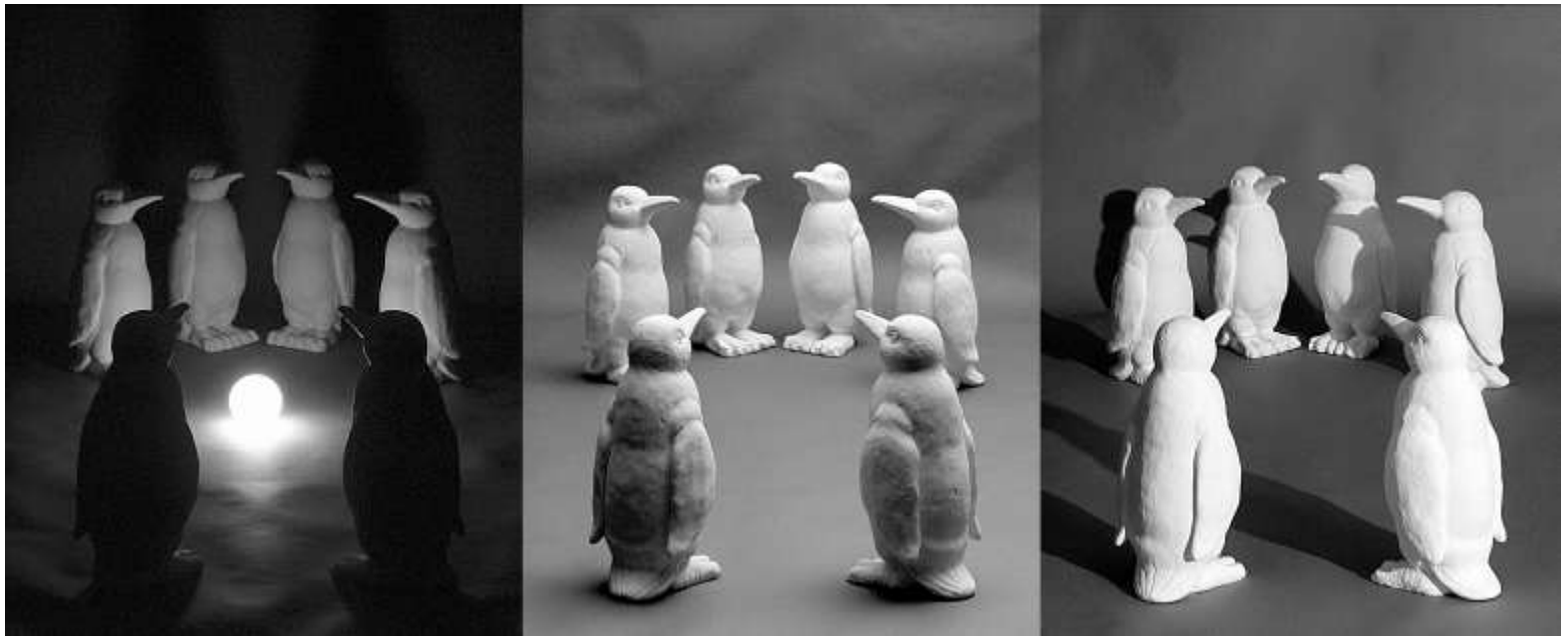
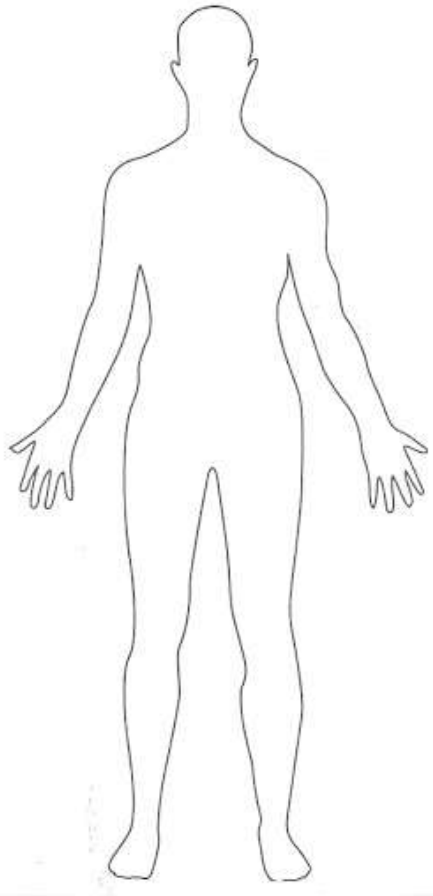


image credit: J. Koenderink

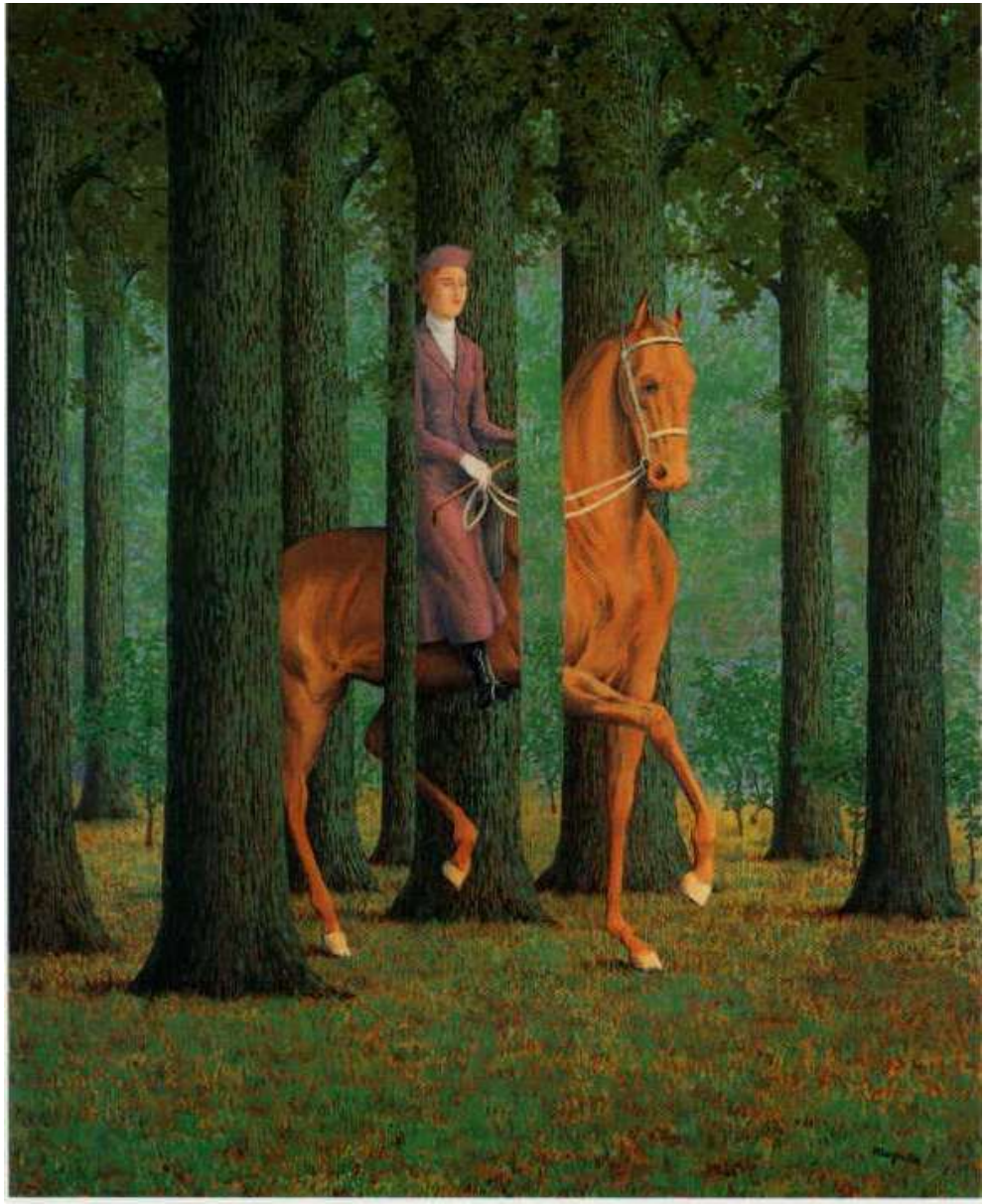
难点：尺度

难点：不同形式




London 2012
London 2012
Punch Layout
Arabian
Double Front
Roundoff,
Back Handspring
Roundoff,
One-and-a-half Stepout
© Corbis/Getty Images

难点：遮挡



Magritte, 1957

难点：背景伪装

Kilmeny Niland. 1995

难点：类内差异

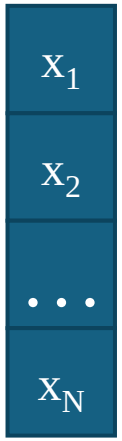
同类目标外观差异很大

难点：类内差异

让我们从最简单的分类开始

分类最简单的情况：二分类

我们会用一个特征向量
代表这张图



分类最简单的形式

记住所有类别的训练样本

If this:
cat.

If this:
dog.

If this:
hippo.

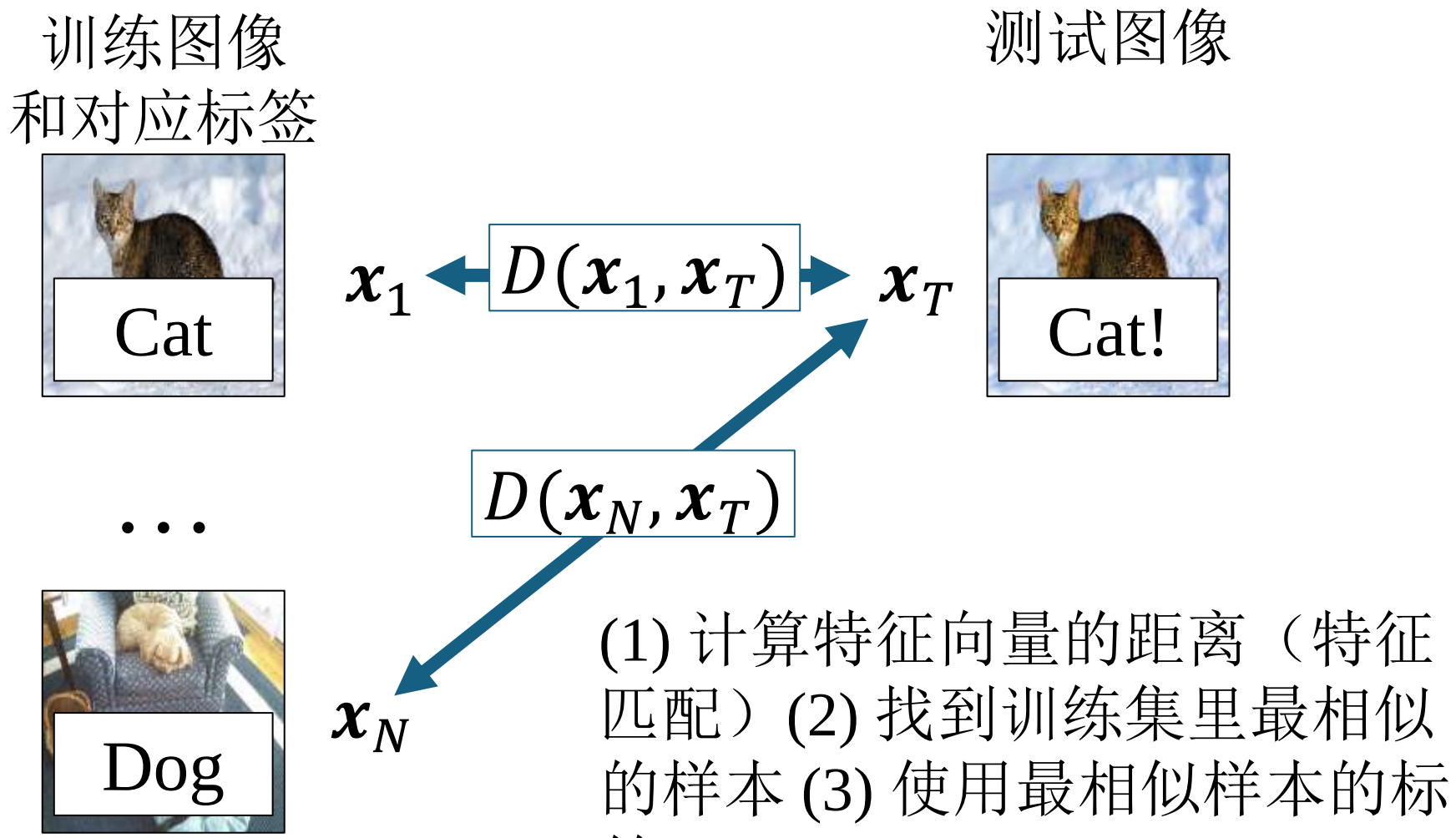
分类最简单的形式

这么做有什么问题？

Rule: if this,
then cat

似乎跟记忆不太一样？

分类最简单的形式：最近邻



(1) 计算特征向量的距离（特征匹配）(2) 找到训练集里最相似的样本 (3) 使用最相似样本的标签.

最近邻算法

训练 ($\mathbf{x}_i$,y_i): 记住所有训练样本

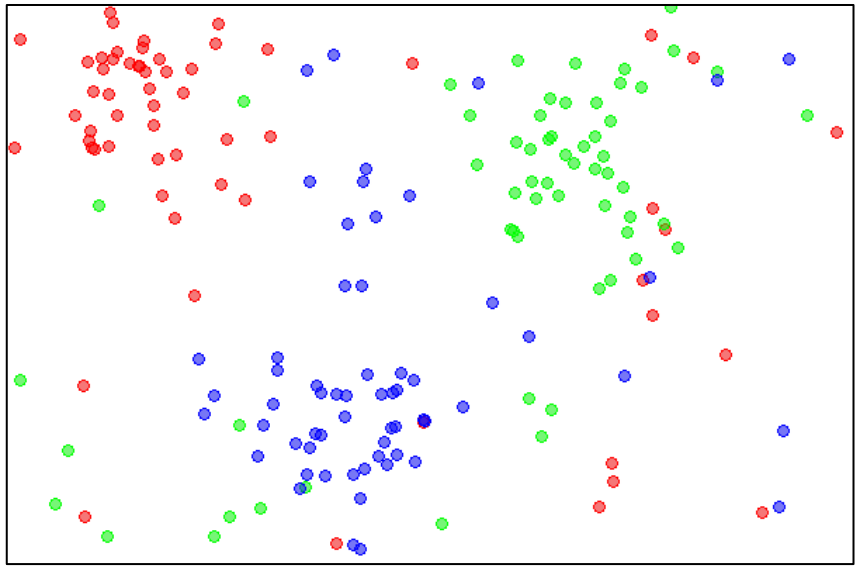
测试 (x):

```
bestDist, prediction = Inf, None
for i in range(N):
        if dist(x_i,x) < bestDist:
                bestDist =
dist(x_i,x)
                prediction = y_i
```

最近邻算法

可能出现什么问题？

2D Datapoints
(colors = labels)



2D Predictions
(colors = labels)

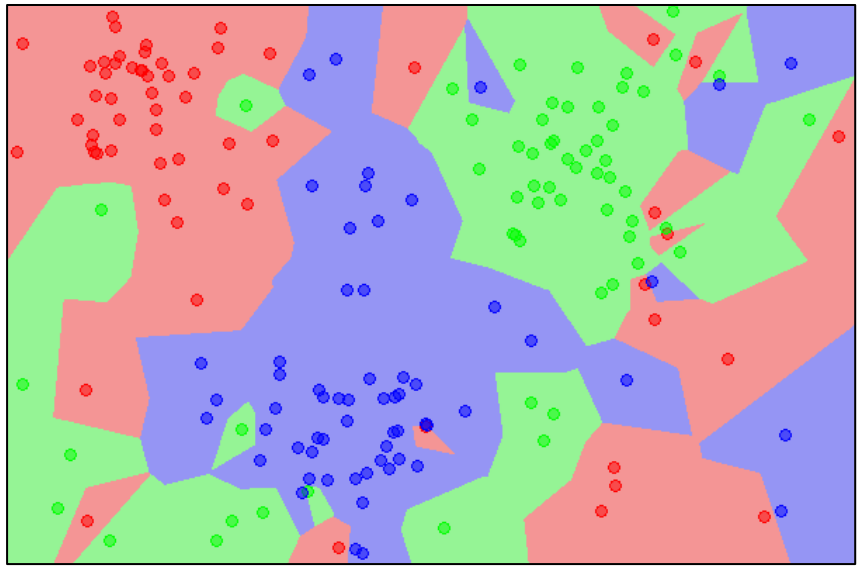
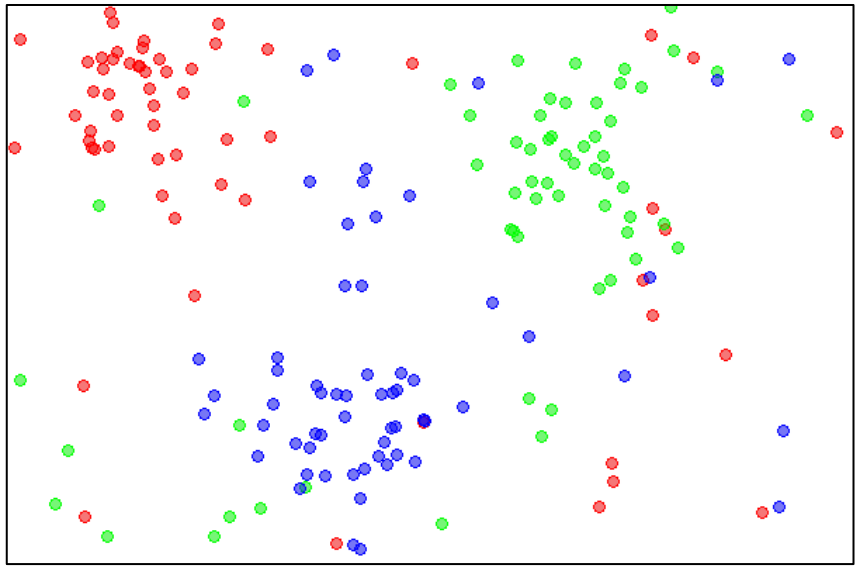


Diagram Credit: Wikipedia

K近邻

找到前K近邻样本，然后投票决定标签

2D Datapoints
(colors = labels)



2D Predictions
(colors = labels)

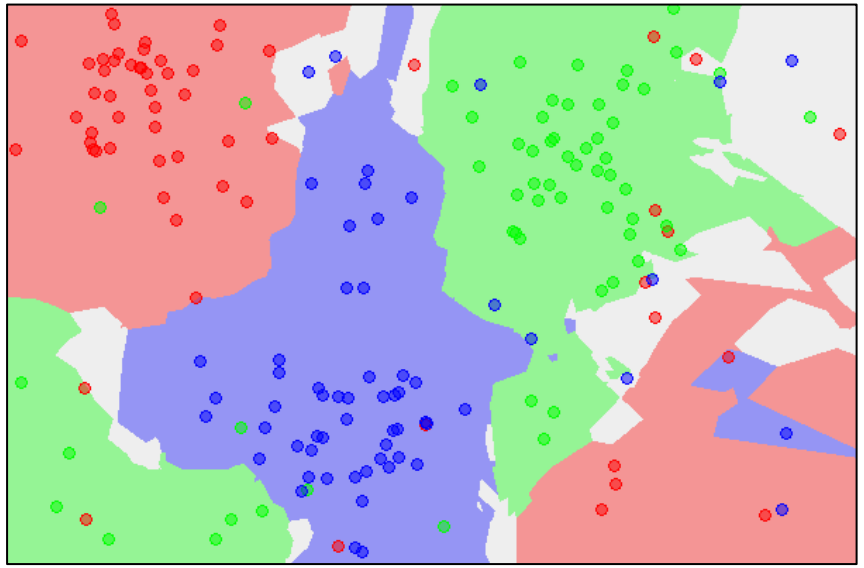
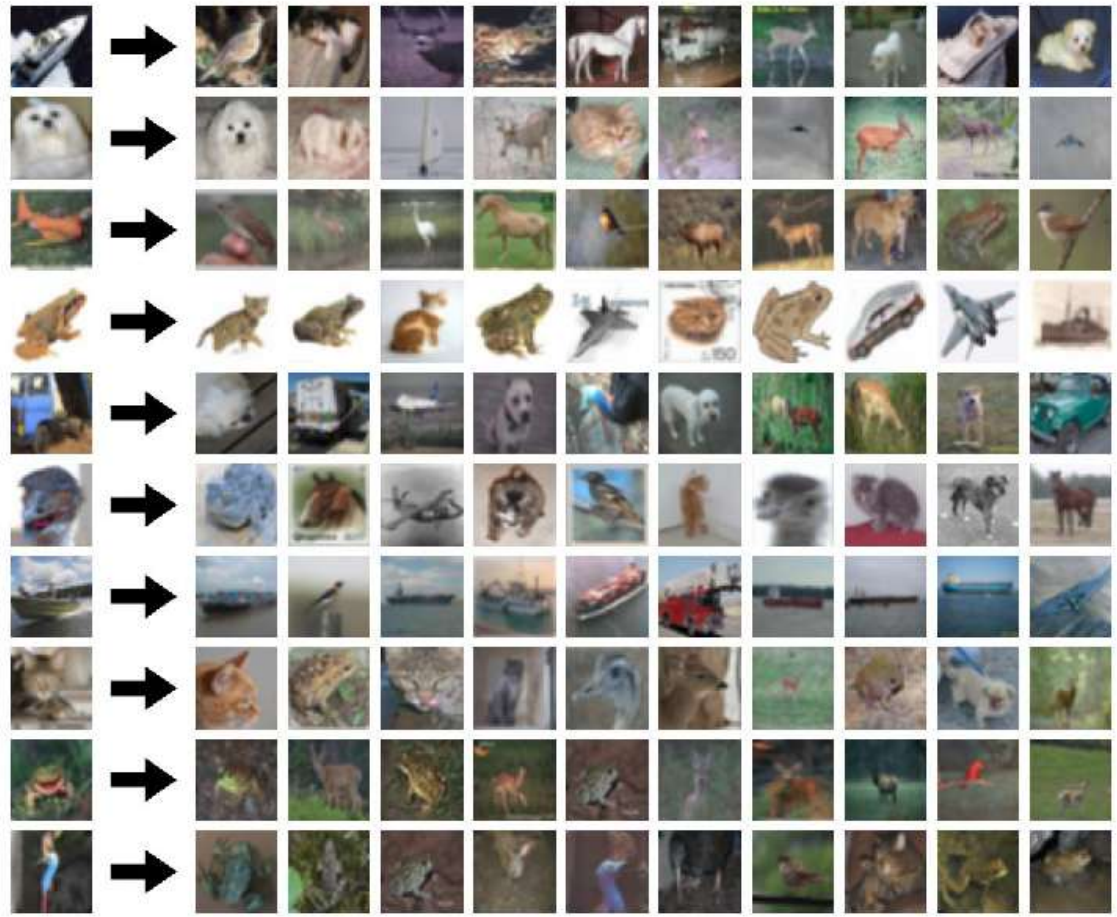


Diagram Credit: Wikipedia

What does this look like?



What does this look like?

How to Define Distance Between Images

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

test image					training image				pixel-wise absolute value differences				
56	32	10	18	−	10	20	24	17	46	12	14	1	→ 456
90	23	128	133		8	10	89	100	82	13	39	33	
24	26	178	200		12	16	178	170	12	10	0	30	
2	0	255	220		4	32	233	112	2	32	22	108	

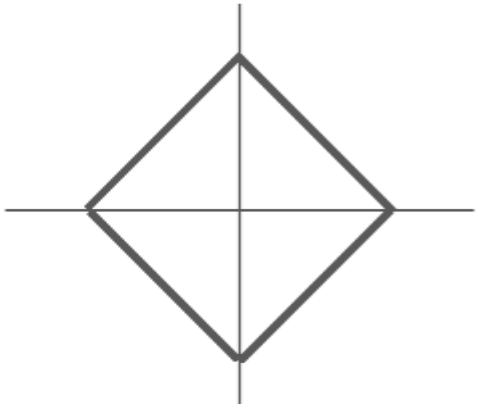
Slides from Andrej Karpathy and Fei-Fei Li http://vision.stanford.edu/teaching/cs231n/

Choice of distance metric

- Hyperparameter

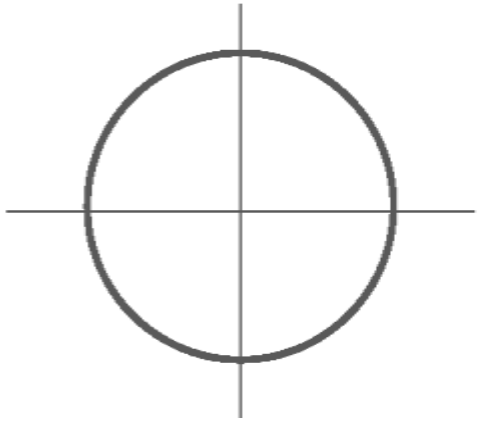
L1 (Manhattan) distance

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



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

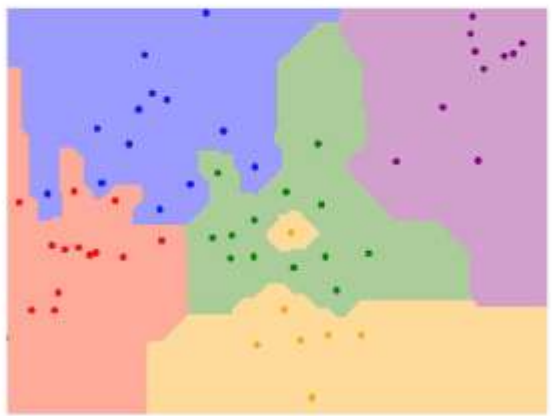
$$||x||_p = \left(|x_1|^p + \cdots + |x_n|^p\right)^{\frac{1}{p}} \qquad p \geq 1, x \in \mathbb{R}^n$$

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

K-Nearest Neighbors: Distance Metric

L1 (Manhattan) distance

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



K = 1

L2 (Euclidean) distance

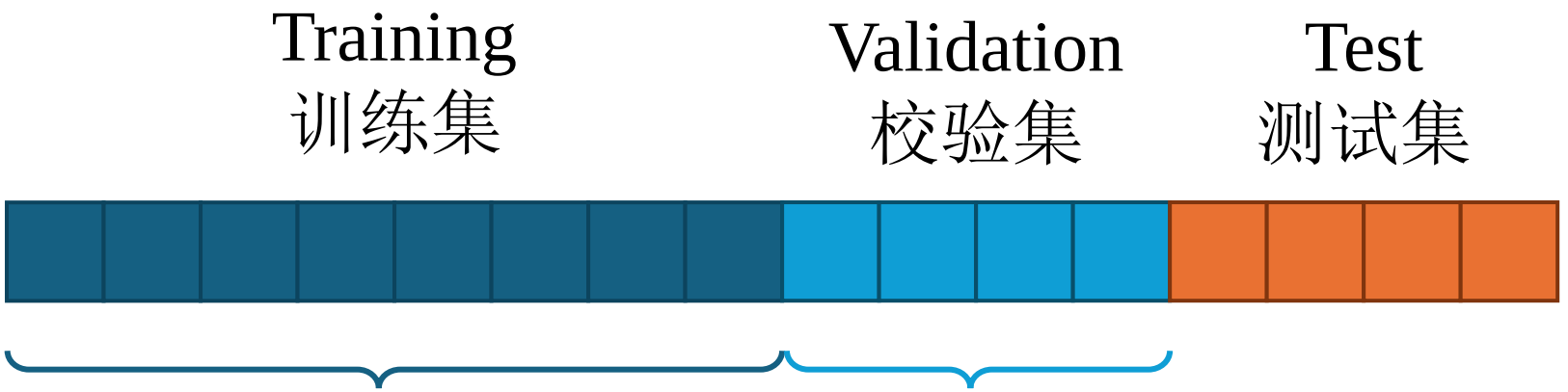
$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$

K = 1

Demo: http://vision.stanford.edu/teaching/cs231n-demos/knn/

K近邻

怎样确定距离度量? 怎样确定K?

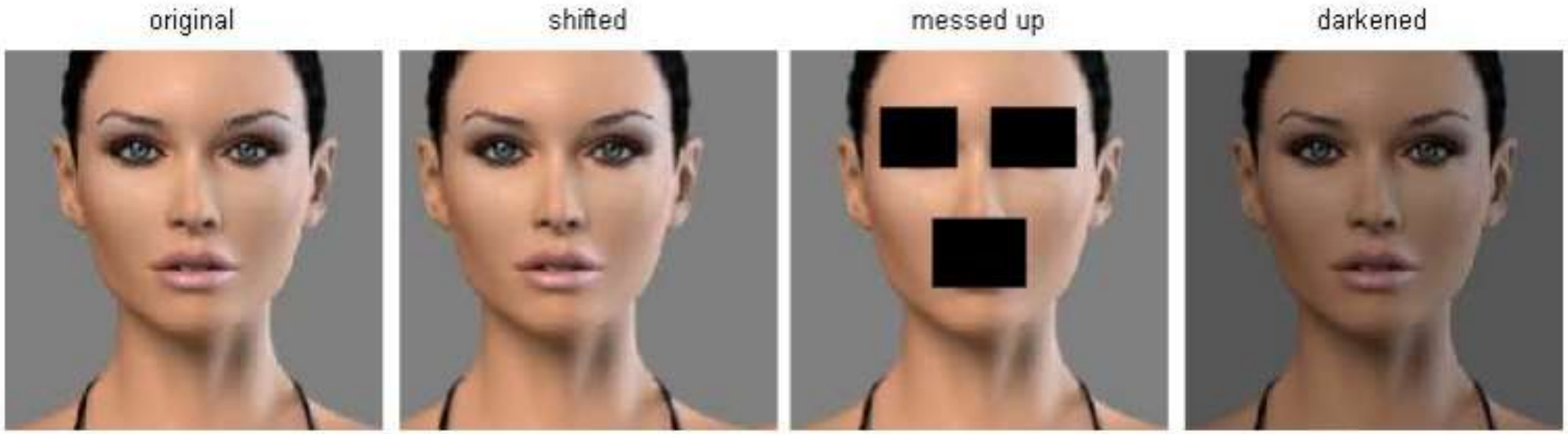


K近邻

- 虽然没有学习过程，但通常是有效的
  - 不学习参数和结构，没有实际的训练过程
- 对于每个任务都使用相同的算法
- 当数据点数量趋近于无穷时，错误率保证最多比数据上的最优解差2倍
  - **局部决策**：K-NN在决策时只考虑局部信息，即输入点的K个最近邻。因为它是基于局部信息做出决策的，所以当数据量很大时，它能够捕捉到数据的微小细节和模式。
  - **大数定律**：当训练数据量增加时，每个点的近邻都更加可能代表真实的数据分布。这意味着K-NN的决策边界变得更加准确。
  - 当K同时增长并且K/N趋近于0，其中N是数据点的数量
  - K-NN的错误率会收敛到贝叶斯最优错误率
  - **看似很好，但是有个严重的问题...**

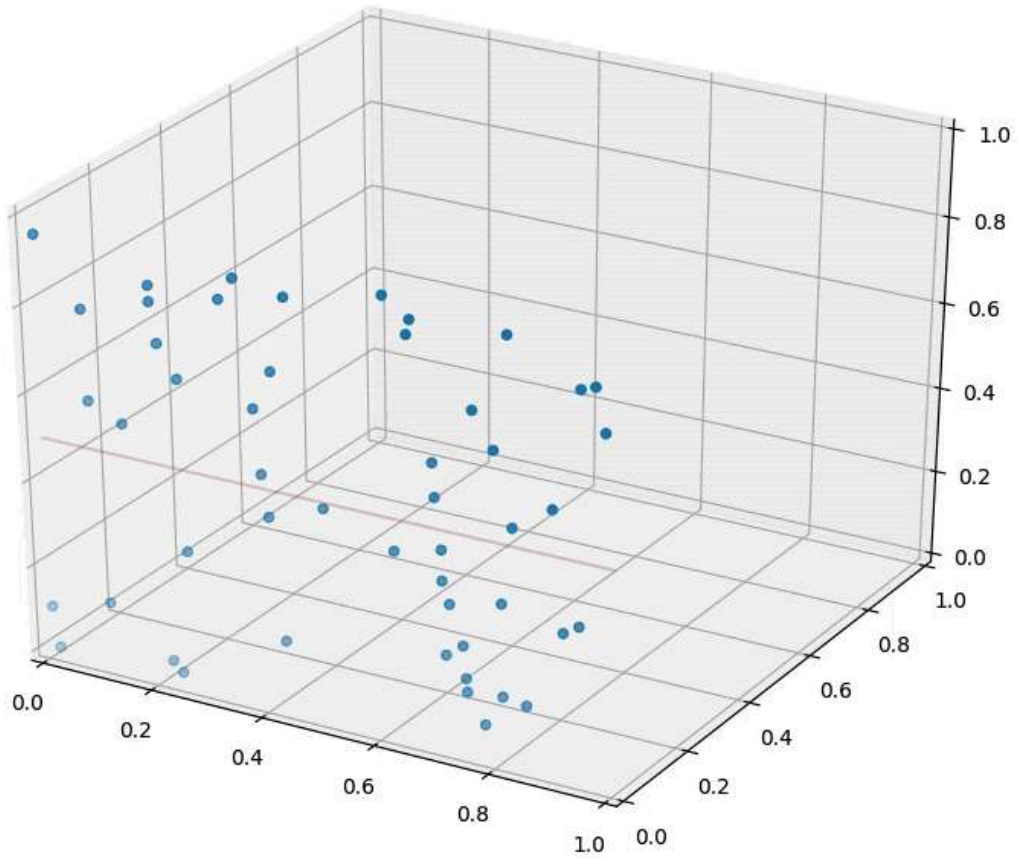
KNN的问题：距离度量

- 测试时性能糟糕
- 基于整体图像的距离度量可能非常不直观



KNN的问题：维度的诅咒

- 随着维度数量的增加，相同数量的数据变得更为稀疏。
  - 数据点之间距离变得均匀
- 我们所需的数据量随维度数量指数级增长。
  - 决策“边界”不稳定



Animation from https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote02_kNN.html

用最小二乘法解决分类问题

将分类视为回归问题: x_i 是图像特征; y_i 为 1 如果图像是猫, 为 0 如果图像不是猫. 最小化均方误差.

训练 $(\mathbf{x}_i, y_i)$: $$\arg\min_{\boldsymbol{w}} \sum_{i=1}^{n} \|\boldsymbol{w}^T \boldsymbol{x_i} - y_i\|^2$$

推断 (x): $$\boldsymbol{w}^T \boldsymbol{x} > t$$

Rifkin, Yeo, Poggio. *Regularized Least Squares Classification* (http://cbcl.mit.edu/publications/ps/rlsc.pdf). 2003

Redmon, Divvala, Girshick, Farhadi. *You Only Look Once: Unified, Real-Time Object Detection.* CVPR 2016.

线性模型

设定：三分类模型



模型：每个类别一个权重

$\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{w}_2$

$\boldsymbol{w}_0^T\boldsymbol{x}$ big if cat

$\boldsymbol{w}_1^T\boldsymbol{x}$ big if dog

$\boldsymbol{w}_2^T\boldsymbol{x}$ big if hippo

全部参数: $\boldsymbol{W_{3xF}}$ where **x** is in R^F

线性模型

Cat weight vector	0.2	-0.5	0.1	2.0	1.1	56
Dog weight vector	1.5	1.3	2.1	0.0	3.2	231
Hippo weight vector	0.0	0.3	0.2	-0.3	-1.2	24
						2
						1

W

x_i

-96.8	Cat score
437.9	Dog score
61.95	Hippo score

Wx_i

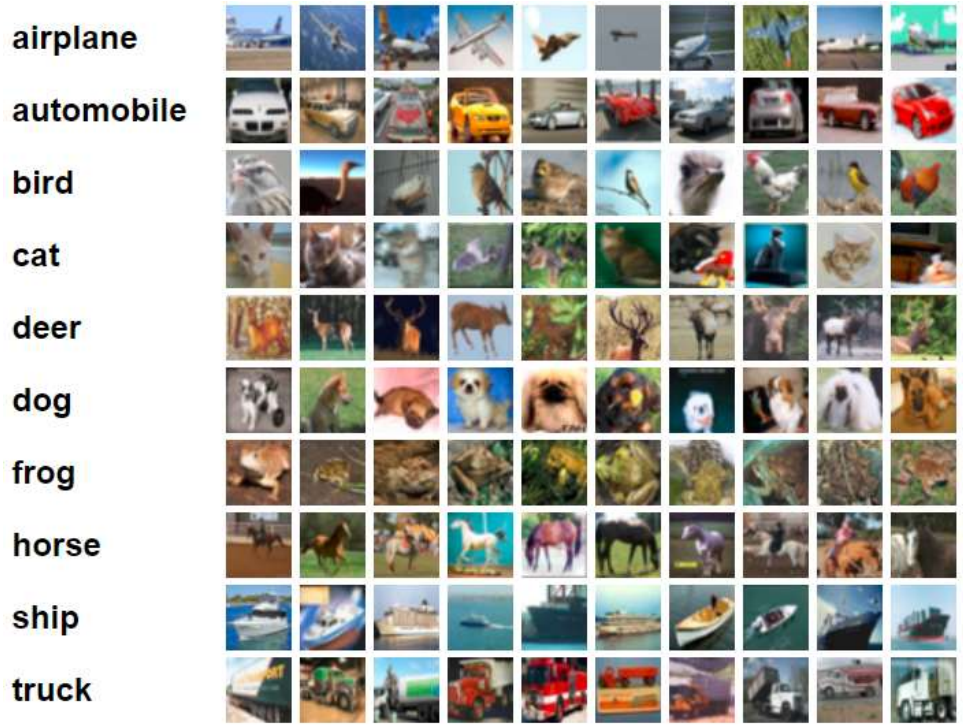
权重矩阵可以认为是“评分函数”

预测结果是一个向量，
代表每个类别的评分

Diagram by: Karpathy, Fei-Fei

视觉层面的直观理解

CIFAR 10:
32x32x3 Images, 10 Classes



- 把每张图像的像素平铺作为特征向量
- 拟合一个10分类的线性模型
- 根据$\mathbf{w}^T\mathbf{x}$做决策. 如果很大则该样本属于这一类.

Slide credit: Karpathy & Fei-Fei

猜类别？

根据$\mathbf{w}^T\mathbf{x}$做决策. 如果很大则该样本属于这一类.
让我们可视化$\mathbf{w}$

Deer or Plane?

Diagram credit: Karpathy & Fei-Fei

猜类别？

根据$\mathbf{w}^T\mathbf{x}$做决策. 如果很大则该样本属于这一类.
让我们可视化$\mathbf{w}$

Ship or Dog?

Diagram credit: Karpathy & Fei-Fei

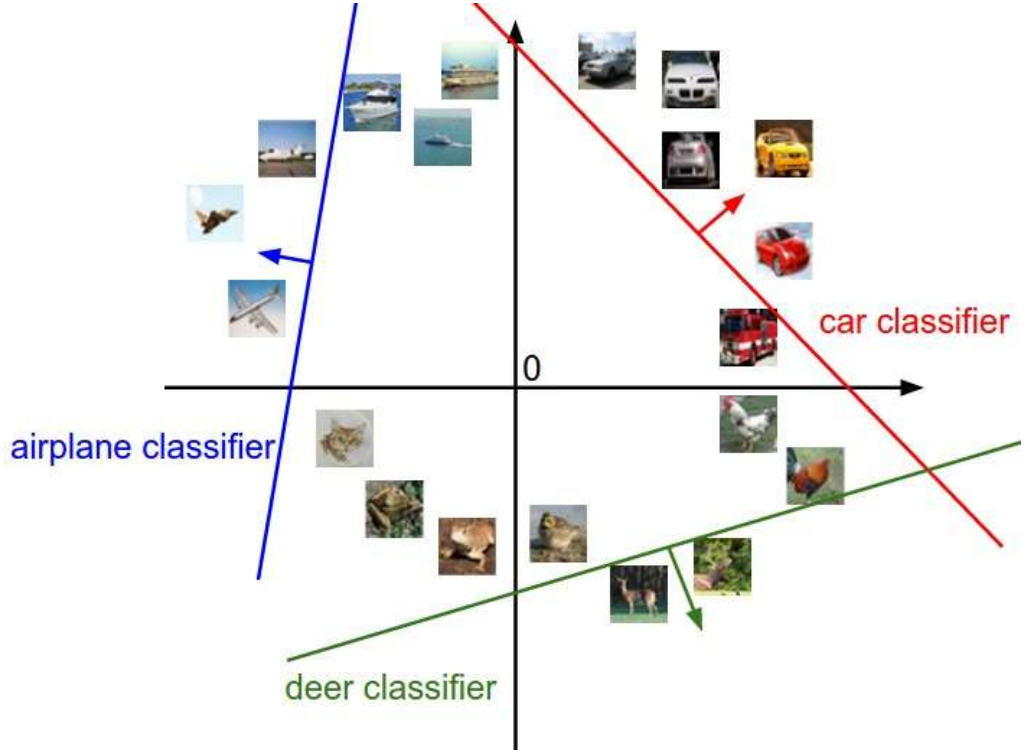
可视化线性分类器

根据$\mathbf{w}^T\mathbf{x}$做决策. 如果很大则该样本属于这一类.

Diagram credit: Karpathy & Fei-Fei

几何视角下的“直观理解”

线性分类器在2D中是什么样子的？



2D很适合做初步的直观感受，但机器学习通常涉及至少数十个，甚至数千个维度。关于空间和几何形状的直观感受，基于三维空间的认知在高维空间是完全错误的。不要信任那些向你展示2D图并写下“直观”字样的人 Charu, Hinneburg, Keim. ICDT 2001

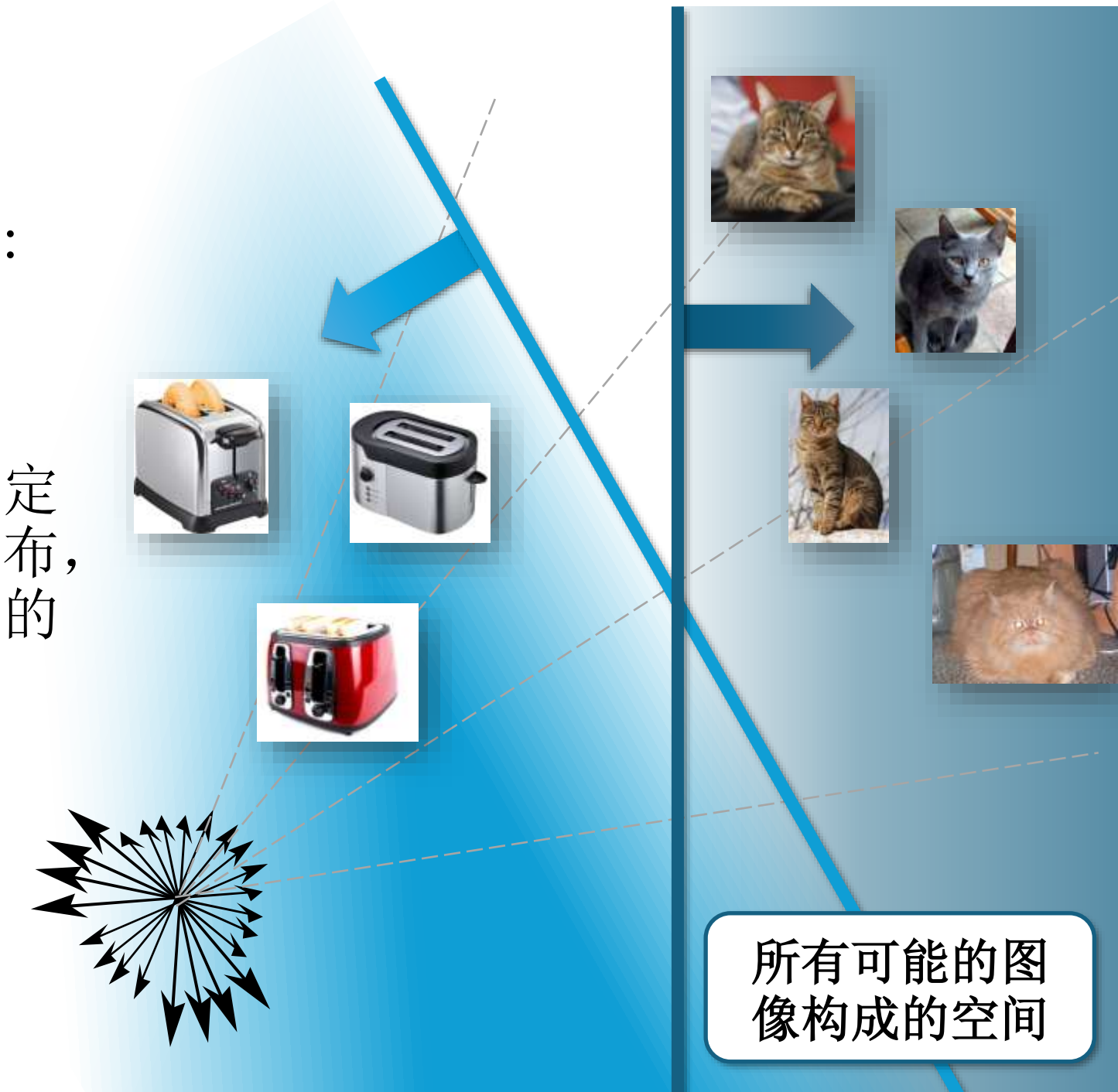
Diagram credit: Karpathy & Fei-Fei. 12-point font mini-rant: me

解释：几何角度

- 参数为每个类定义一个超平面:

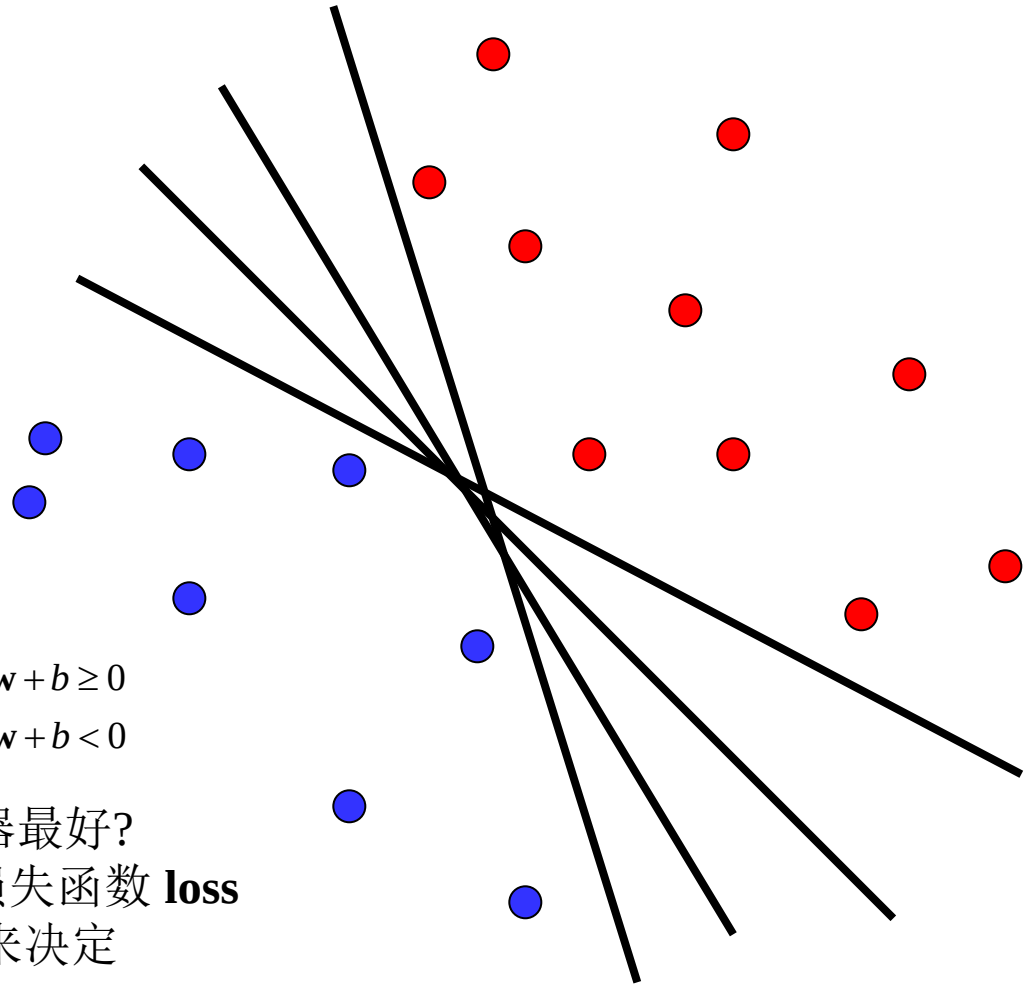
$$f(x_i, W, b) = Wx_i + b$$

- 我们可以将每个类的得分视为定义了一个与其距离成正比的分布，距离是指从对应的超平面到点的距离。



解释：几何角度

- 找到线性分类器（找平面）可以分开正负样本



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

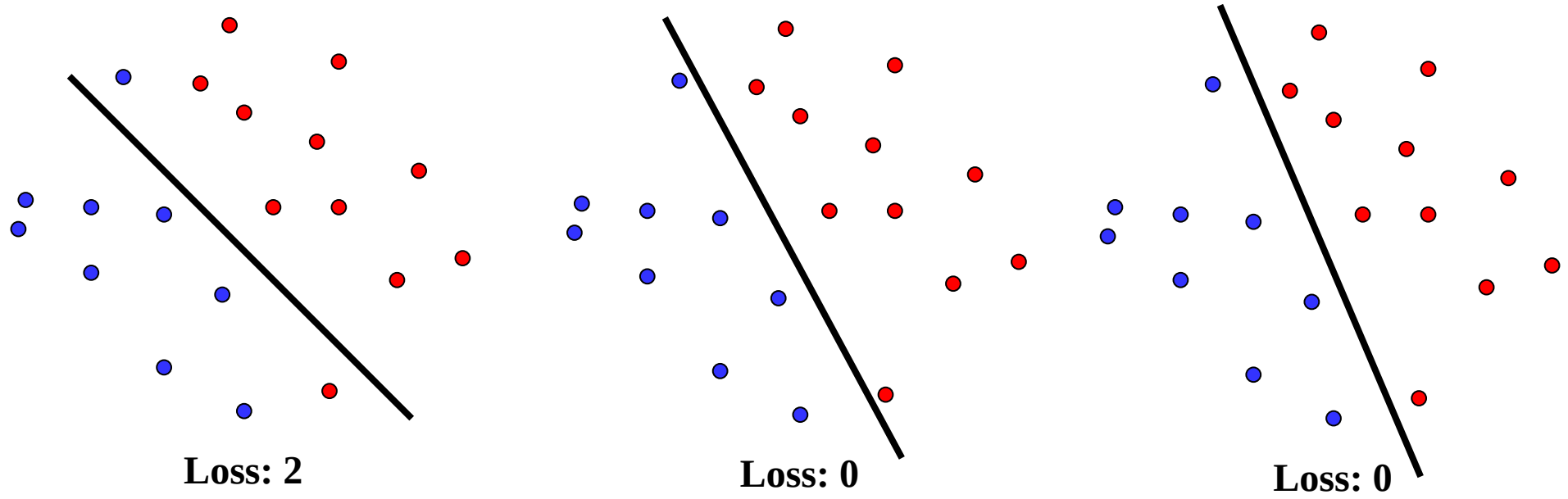
$\mathbf{x}_i$ negative： $\mathbf{x}_i \cdot \mathbf{w} + b < 0$

哪个分类器最好?

我们需要一个损失函数 **loss function** 来决定

什么是好的损失函数？

- 检视分类错的样本
  - 问题：样本是离散的，我们没办法区分相似的分类器
  - 我们需要更好的 **泛化性** *generalization*
  - 还要处理多于两类的情况



如何设计损失函数 —— 最大间距

这样有什么问题？

Loss: dog score – cat score

怎么定义“dog” vs “cat”的评分?

$$(\boldsymbol{Wx})_2 - (\boldsymbol{Wx})_1$$

避免优化到负无穷

$$\max(0, (\boldsymbol{Wx})_2 - (\boldsymbol{Wx})_1)$$

$(\boldsymbol{Wx})_1$ -96.8 Cat score

$(\boldsymbol{Wx})_2$ 437.9 Dog score

$(\boldsymbol{Wx})_3$ 61.95 Hippo score

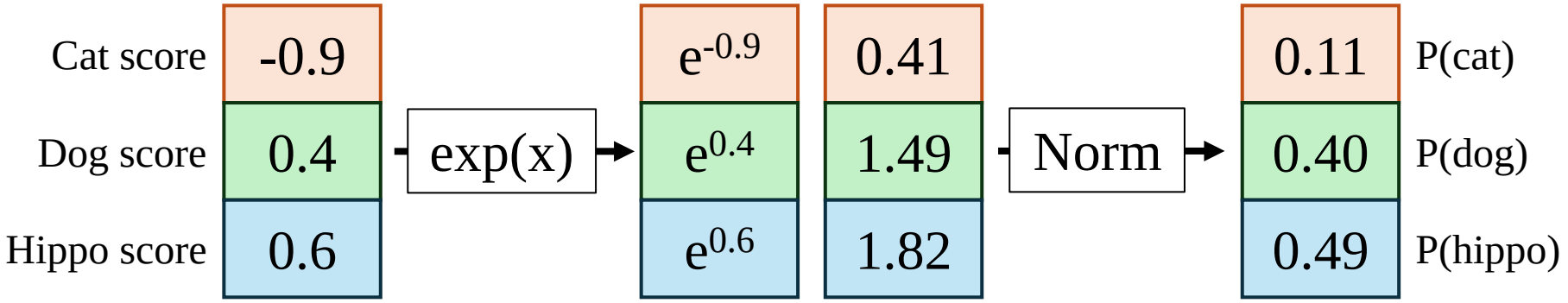
$$\boldsymbol{Wx}$$

预测结果是一个向量，
第j个值代表j类别的评分

Diagram by: Karpathy, Fei-Fei

Softmax

把评分转化为“概率分布”



$\sum$=3.72

P(class j): $\frac{\exp((Wx)_j)}{\sum_k \exp((Wx)_k)}$

Softmax

推断时 (x): $\arg\max_k \ (\boldsymbol{W}\boldsymbol{x})_k$ (找到评分最高的那一类)

P(class j): $\dfrac{\exp((Wx)_j)}{\sum_k \exp((Wx)_k)}$

为什么我们在测试时可以省略 exp/sum exp ?

Softmax

推断时 (x): $\arg\max_k (\boldsymbol{W}\boldsymbol{x})_k$ (找到评分最高的那一类)

训练时 $(\mathbf{x}_i, y_i)$:

$$\arg\min_{\boldsymbol{W}} \boldsymbol{\lambda}\|\boldsymbol{W}\|_2^2 + \sum_i^n -\log\left(\frac{\exp((Wx)_{y_i})}{\sum_k \exp((Wx)_k)}\right)$$

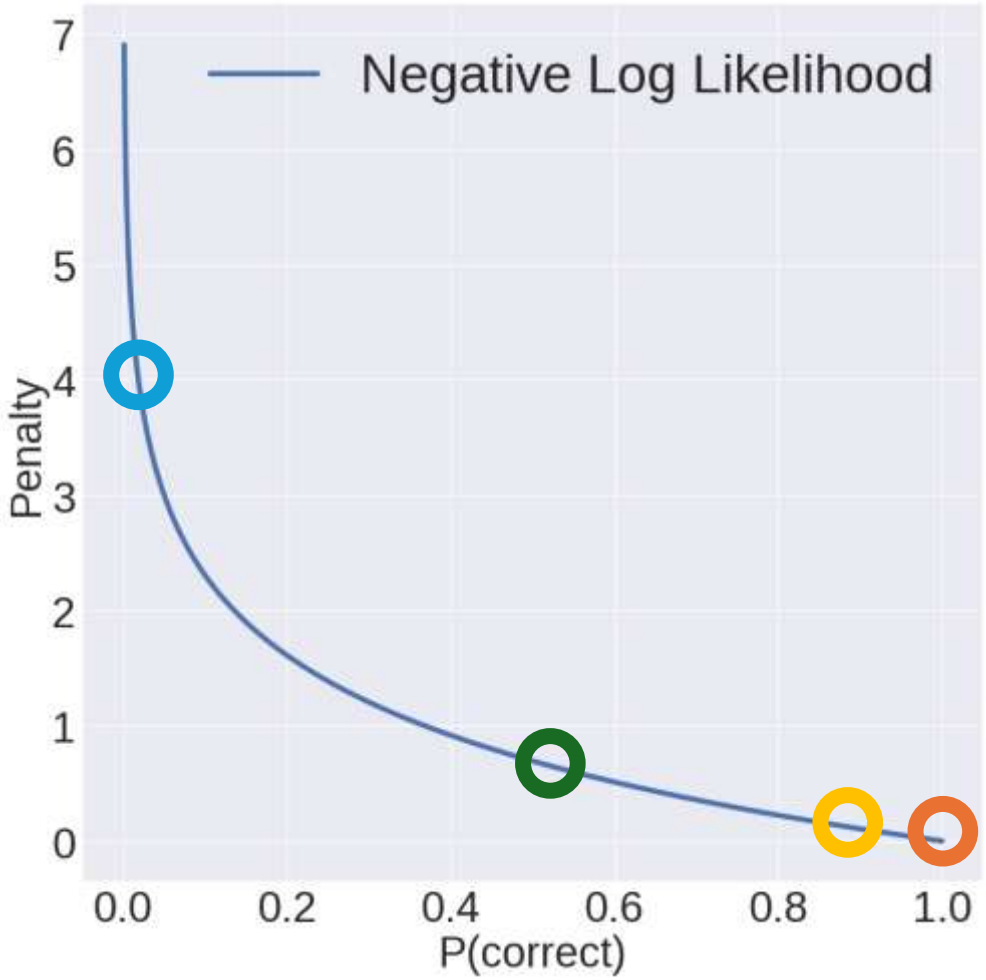
P(correct class)

Regularization
正则项

对所有训练样本

对正确分类对应的 negative log-likelihood 进行损失优化

Softmax的优点



P(correct) = 0.05:
3.0 penalty

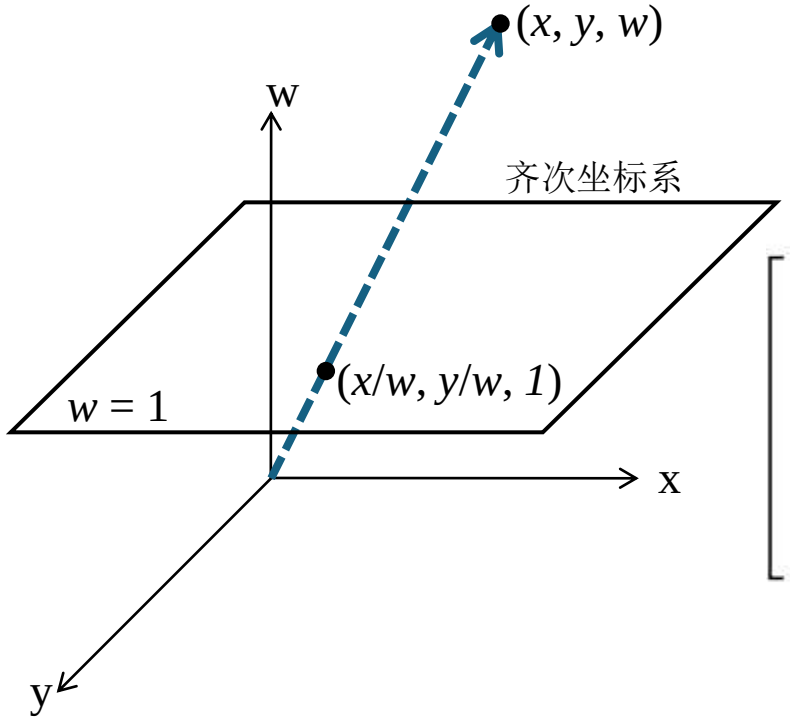
P(correct) = 0.5:
0.11 penalty

P(correct) = 0.9:
0.11 penalty

P(correct) = 1:
No penalty!

回顾——变换回归、图像分类

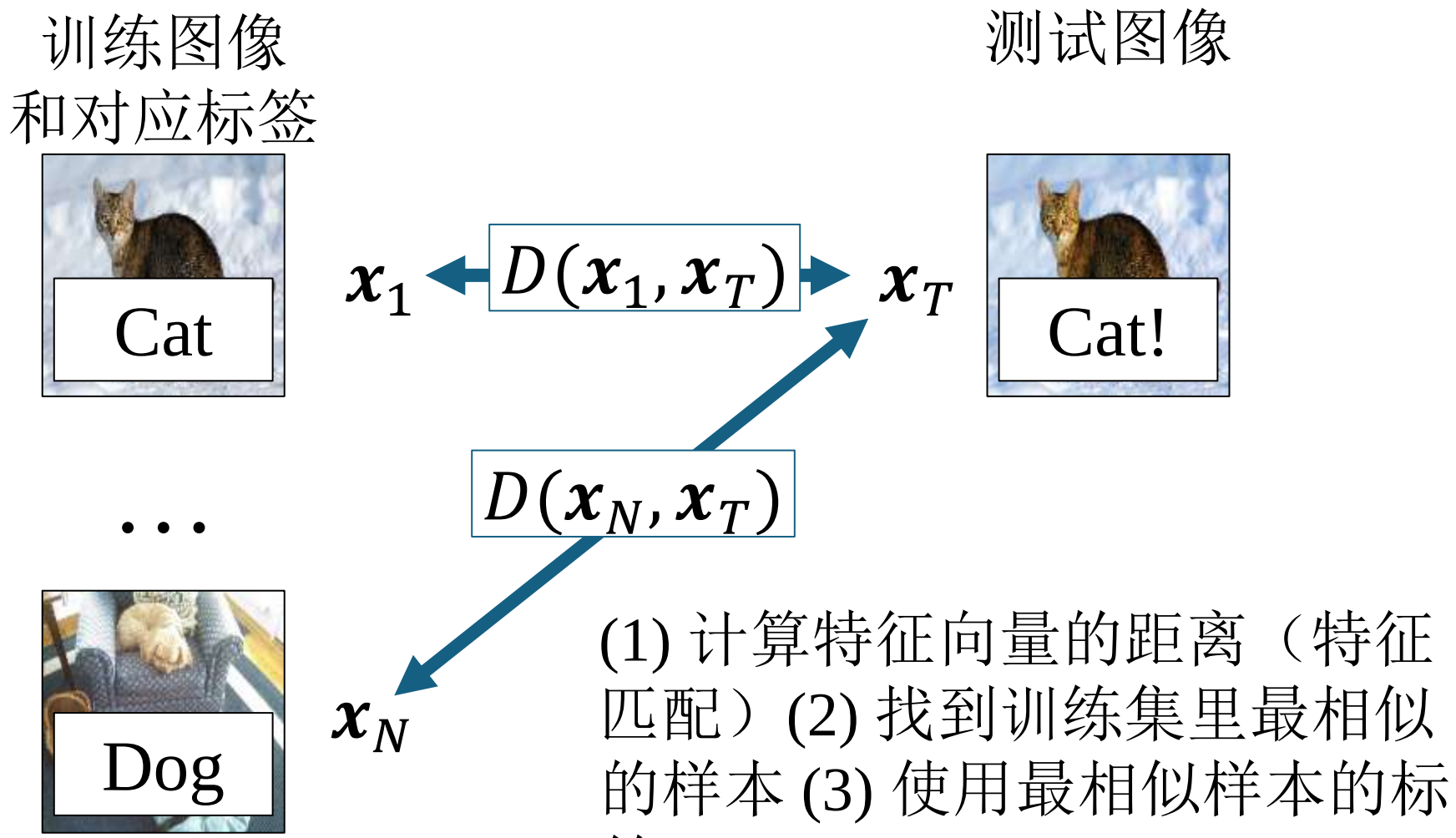
求解同构映射



$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ & & & & \vdots & & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_nx_n & -y'_ny_n & -y'_n \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

$$\underset{2n \times 9}{\mathbf{A}} \qquad \underset{9}{\mathbf{h}} \qquad \underset{2n}{\mathbf{0}}$$

分类最简单的形式：最近邻

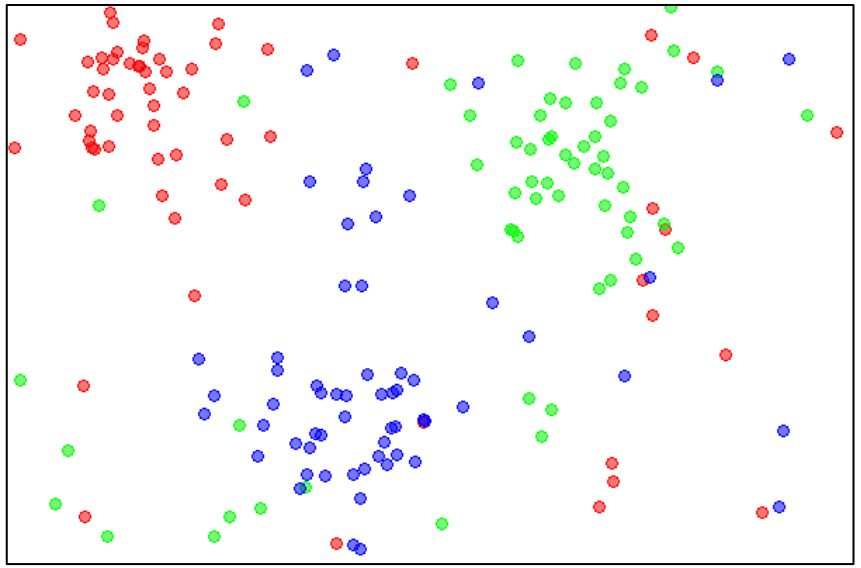


(1) 计算特征向量的距离（特征匹配）(2) 找到训练集里最相似的样本 (3) 使用最相似样本的标签.

K近邻

找到前K近邻样本，然后投票决定标签

2D Datapoints
(colors = labels)



2D Predictions
(colors = labels)

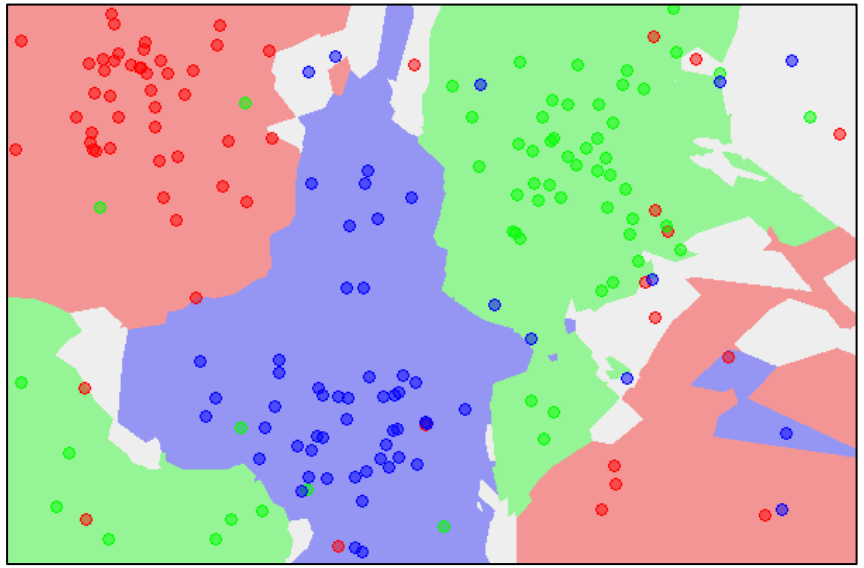


Diagram Credit: Wikipedia

K近邻

- 虽然没有学习过程，但通常是有效的
  - 不学习参数和结构，没有实际的训练过程
- 对于每个任务都使用相同的算法
- 当数据点数量趋近于无穷时，错误率保证最多比数据上的最优解差2倍
  - **局部决策**：K-NN在决策时只考虑局部信息，即输入点的K个最近邻。因为它是基于局部信息做出决策的，所以当数据量很大时，它能够捕捉到数据的微小细节和模式。
  - **大数定律**：当训练数据量增加时，每个点的近邻都更加可能代表真实的数据分布。这意味着K-NN的决策边界变得更加准确。
  - 当K同时增长并且K/N趋近于0，其中N是数据点的数量
  - K-NN的错误率会收敛到贝叶斯最优错误率
  - **看似很好，但是有效率问题，且性能一般**

线性模型

设定：三分类模型



模型：每个类别一个权重

$\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{w}_2$

$\boldsymbol{w}_0^T\boldsymbol{x}$ big if cat

$\boldsymbol{w}_1^T\boldsymbol{x}$ big if dog

$\boldsymbol{w}_2^T\boldsymbol{x}$ big if hippo

全部参数: $\boldsymbol{W_{3xF}}$ where **x** is in R^F

可视化线性分类器

根据$\mathbf{w}^T\mathbf{x}$做决策. 如果很大则该样本属于这一类.

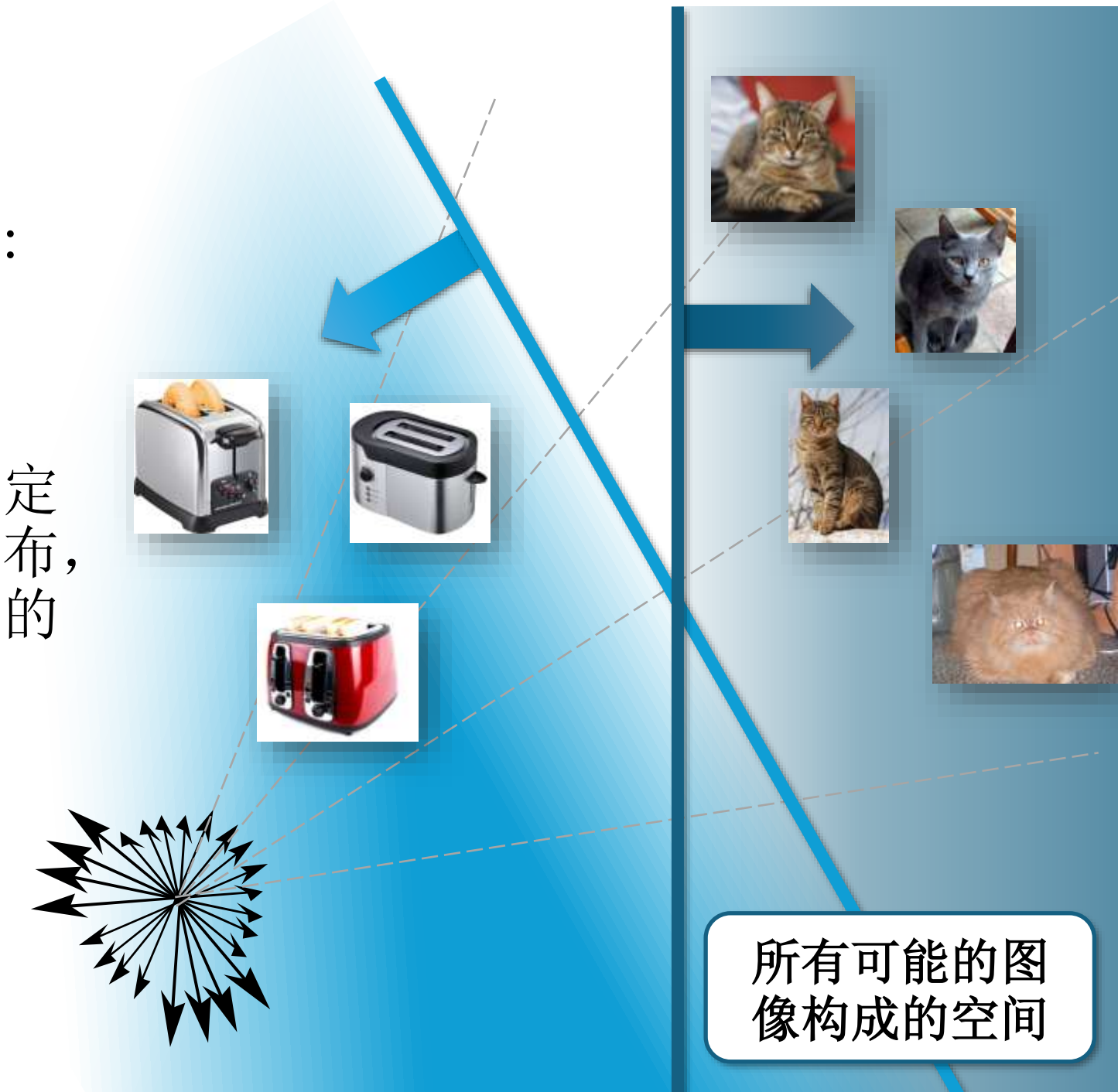
Diagram credit: Karpathy & Fei-Fei

解释：几何角度

- 参数为每个类定义一个超平面:

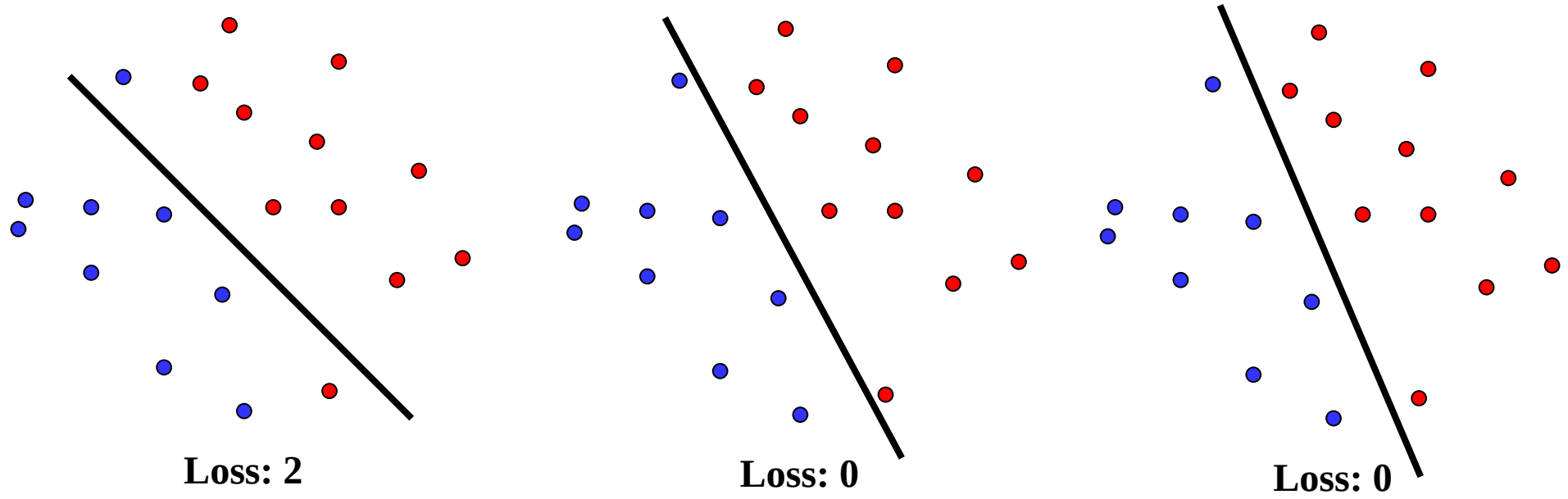
$$f(x_i, W, b) = Wx_i + b$$

- 我们可以将每个类的得分视为定义了一个与其距离成正比的分布，距离是指从对应的超平面到点的距离。



什么是好的损失函数？

- 检视分类错的样本
  - 问题：样本是离散的，我们没办法区分相似的分类器
  - 我们需要更好的 **泛化性** *generalization*
  - 还要处理多于两类的情况



如何设计损失函数 —— 最大间距

这样有什么问题？

Loss: dog score – cat score

怎么定义“dog” vs “cat”的评分?

$$(\boldsymbol{Wx})_2 - (\boldsymbol{Wx})_1$$

避免优化到负无穷

$$\max(0, (\boldsymbol{Wx})_2 - (\boldsymbol{Wx})_1)$$

$(\boldsymbol{Wx})_1$ -96.8 Cat score

$(\boldsymbol{Wx})_2$ 437.9 Dog score

$(\boldsymbol{Wx})_3$ 61.95 Hippo score

$$\boldsymbol{Wx}$$

预测结果是一个向量，
第j个值代表j类别的评分

Diagram by: Karpathy, Fei-Fei

Softmax

推断时 (x): $\arg\max_k (\boldsymbol{W}\boldsymbol{x})_k$ (找到评分最高的那一类)

训练时 $(\mathbf{x}_i, y_i)$:

$$\arg\min_{\boldsymbol{W}} \boldsymbol{\lambda}\|\boldsymbol{W}\|_2^2 + \sum_i^n -\log\left(\frac{\exp((Wx)_{y_i})}{\sum_k \exp((Wx)_k)}\right)$$

P(correct class)

Regularization
正则项

对所有训练样本

对正确分类对应的 negative log-likelihood 进行损失优化

Cross-entropy loss 交叉熵

$$f(x_i, W) = Wx_i \quad \text{(score function)}$$

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right) \qquad L_i = -f_{y_i} + \log\sum_j e^{f_j}$$

我们称 L_i 为 交叉熵*cross-entropy loss*

$P(y_i \mid x_i; W)$

i.e. we're minimizing the negative log likelihood.

损失函数

- 交叉熵损失仅仅是一种可能的损失函数。
  - 它的一个好属性是它将得分重新解释为概率，这些概率具有自然的含义。

- 支持向量机（SVM，最大间距）损失函数过去也很受欢迎。
  - 但目前，交叉熵是最常见的分类损失。

怎么优化线性模型？

目标：找到参数$\mathbf{w}$ 最小化损失函数 L.

$$\arg\min_{\boldsymbol{w}\in R^N} L(\boldsymbol{w})$$

不同的损失函数 L:

$$L(\boldsymbol{W}) = \boldsymbol{\lambda}\|\boldsymbol{W}\|_{\boldsymbol{2}}^{\boldsymbol{2}} + \sum_{i=1}^{n} -\log\left(\frac{\exp((Wx)_{y_i})}{\sum_k \exp((Wx)_k))}\right)$$

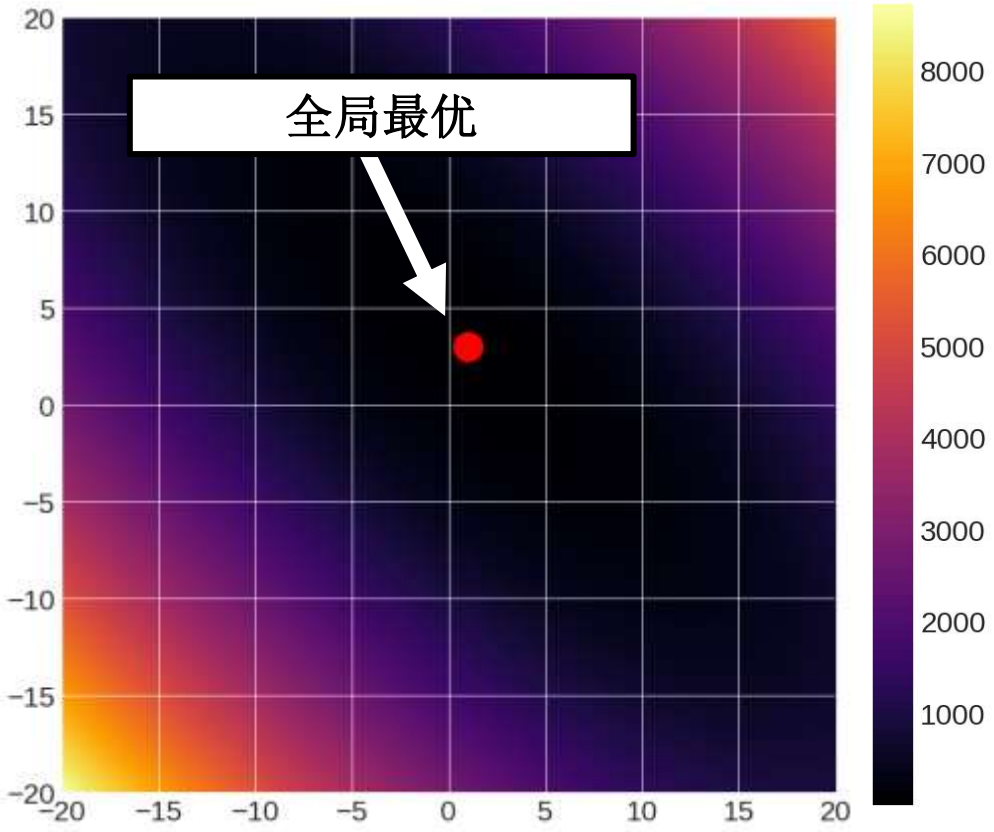
$$L(\boldsymbol{w}) = \lambda\|\boldsymbol{w}\|_2^2 + \sum_{i=1}^{n}\left(y_i - \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}_{\boldsymbol{i}}\right)^2$$

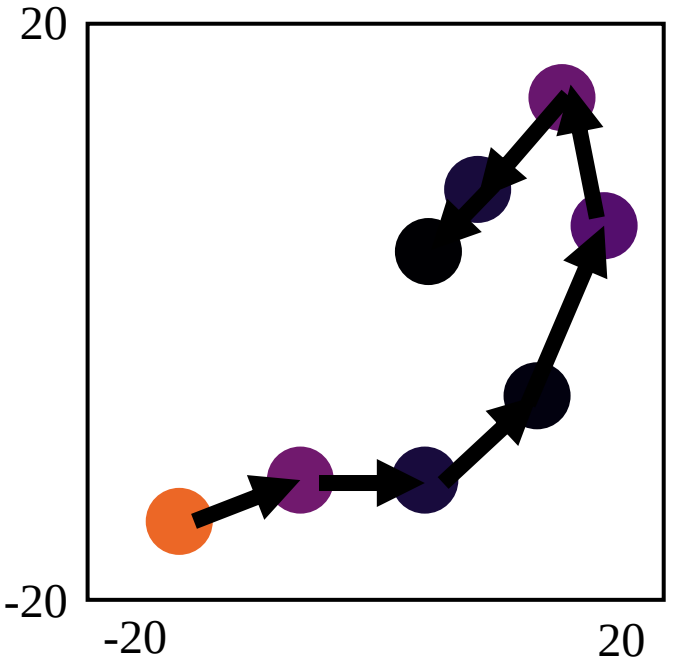
$$L(\boldsymbol{w}) = C\|\boldsymbol{w}\|_2^2 + \sum_{i=1}^{n}\max(0,1 - y_i\boldsymbol{w}^T\boldsymbol{x}_{\boldsymbol{i}})$$

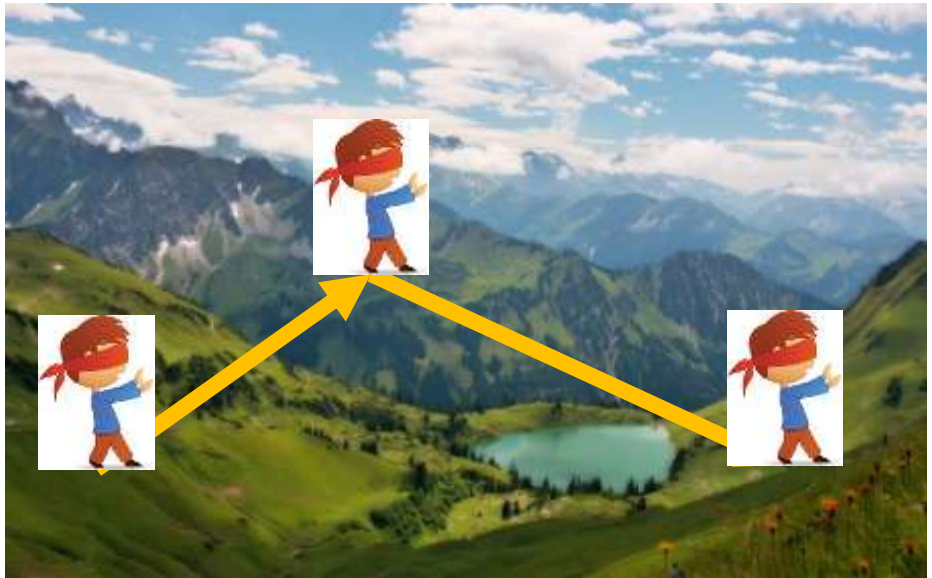
怎么优化？

- 每个点代表函数的值
  （大小参照右边热度图）
- 类似“爬山”与“下山”

$$f(x,y) = (x+2y-7)^2 + (2x+y-5)^2$$

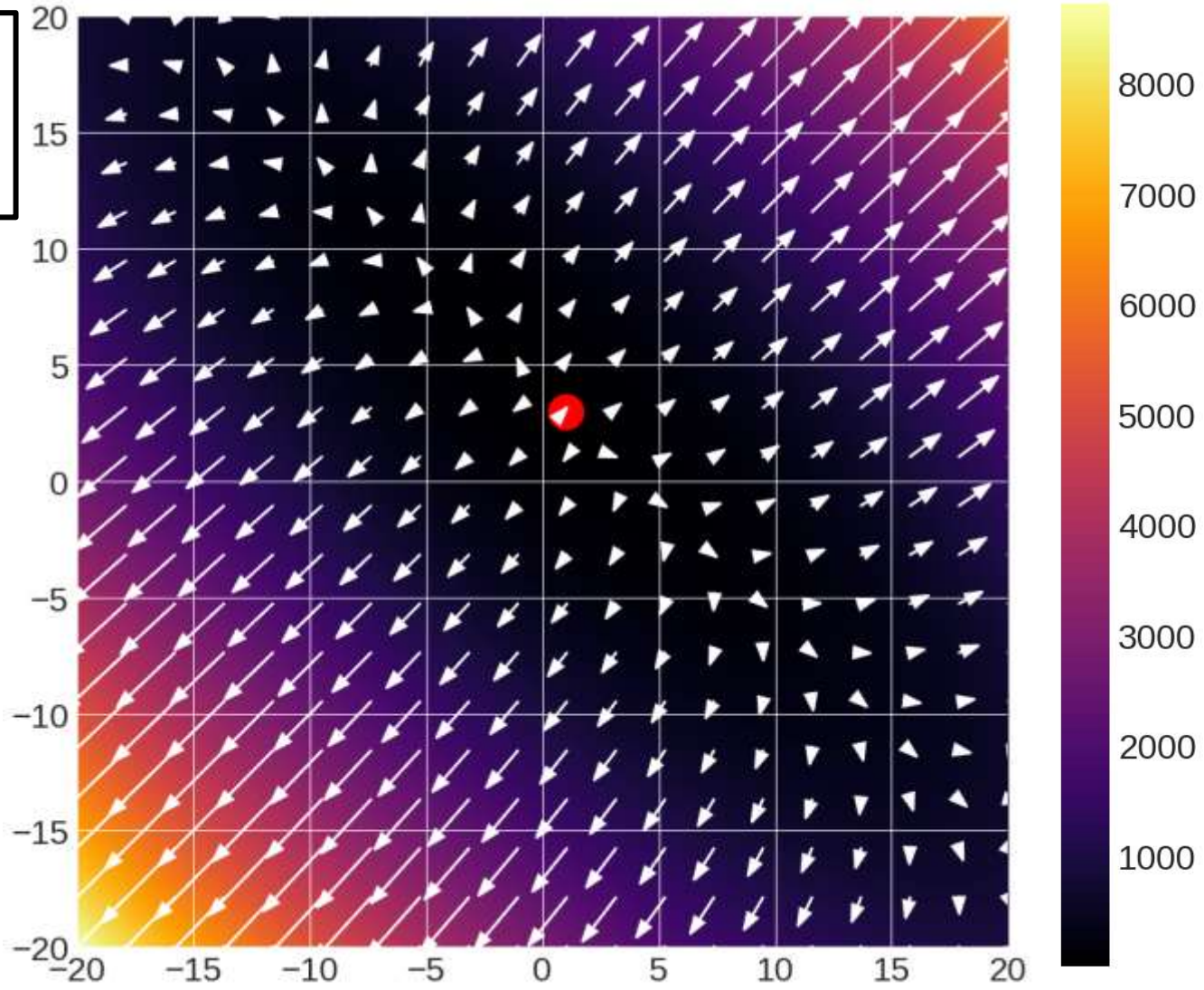






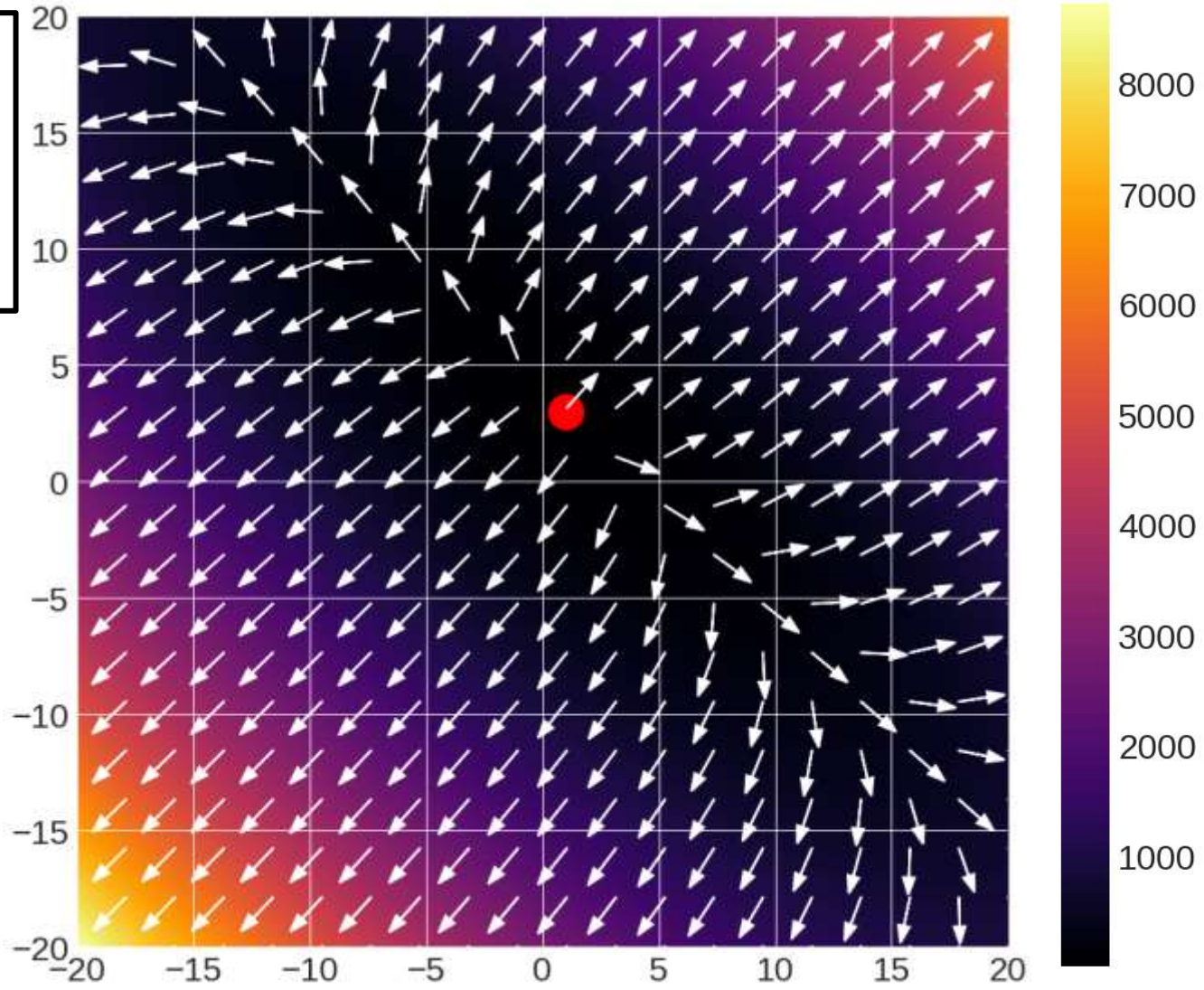
怎么优化？使用梯度

箭头:
梯度 **gradient**



怎么优化？使用梯度

箭头:
梯度方向
(单位长度)



怎么优化？使用梯度

目标: $\arg\min_{\boldsymbol{w}} L(\boldsymbol{w})$

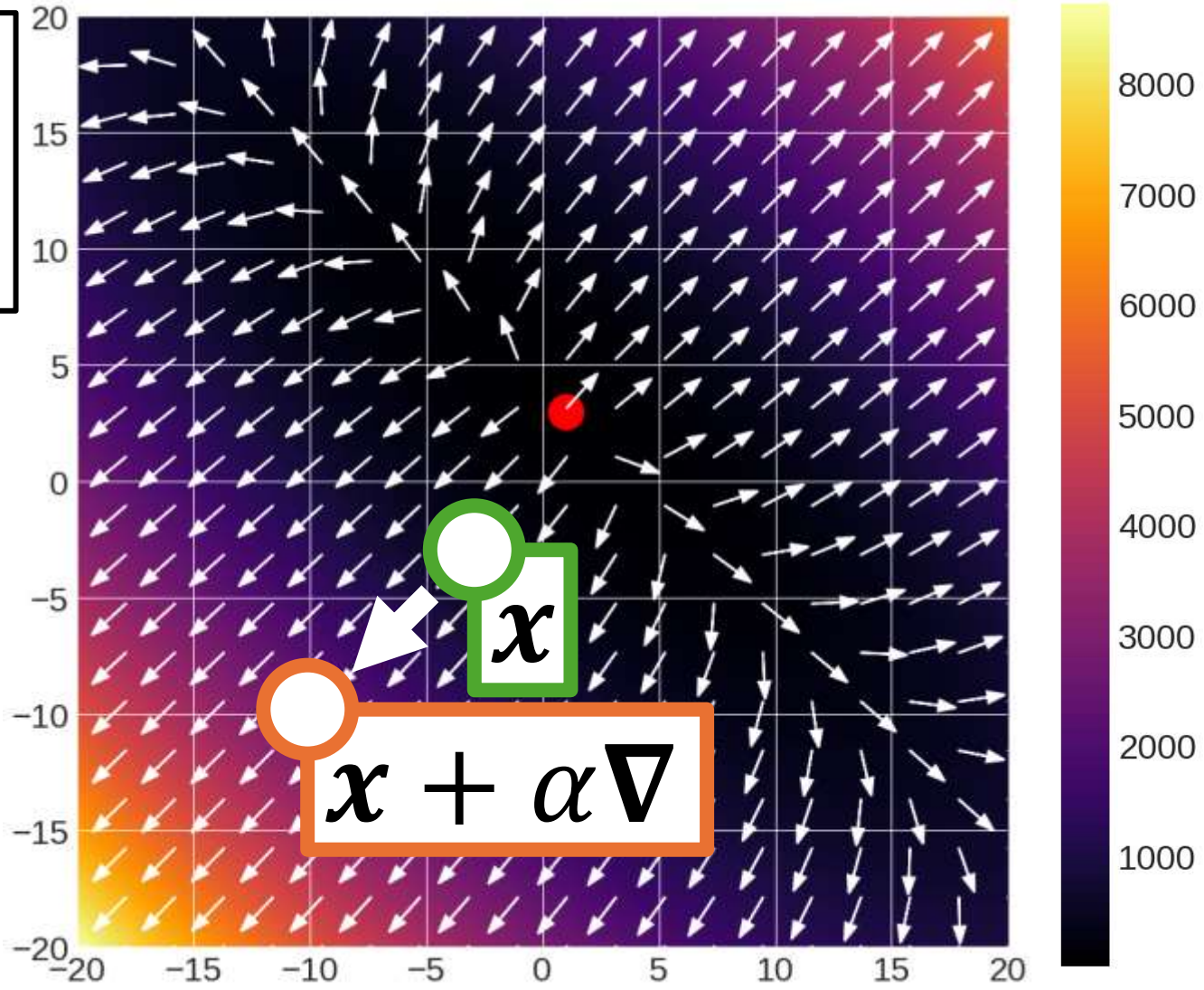
计算梯度: $\nabla_{\boldsymbol{w}} L(\boldsymbol{w}) = \begin{bmatrix} \partial L/\partial \boldsymbol{x}_1 \\ \vdots \\ \partial L/\partial \boldsymbol{x}_N \end{bmatrix}$

哪个更大(当α较小时)?

$$L(\boldsymbol{w}) \quad \begin{matrix} \leq ? \\ > ? \end{matrix} \quad L(\boldsymbol{w} + \alpha \nabla_{\boldsymbol{w}} L(\boldsymbol{w}))$$

怎么优化？使用梯度

箭头:
梯度方向
(单位长度)



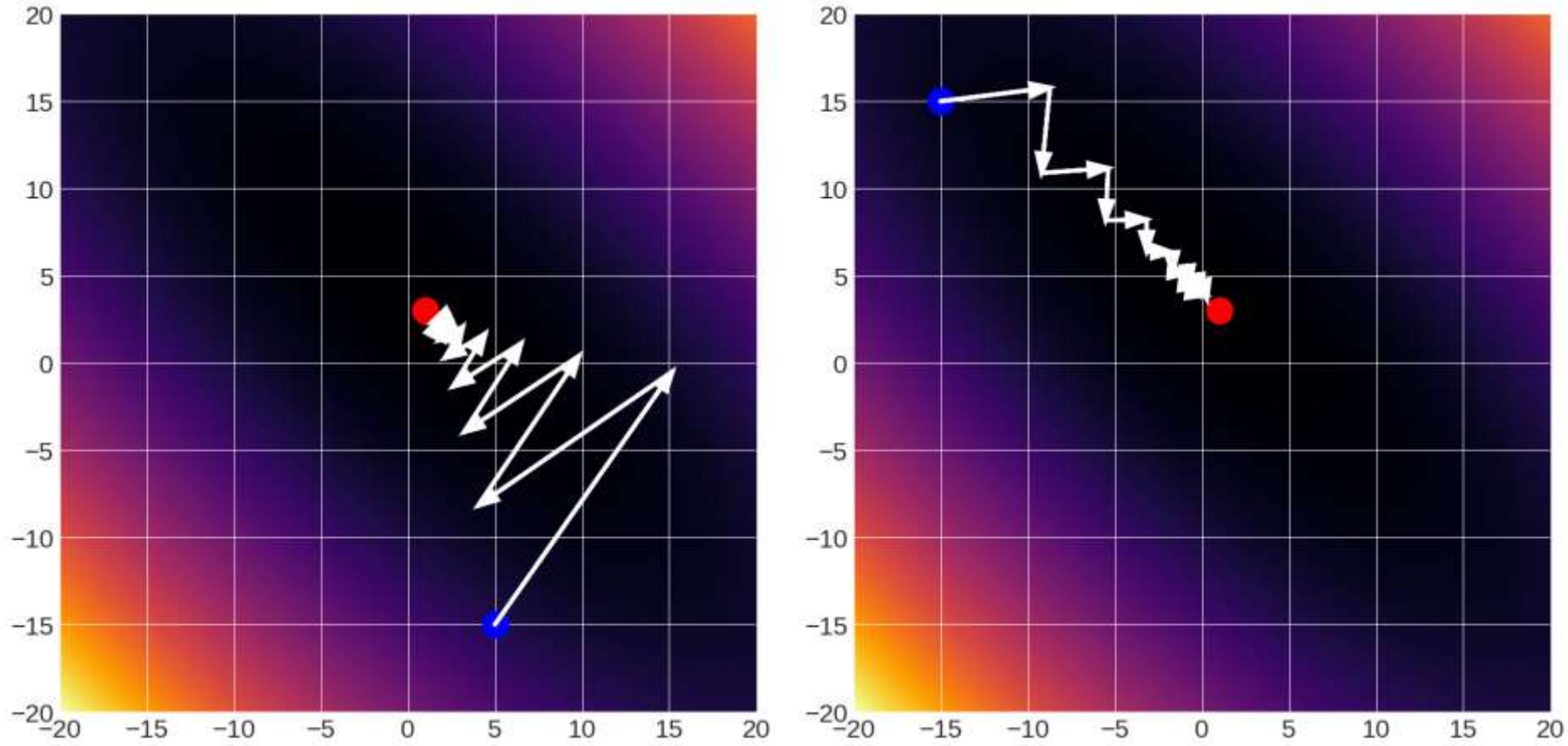
梯度下降

每一步优化，向梯度的反方向移动

w0 = *initialize***()** #initialize
for iter in range(numIters):
    g = $\nabla_{\boldsymbol{w}} L(\boldsymbol{w})$ #eval gradient
    w = **w** + **-***stepsize*(iter)***g** #update w
return **w**

梯度下降

基于蓝色起始点，$w_{i+1} = w_i + -9.8x10^{-2} \times gradient$

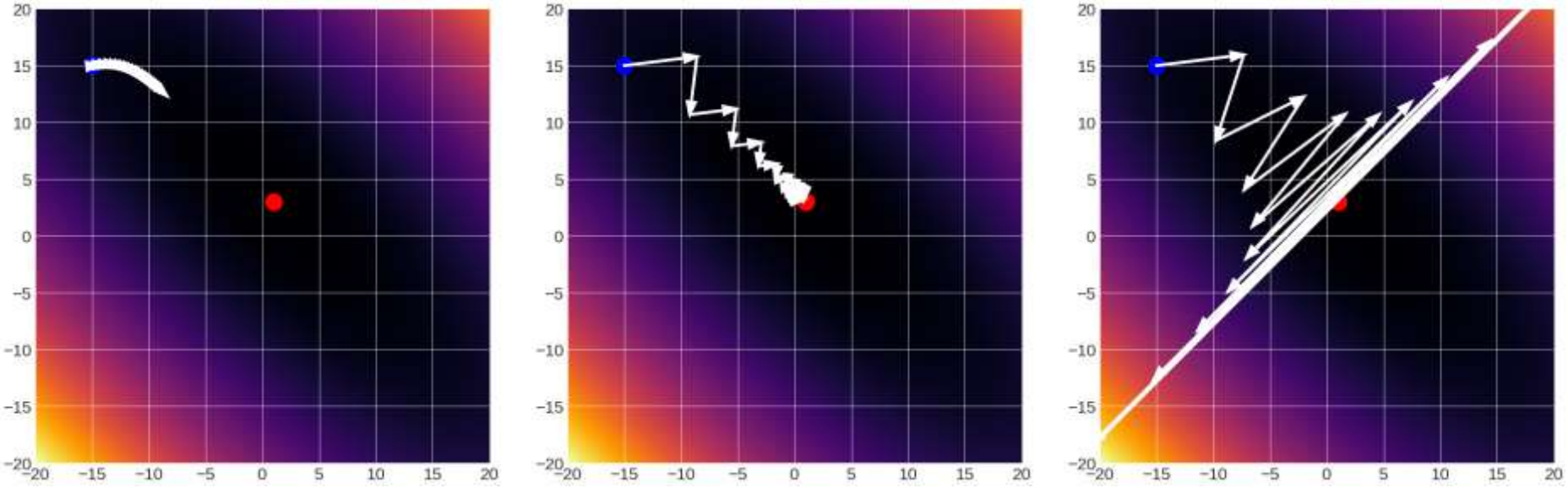


梯度下降

步长 Step size (也叫做 学习率**learning rate / lr**)

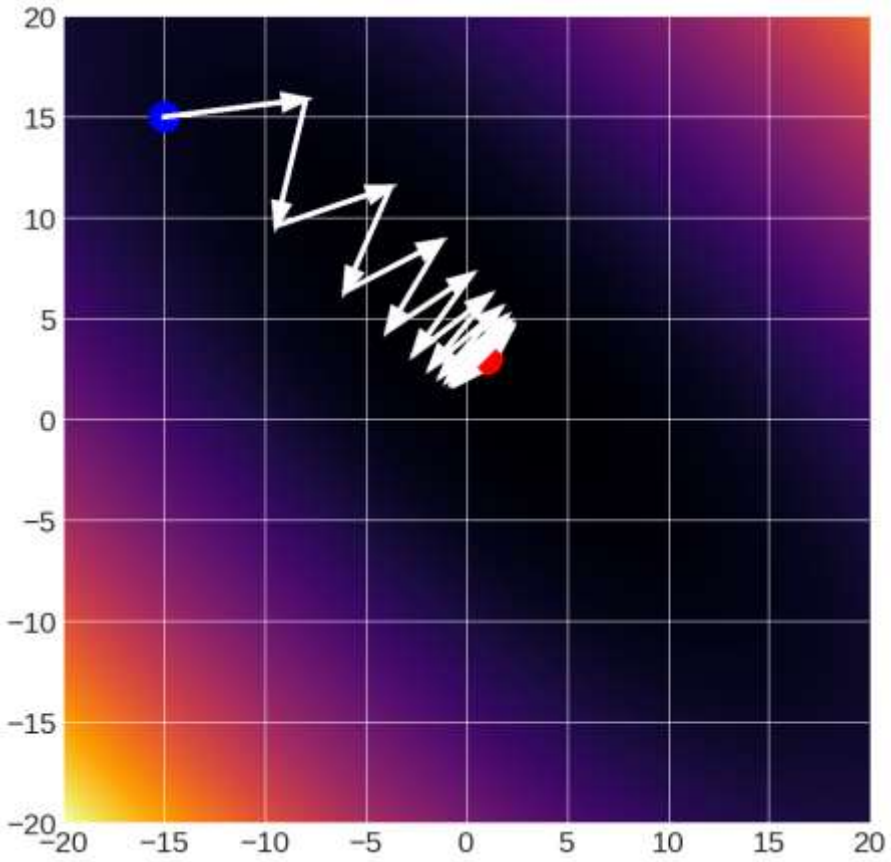
十分重要的参数

$1x10^{-2}$	$10x10^{-2}$	$12x10^{-2}$
falls short	converges	diverges



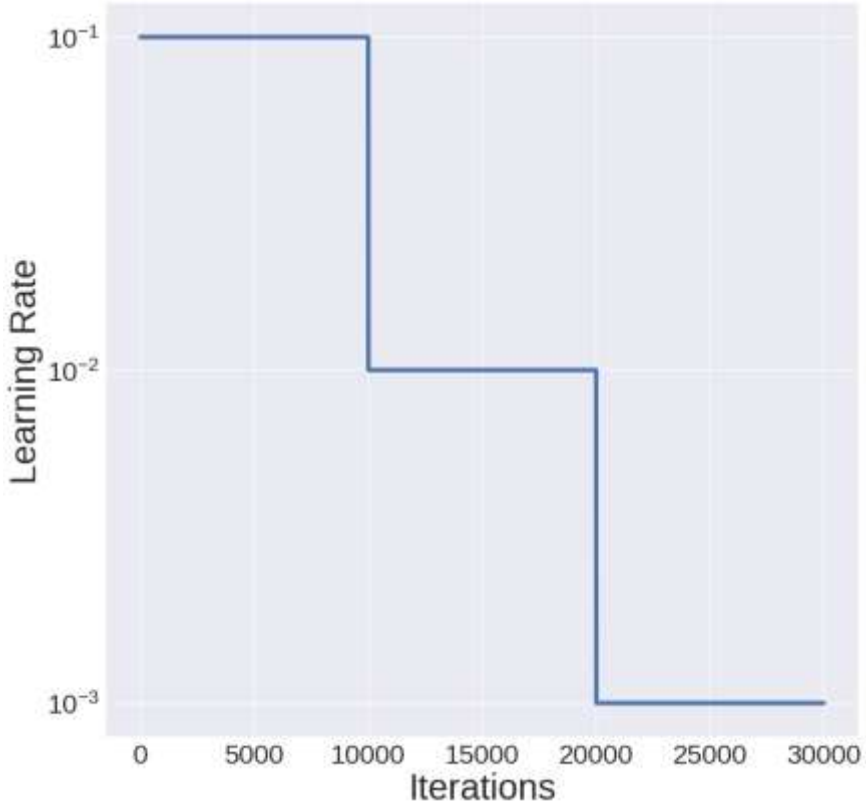
梯度下降

$11x10^{-2}$:oscillates
(Raw gradients)



梯度下降

使用初始的 lr, 每N步以后乘以f



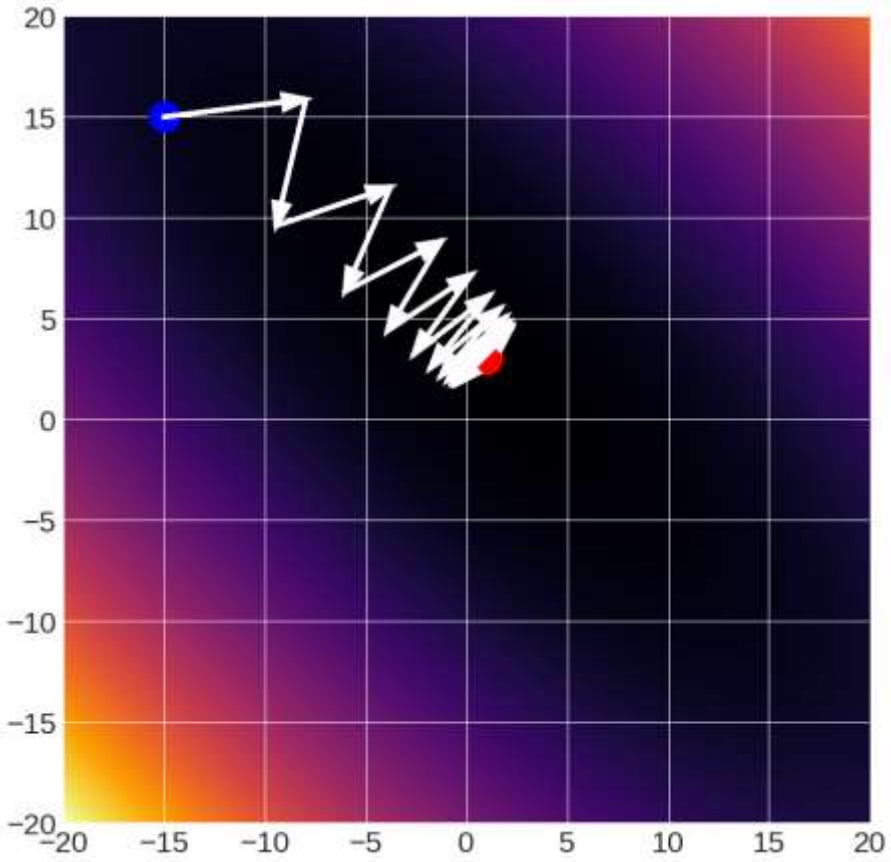
init_lr = 10^{-1}

f = 0.1

N = 10K

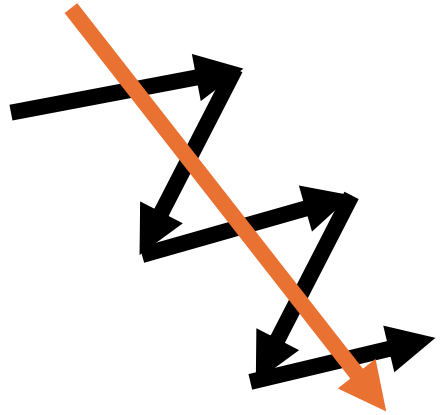
梯度下降

$11x10^{-2}$:oscillates
(Raw gradients)



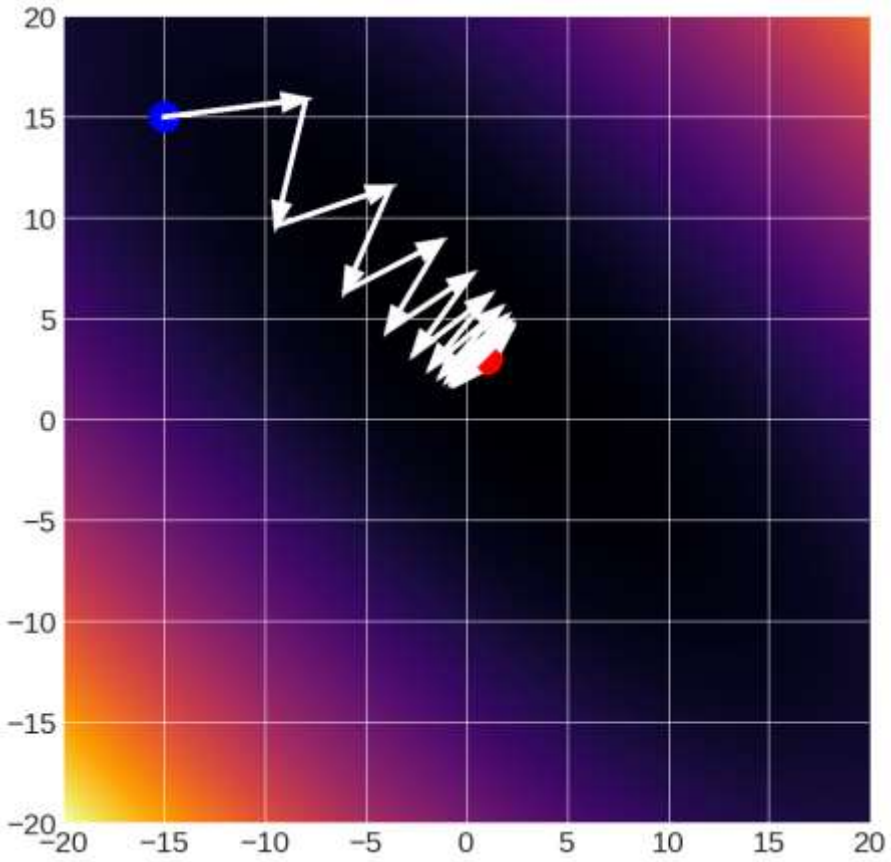
“平均梯度”

记录历史梯度，并且与当前步梯度取加权平均， 也被称为动量法 “momentum”

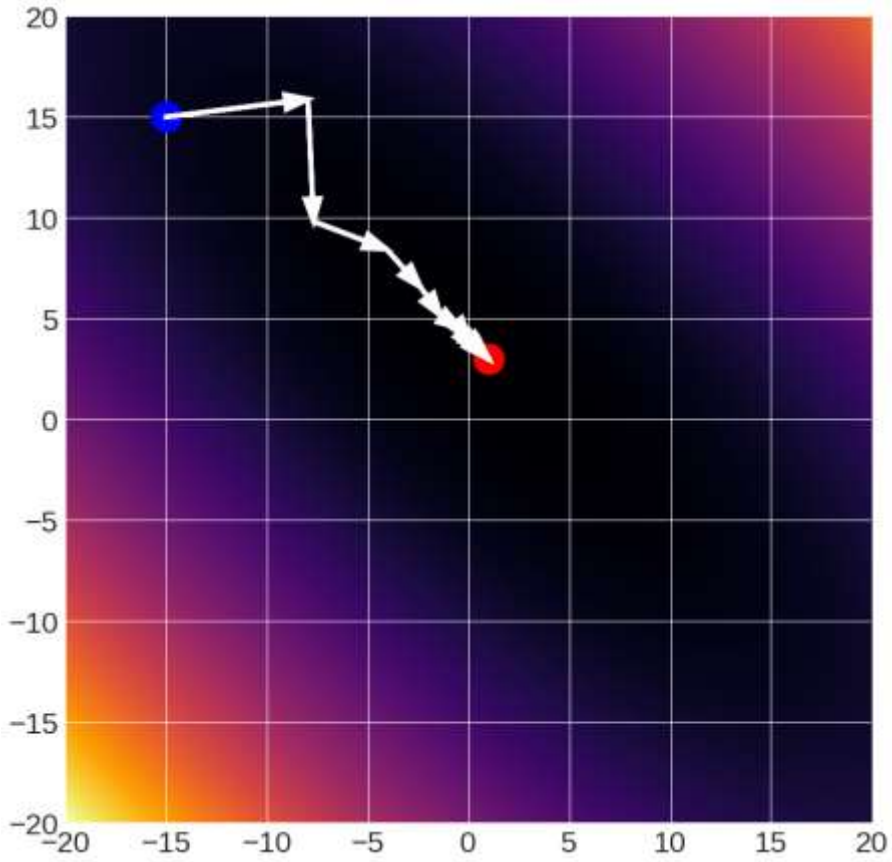


梯度下降

$11x10^{-2}$:oscillates
(Raw gradients)

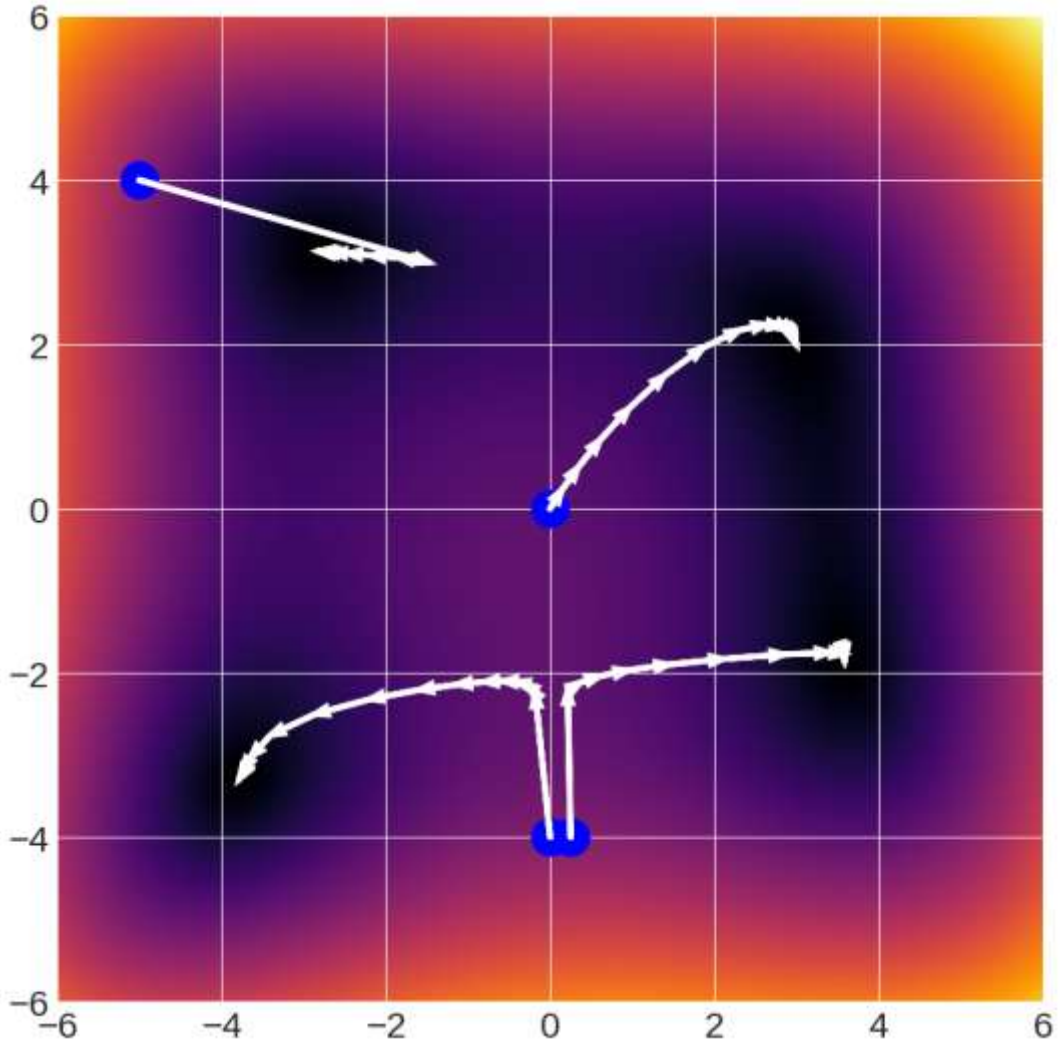


$11x10^{-2}$
(0.25 momentum)



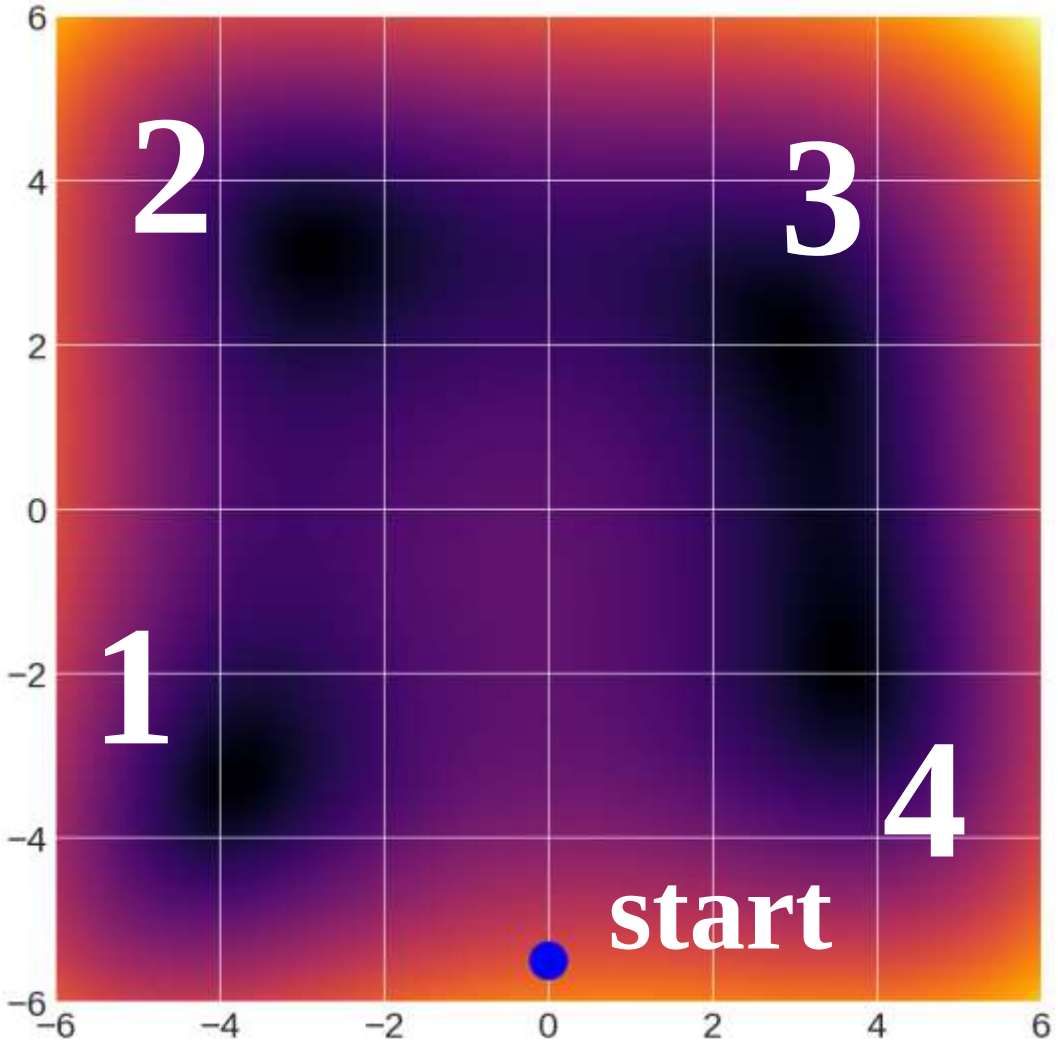
梯度下降

多个极小值时
→
梯度下降会找到
局部极小



梯度下降

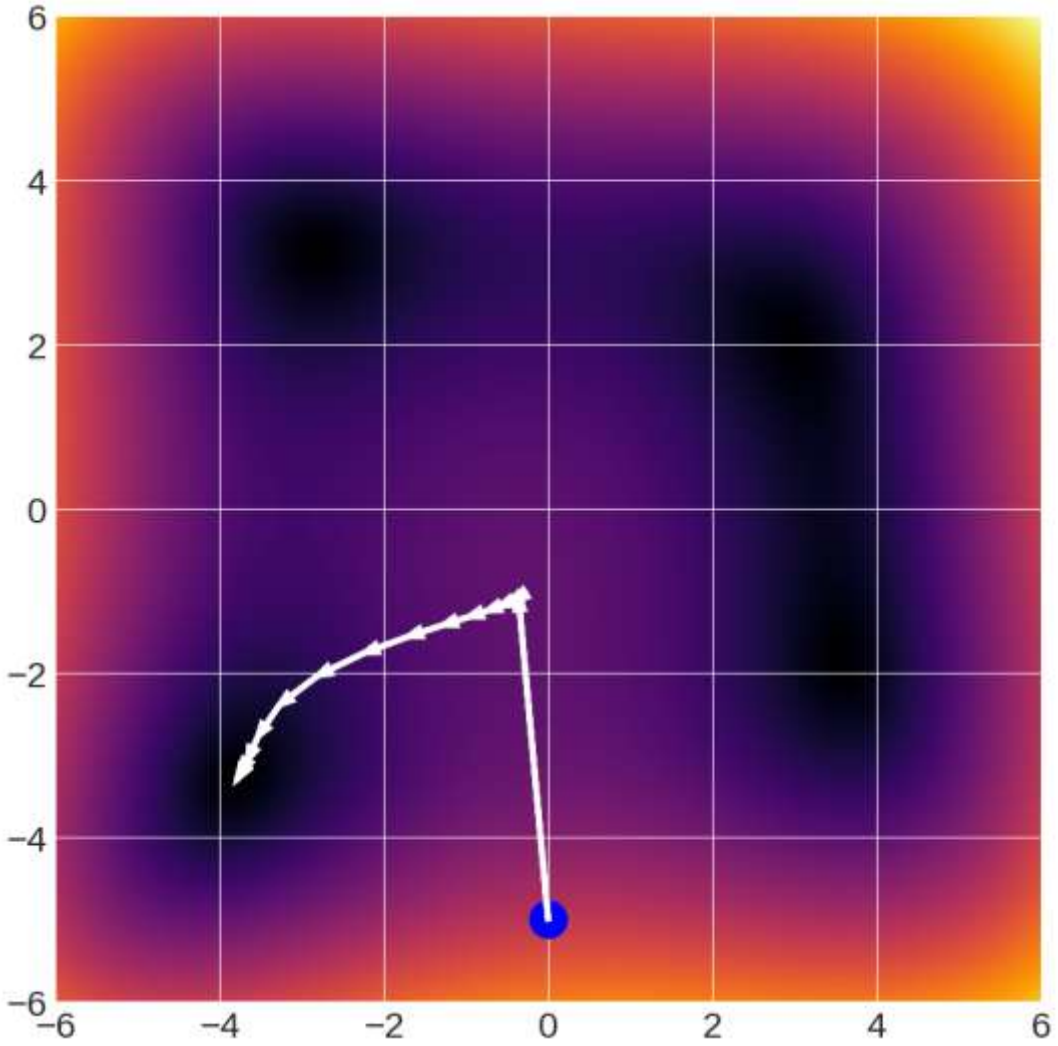
如果我们从蓝色点开始，会走向哪个局部最小？



梯度下降

许多函数是凸的
convex: 局部最小
就是全局最小

但是大多数函数
并不是



经验

- **一般来说**:小批量随机梯度下降（SGD）+ 动量+ 逐步调整的学习率
- 存在其他的更新规则（例如，AdamW）。在某些问题上，它们经常表现得更好。

损失函数

- 交叉熵损失仅仅是一种可能的损失函数。
  - 它的一个好属性是它将得分重新解释为概率，这些概率具有自然的含义。

- 支持向量机（SVM，最大间距）损失函数过去也很受欢迎。
  - 但目前，交叉熵是最常见的分类损失。

- 交叉熵可能更容易过拟合，特别是当模型在训练数据上获得非常高的分类准确性时。
- 最大间距可能更加健壮，因为它关注的是那些难以正确分类的样本，并试图确保它们被正确分类，而不是关注所有样本。

过拟合、欠拟合 与 模型复杂度

多项式回归: 给定 x, 预测 y

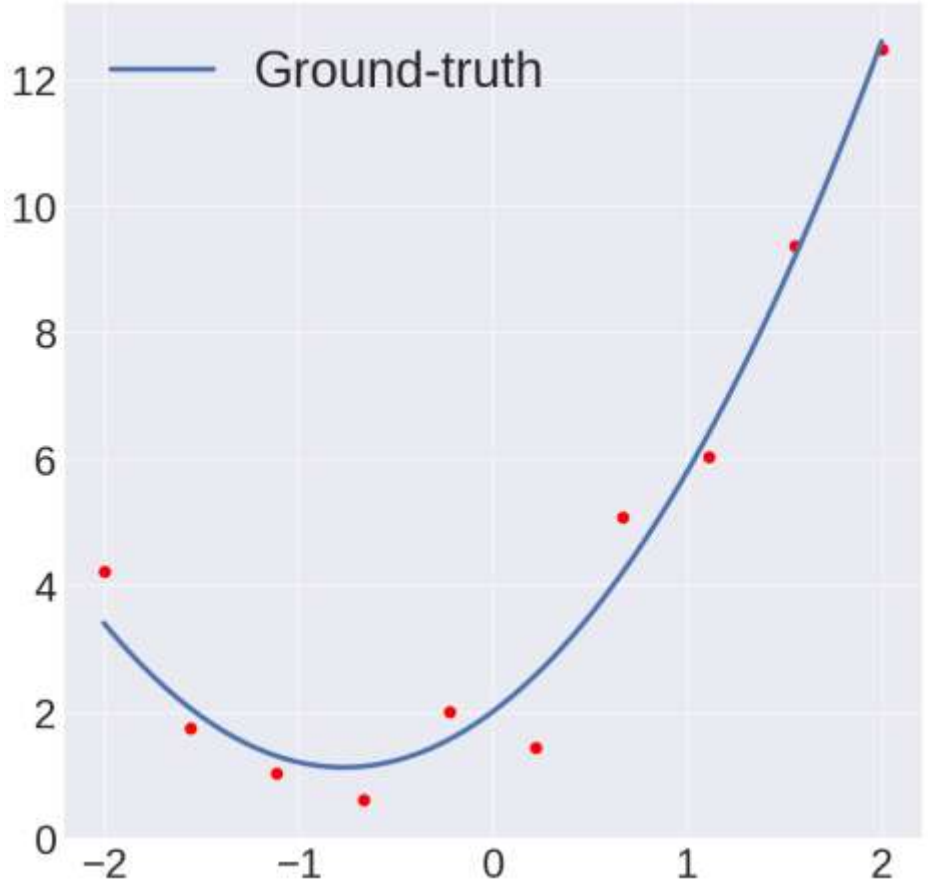
$$\begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} x_1^F & \cdots & x_1^2 & x_1 & 1 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ x_N^F & \cdots & x_N^2 & x_N & 1 \end{bmatrix} \begin{bmatrix} w_F \\ \vdots \\ w_2 \\ w_1 \\ w_0 \end{bmatrix}$$

我们有一个输入矩阵 **X**，其中包含了各种多项式次数的数据（例如，$\boldsymbol{x}, x^2, x^3$ 等）。这样，模型可以选择最佳的多项式次数来拟合数据。

权重W: 每个多项式度数都有一个相应的权重,这些权重决定了每个多项式度数在模型中的重要性。

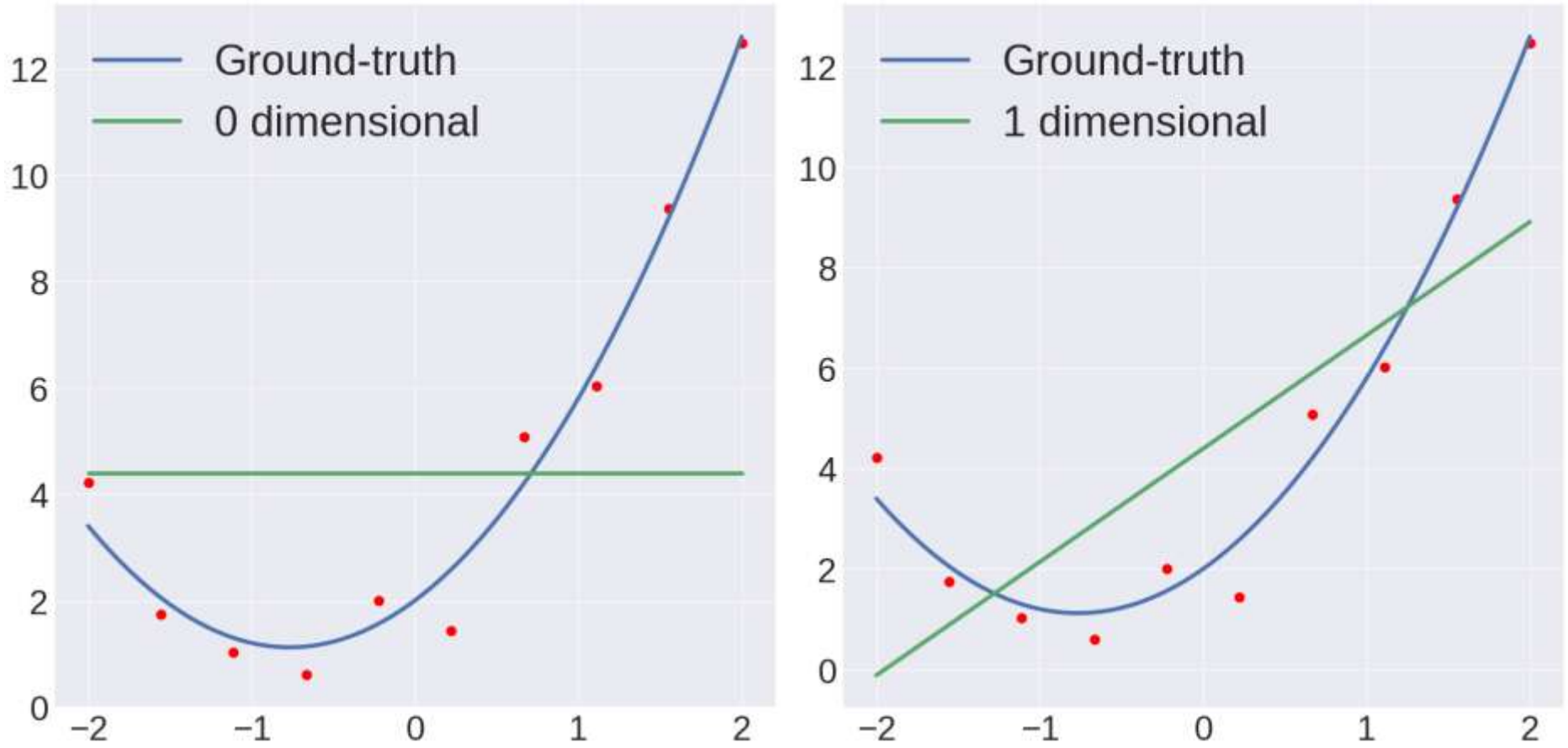
过拟合、欠拟合 与 模型复杂度

Model: $1.5x^2 + 2.3x+2 + N(0,0.5)$

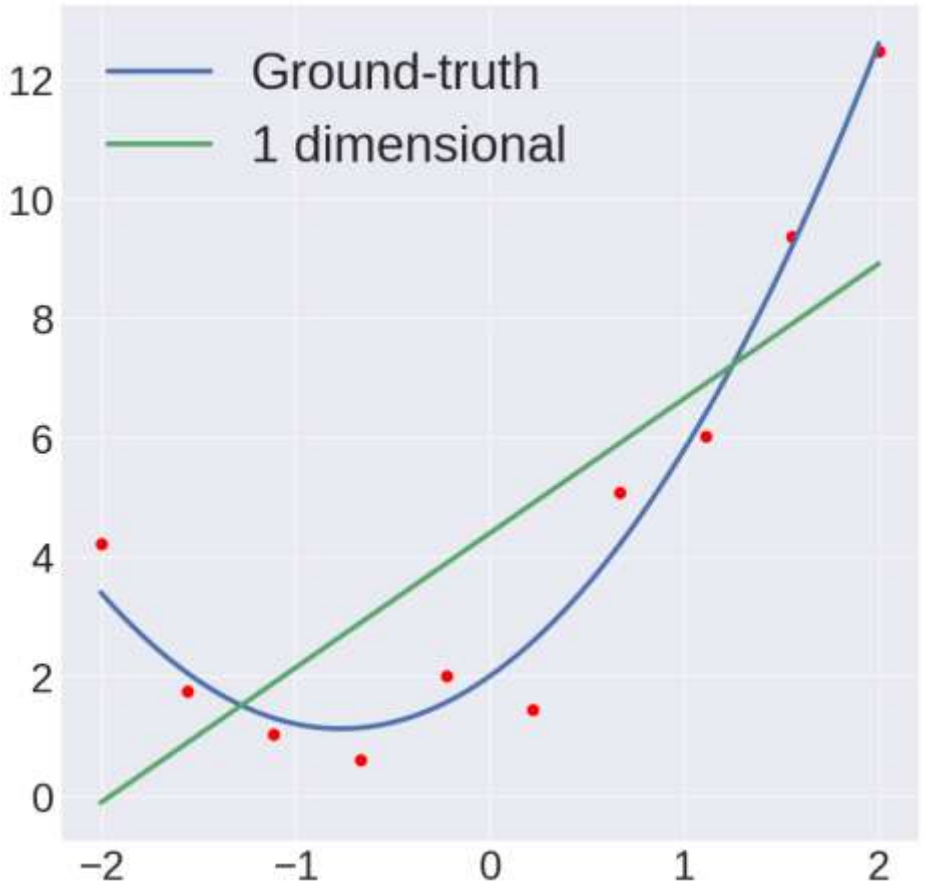


欠拟合

数据: $1.5x^2 + 2.3x+2 + N(0,0.5)$



欠拟合

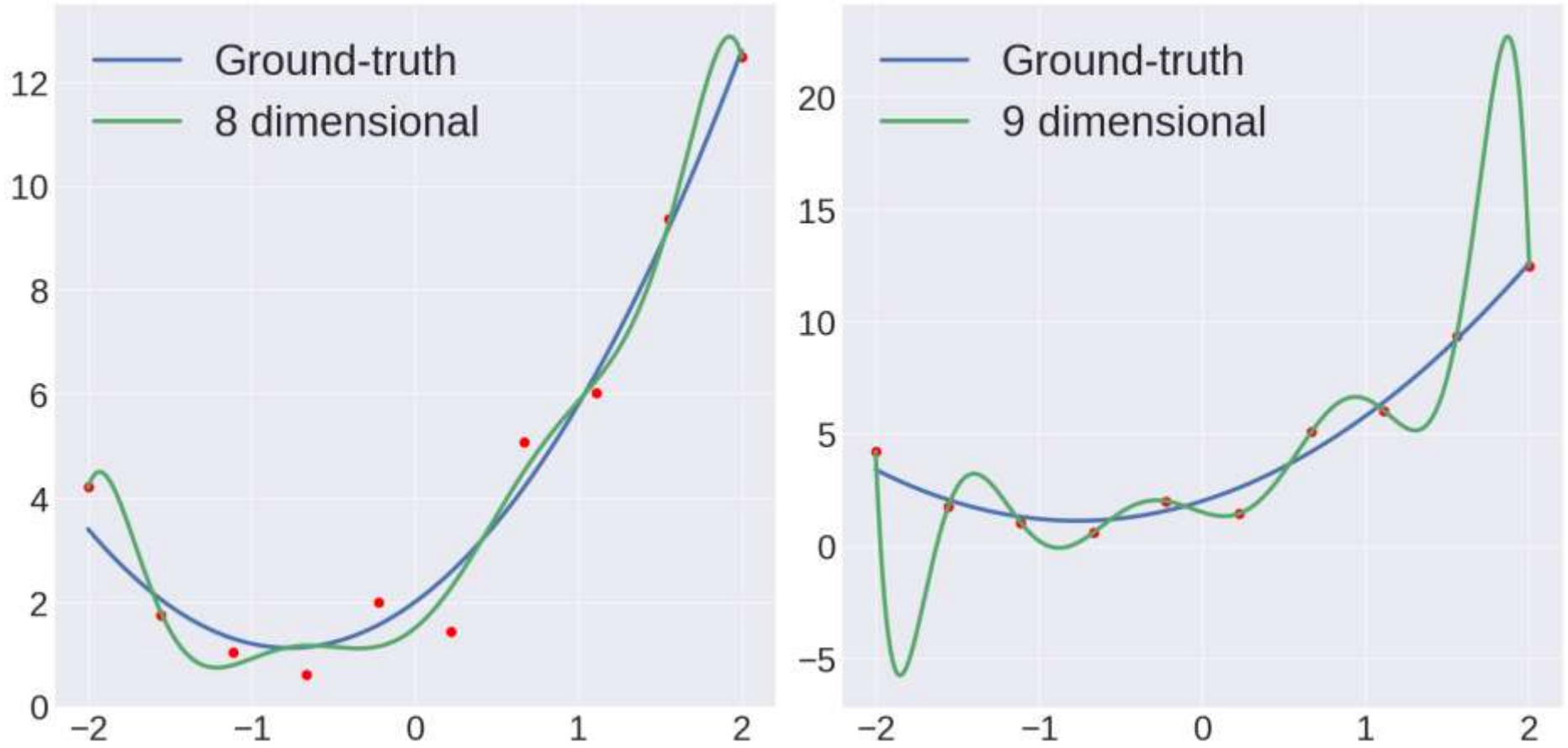


模型没有足够的参数或复杂度去拟合数据。即使在训练数据上，模型的表现也可能不尽如人意。

Bias 偏差 (statistics): 模型固有的错误

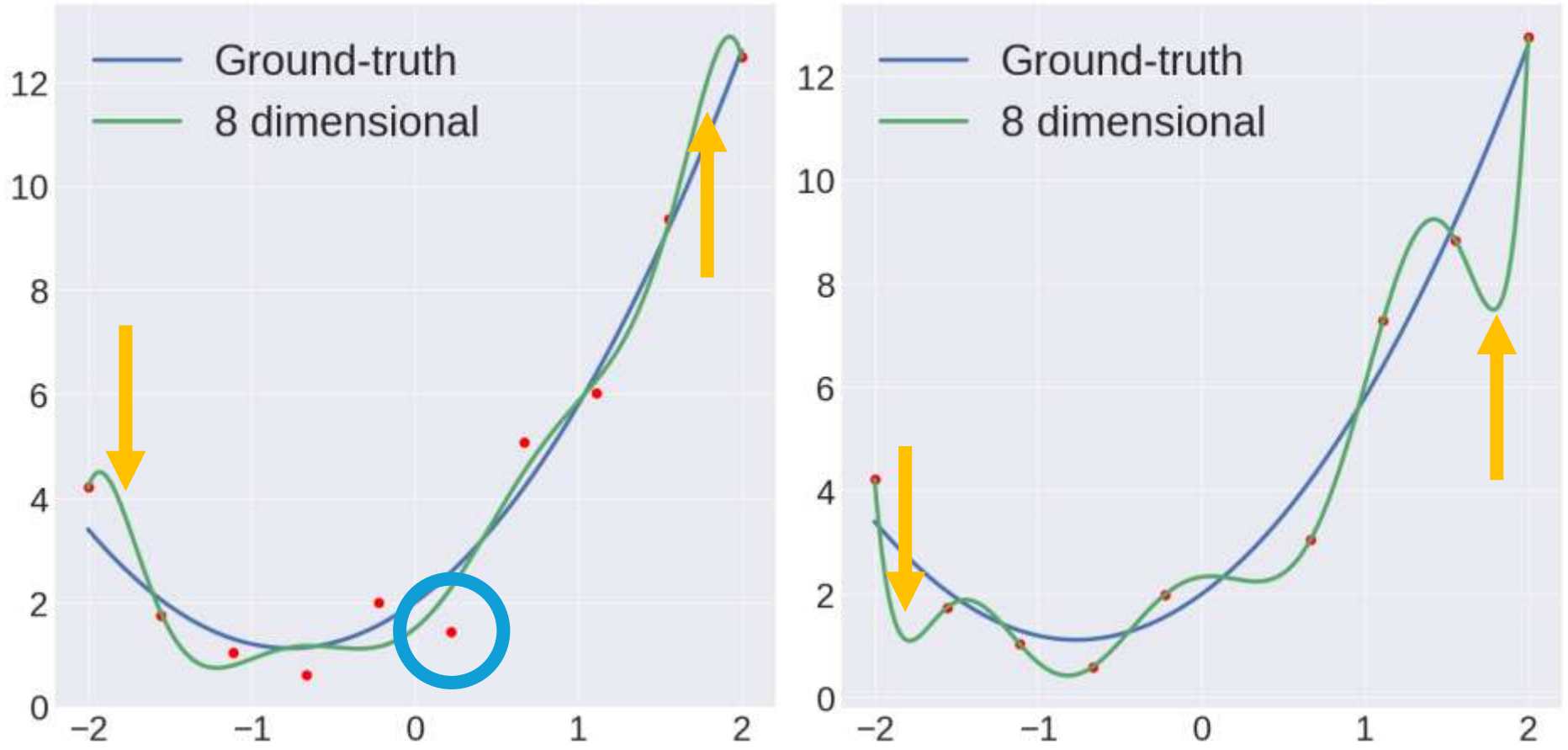
过拟合

数据: $1.5x^2 + 2.3x+2 + N(0,0.5)$



过拟合

高方差 variance:过高的方差意味着模型对训练数据的微小变化非常敏感 移除 一个数据点, 模型就会出现很大变化



模型复杂度

$$\arg\min_{W} \lambda \|W\|_2^2 + \sum_{i=1}^{n} -\log\left(\frac{\exp((Wx)_{y_i})}{\sum_k \exp((Wx)_k)}\right)$$

Regularization 正则化:
对模型的权重添加惩罚

交叉熵损失

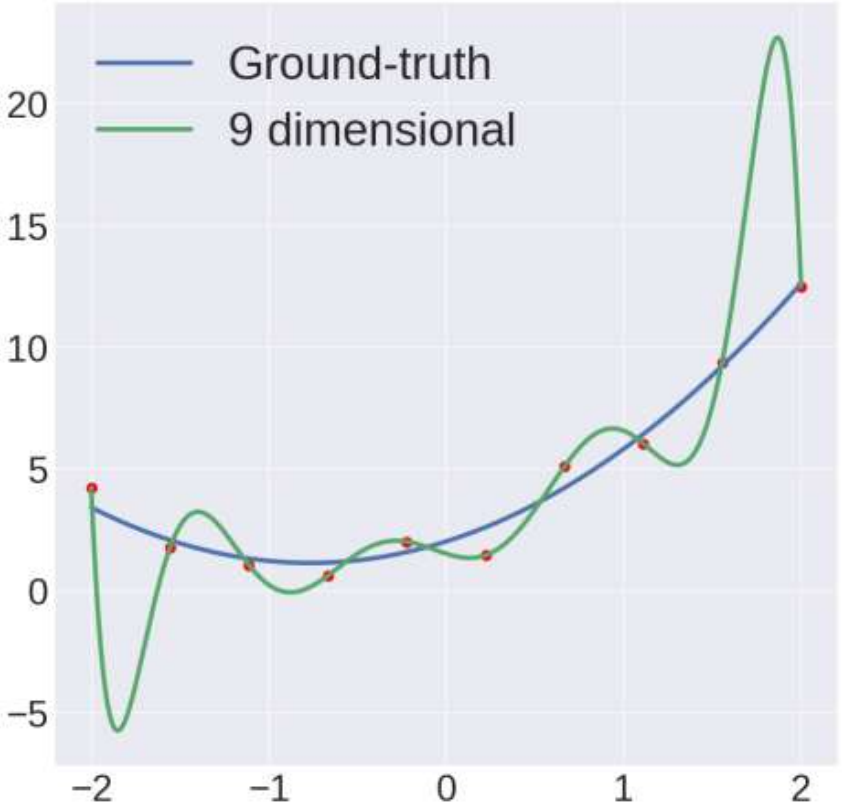
模型复杂性是指模型的容量或能力，它决定了模型可以适应多复杂的数据分布。
复杂的模型可能在训练数据上拟合得很好，但在新数据上可能表现得不好，
这被称为过拟合。

Model 1: $0.01*x_1 + 1.3*x_2 + -0.02*x_3 + -2.1x_4 + 10$

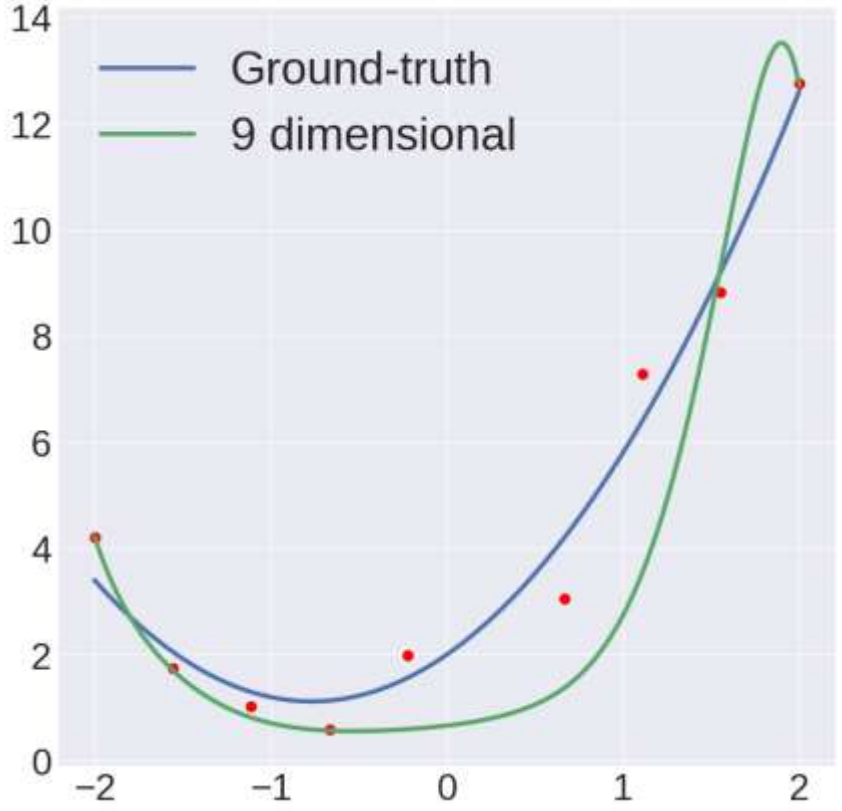
Model 2: $37.2*x_1 + 13.4*x_2 + 5.6*x_3 + -6.1x_4 + 30$

正则化

没有正则化:
（过）拟合所有点



使用正则化:
无法拟合所有点



总的来说

当模型在新数据上表现不佳时，通常是由三种误差组合而成的：偏差、方差和固有误差。

1. Bias: 偏差描述的是模型的简化性，它是模型假设与真实数据之间的差异。一个高偏差的模型可能太简单，无法捕捉到数据的真实模式。
2. Variance: 方差描述的是模型对训练数据的敏感性。高方差模型在不同的训练数据集上可能会有很大的差异。这通常是因为模型过于复杂，试图捕捉训练数据中的每一个小细节，甚至包括噪声。
3. Inherent: 固有误差是与数据本身相关的误差，与模型无关。例如，数据可能包含噪声，或者某些特征无法被测量。

通常，减少偏差可能会增加方差，反之亦然。选择合适的模型复杂度是关键。

欠拟合与过拟合

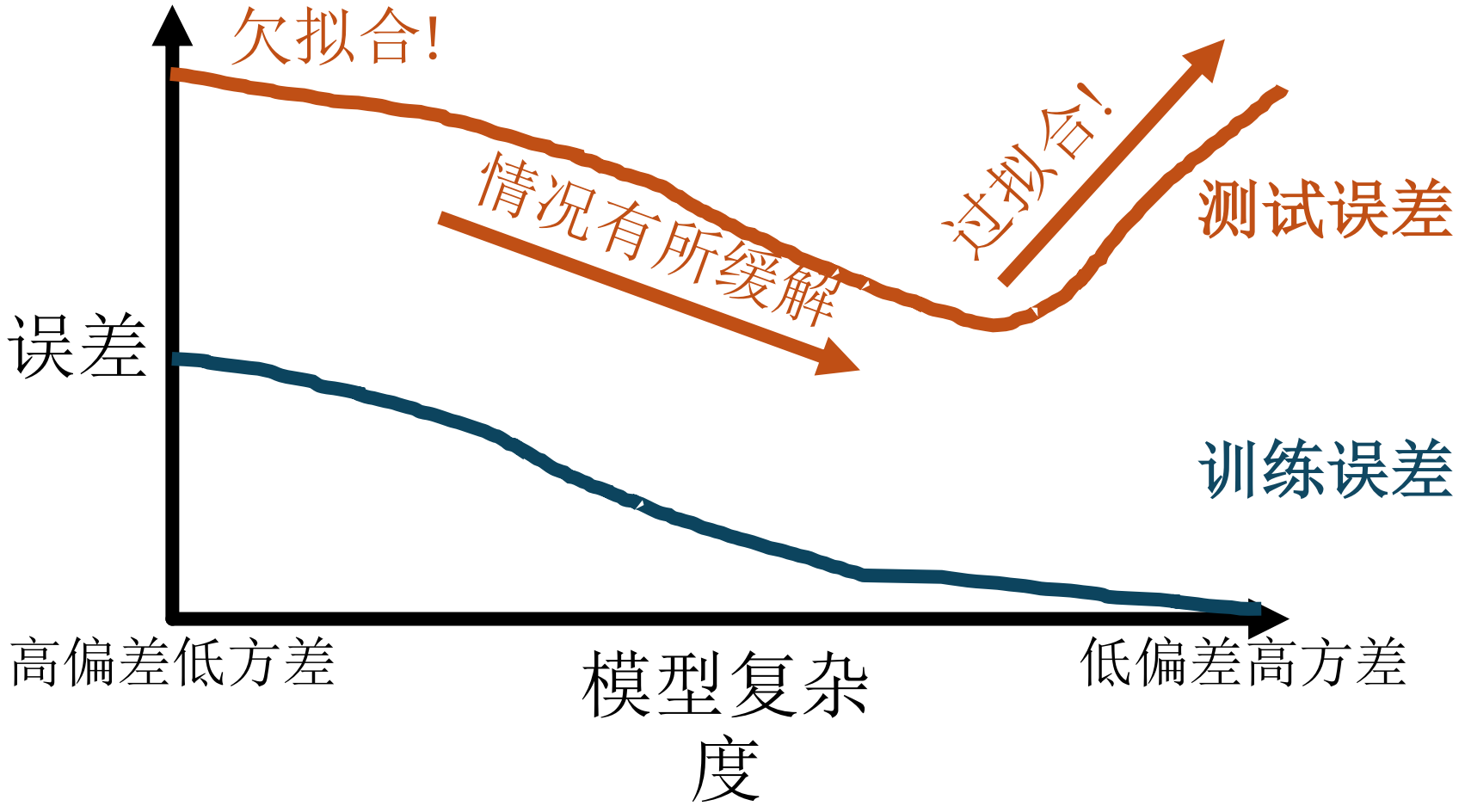


Diagram adapted from: D. Hoiem

欠拟合与过拟合

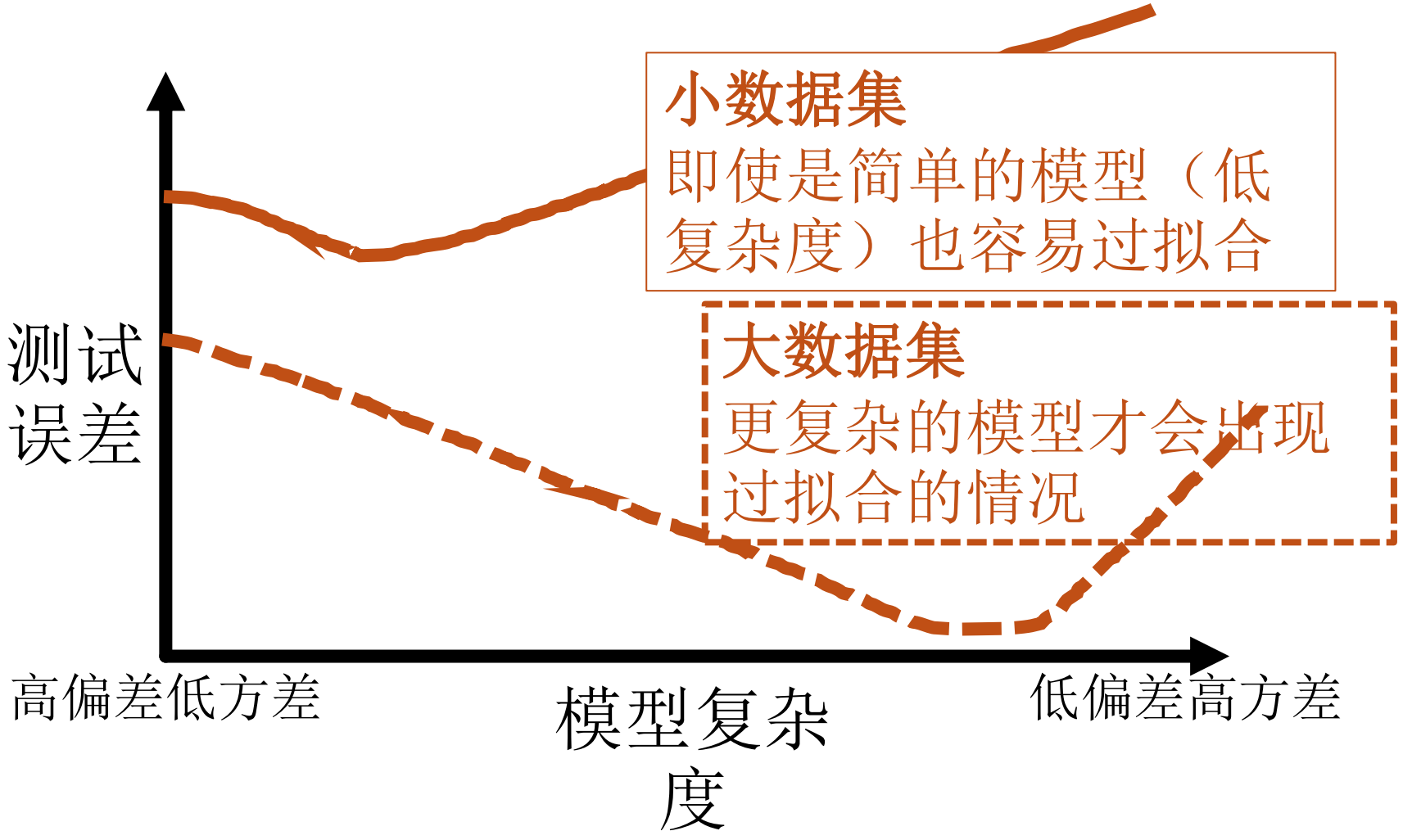
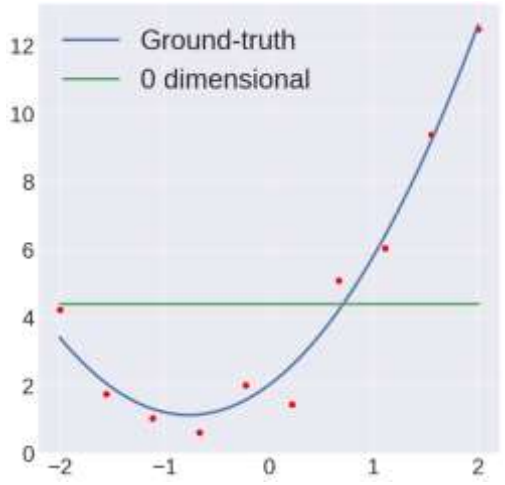
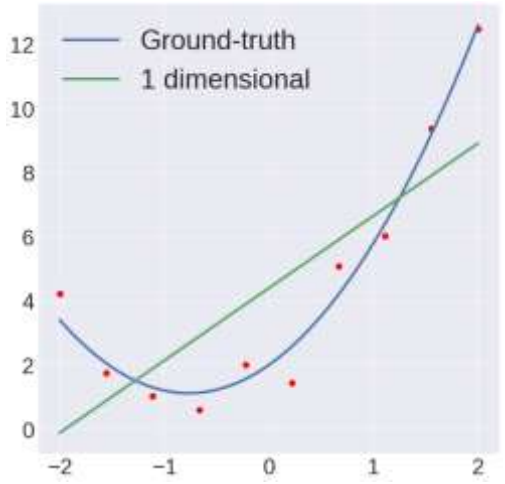


Diagram adapted from: D. Hoiem

欠拟合



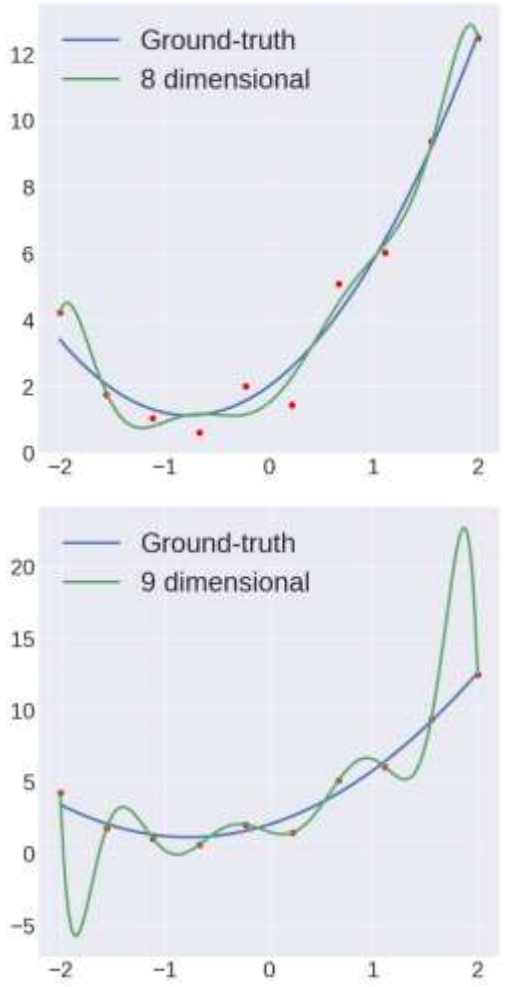


当模型的复杂度过低时，即使在训练数据上也不能得到满意的拟合效果。这通常意味着模型太简单，不能捕获数据的真实模式。

解决方案:

- **增加更多特征**：更多的特征可以增加模型的复杂度，使其更容易捕获到数据中的模式。
- **使用更强大的模型**：如从线性模型升级到多项式模型或使用深度学习模型。
- **减少正则化**：正则化是为了避免过拟合而添加的惩罚项，但如果模型欠拟合，则可以减少或移除正则化。

过拟合



模型在训练数据上表现很好，但在验证数据上表现较差，通常是因为模型太复杂，过度拟合了训练数据中的噪声或异常值。

解决方案:

- **增加更多数据**：更多的训练数据可以帮助模型学习到更广泛的数据分布，从而减少过拟合。
- **使用简单模型**：选择一个复杂度较低的模型，避免过度拟合。
- **增加正则化**：正则化可以防止模型过于复杂，从而限制其拟合能力。

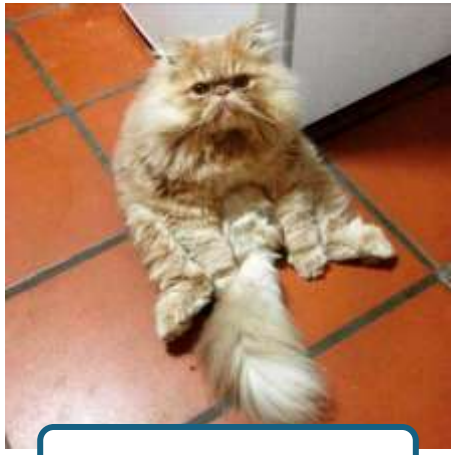
建议：首先确保模型有足够的能力进行拟合（即避免欠拟合），然后再考虑如何避免过拟合。

回顾：外观的巨大差异让识别问题困难倍增

- 相同类别的物体图像差异极大

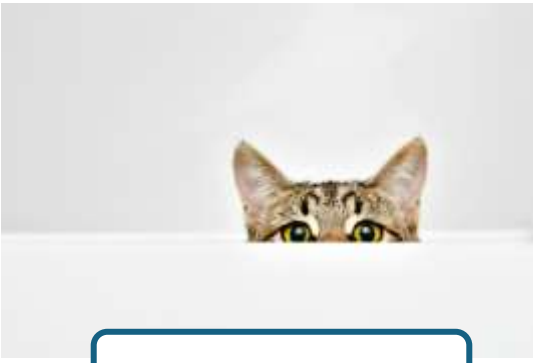
Viewpoint Variation

Lighting Variation



Deformation

Background Clutter



Occlusion

问题：欠约束 Under-constrained

- **不同的现实情境可以产生相同的图像**：这意味着从同一个图像出发，我们可能会得到多种不同的解释或解决方案。
- **通常我们不能计算“正确”的答案**：由于存在多种可能的现实情境，我们通常不能确定一个图像的“正确”解释。但我们可以计算最有可能的答案。
- **我们需要某种先验来加以条件**：由于问题是欠约束的，我们需要某种先验知识或假设来缩小可能的答案范围。
- **我们可以从数据中学习这个先验**：通过训练数据，我们可以学习到这些先验知识或假设，以更好地解决问题。

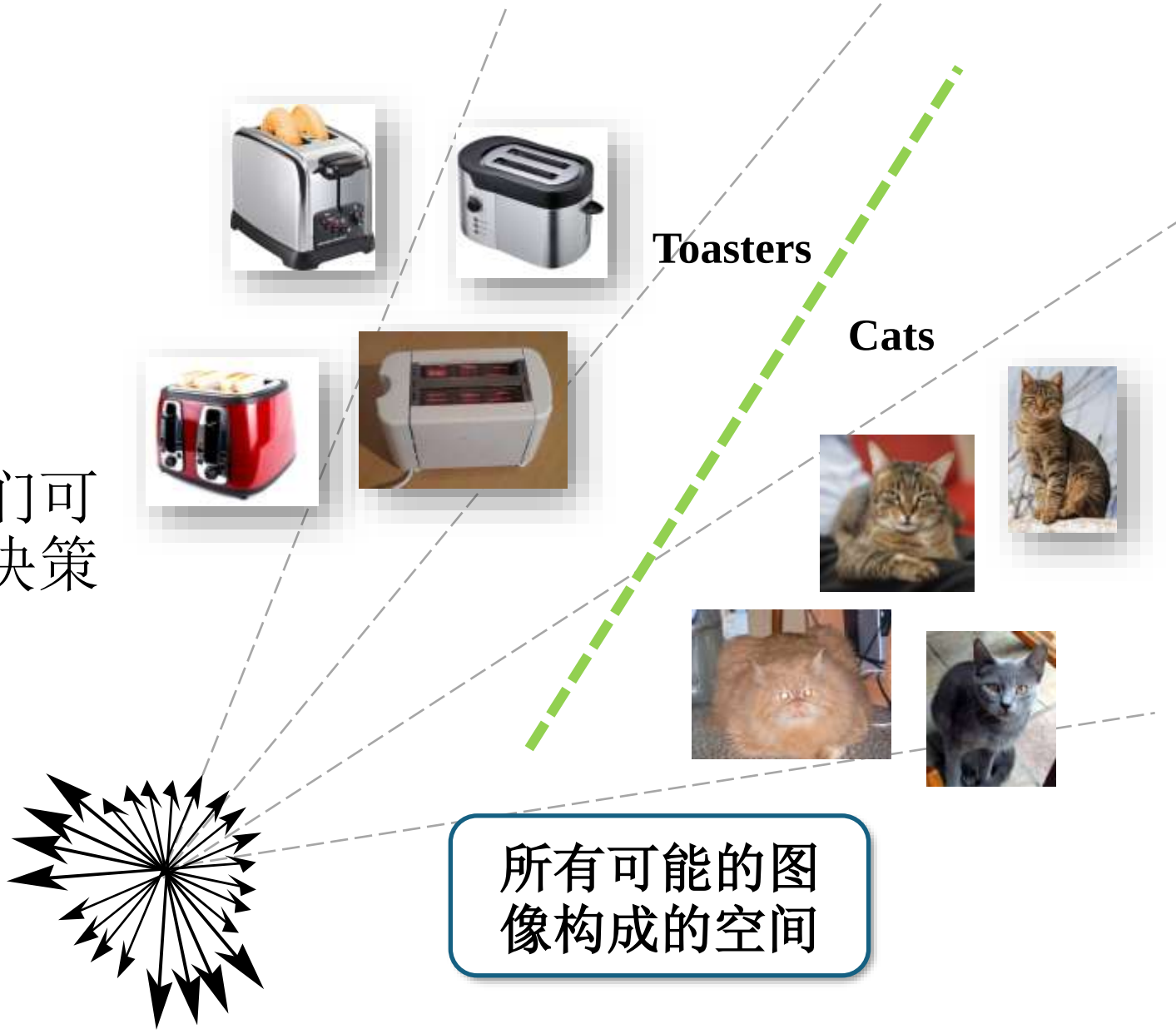
$$f(x) = \underset{\ell_x}{argmax}\ P(\ell_x | data)$$

将图像视为高维向量

- **图像只是一组数字：**每幅图像都可以看作是一个由像素值组成的数字数组。对于灰度图像，这些数字通常是0到255之间的值。对于彩色图像，每个像素通常包含三个值（RGB）。
- **将图像转换为向量：**为了便于计算和处理，我们可以将这些数字堆叠成一个向量。例如，对于一个100x100的灰度图像，我们可以将它转换为一个10,000维的向量。
- **训练数据变为高维点集：**当我们有多幅图像时，每幅图像都可以转换为一个高维向量。因此，我们的训练数据集就可以看作是在高维空间中的一组点。

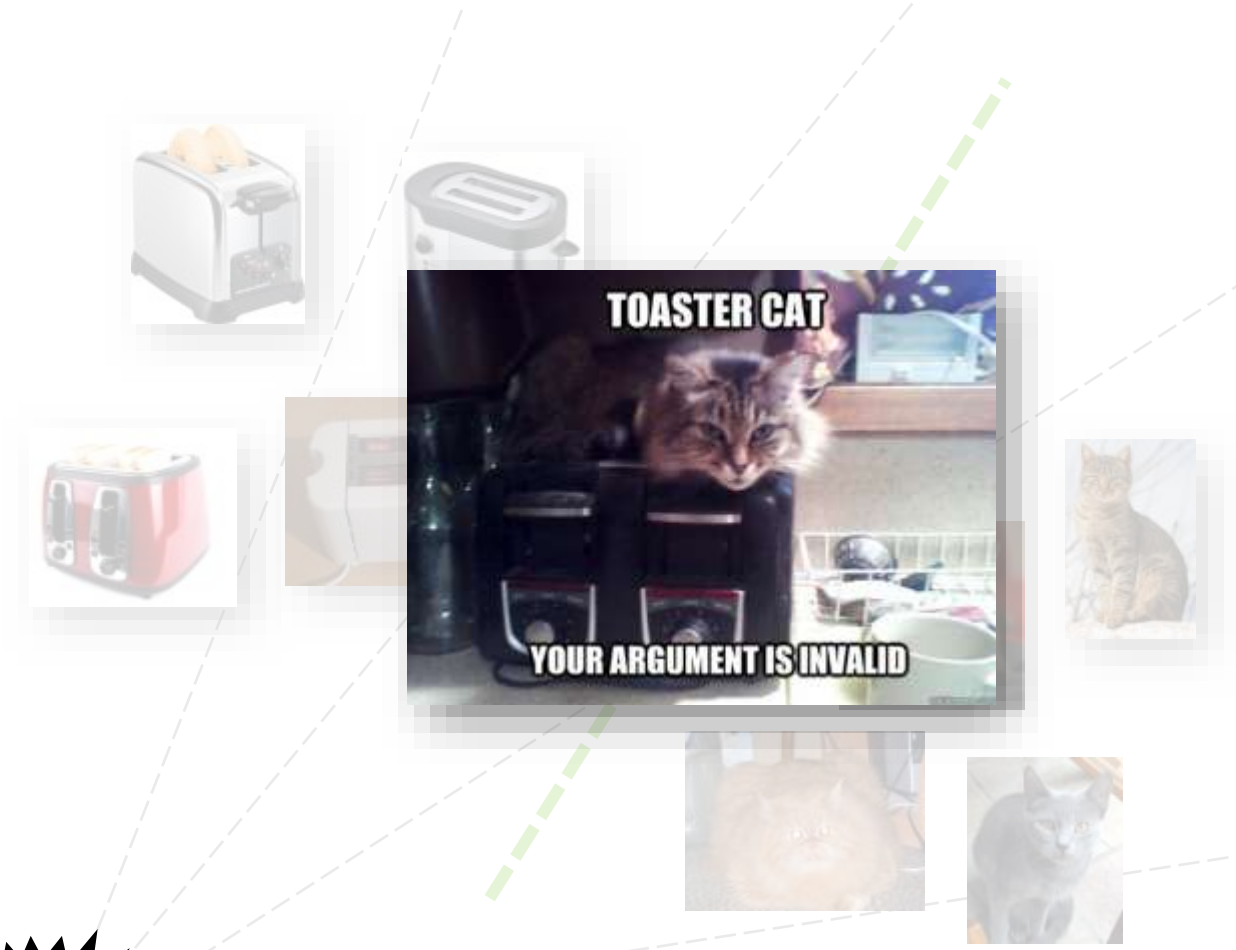
将图像视为高维向量

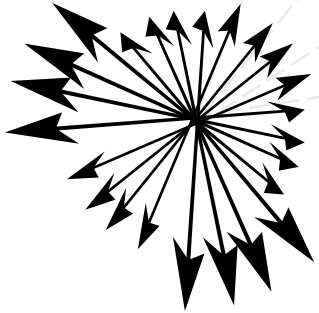
- 图像只是一组数字
- 将图像转换为向量
- 训练数据变为高维点集
- 将图像转换为高维向量后，我们可以尝试在这个高维空间中找到决策边界来分隔不同的类别。



将图像视为高维向量

- 图像只是一组数字
- 将图像转换为向量
- 训练数据变为高维点集
- 将图像转换为高维向量后，我们可以尝试在这个高维空间中找到决策边界来分隔不同的类别。





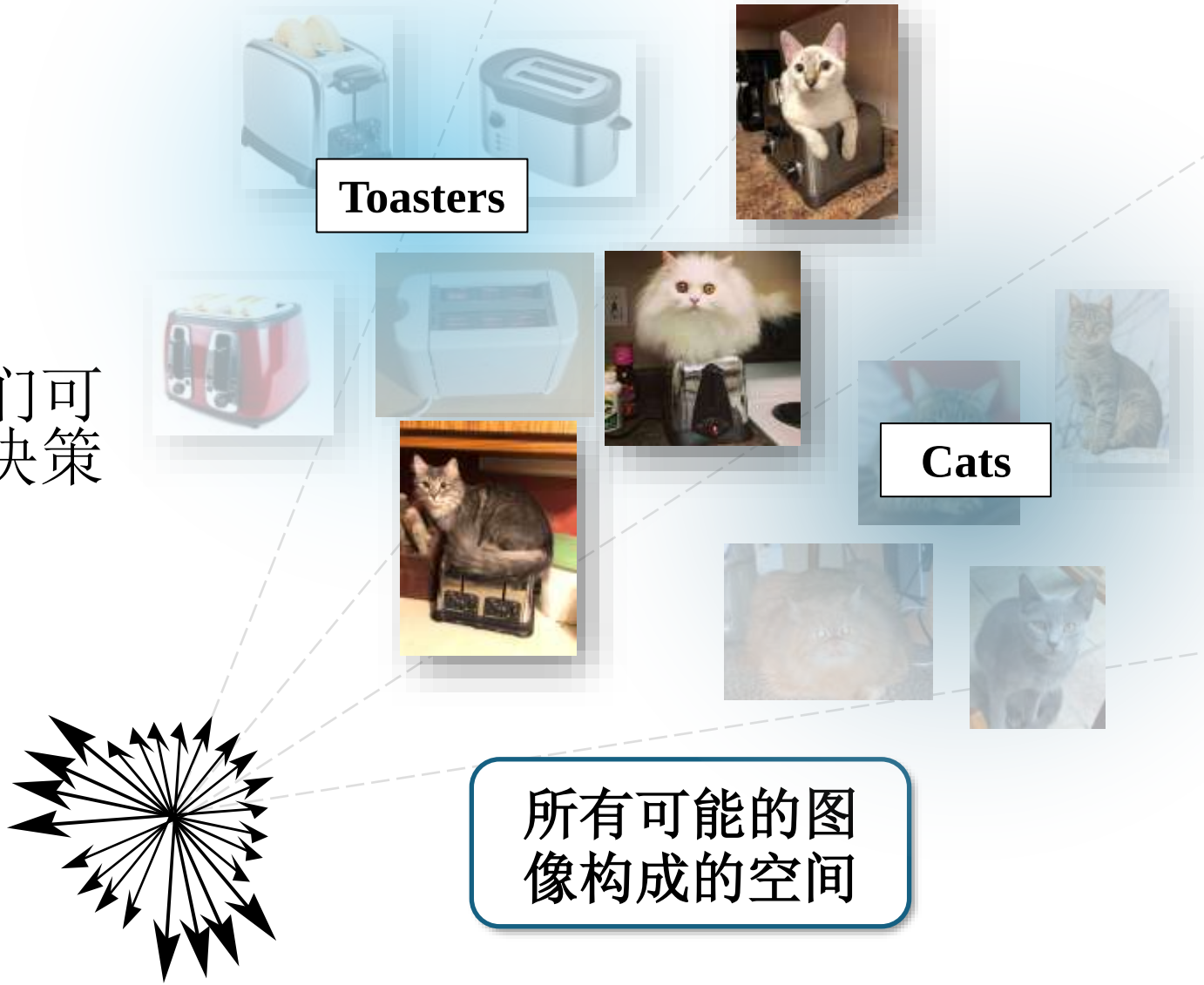
所有可能的图像构成的空间

将图像视为高维向量

- 图像只是一组数字
- 将图像转换为向量
- 训练数据变为高维点集
- 将图像转换为高维向量后，我们可以尝试在这个高维空间中找到决策边界来分隔不同的类别。

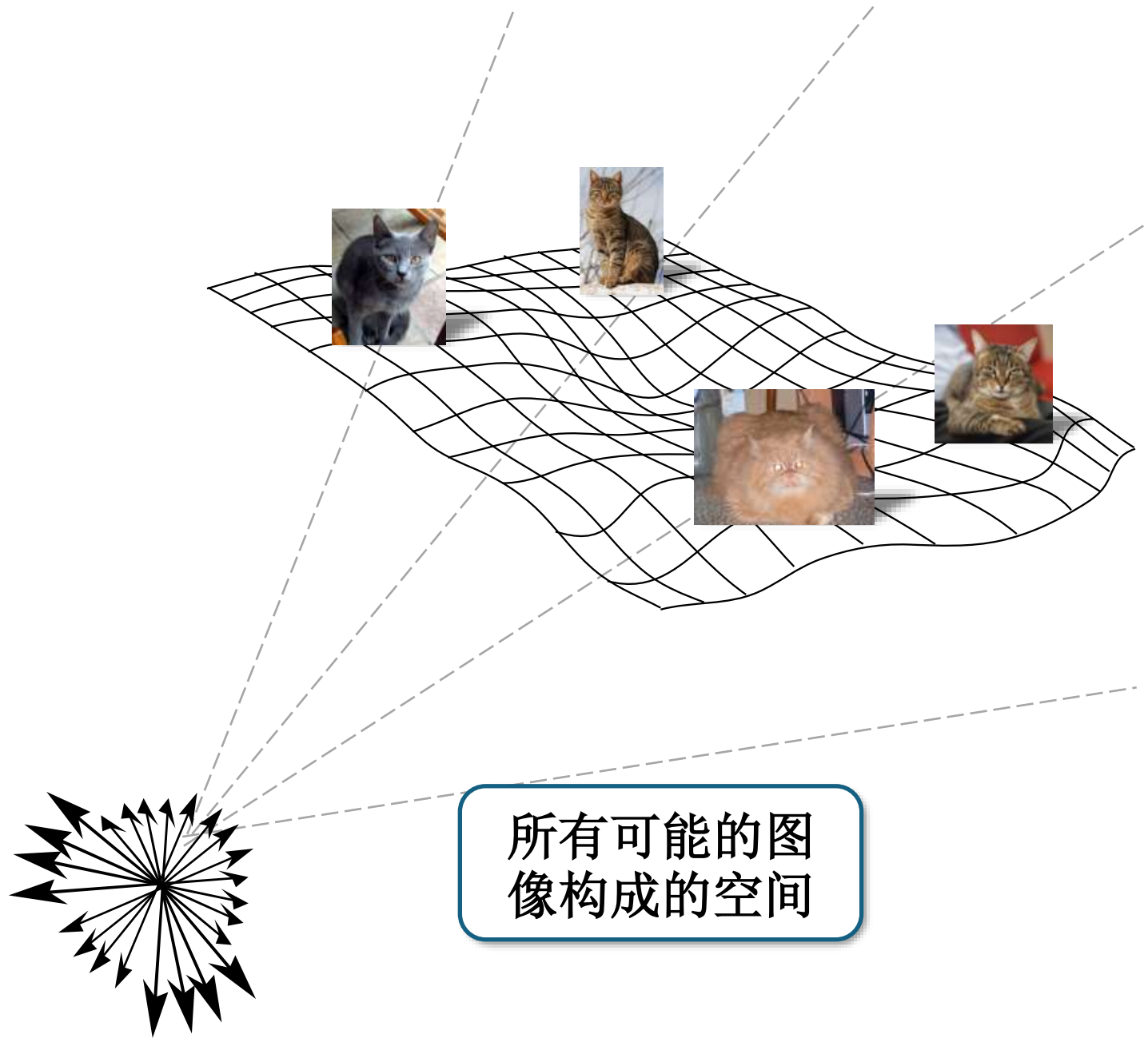
或者

- 为每个类别定义一个概率分布。
  - 来允许toaster-cat的存在。



图像的高维性 与 降维表示

- 一个图像有多少维度？
  - 图像的所有像素和它们的颜色通道。
  - iPhone X 照片:
    - 4032 x 3024 pixels
    - 3 colors
    - 36,578,304 pixels (36.5 Mega pixels)
- 在实际应用中，虽然图像可能具有高度的维度，但许多图像都倾向于聚集在某种低维结构或流形上。
- 降维可以帮助我们提取图像的关键特征，并用更简洁的方式表示图像。

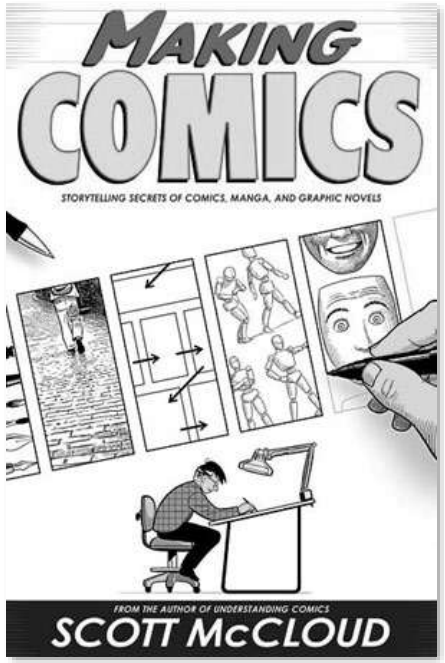


图像变换

全景图：图像对齐

为什么这样拼接下端的栏杆没有对齐?

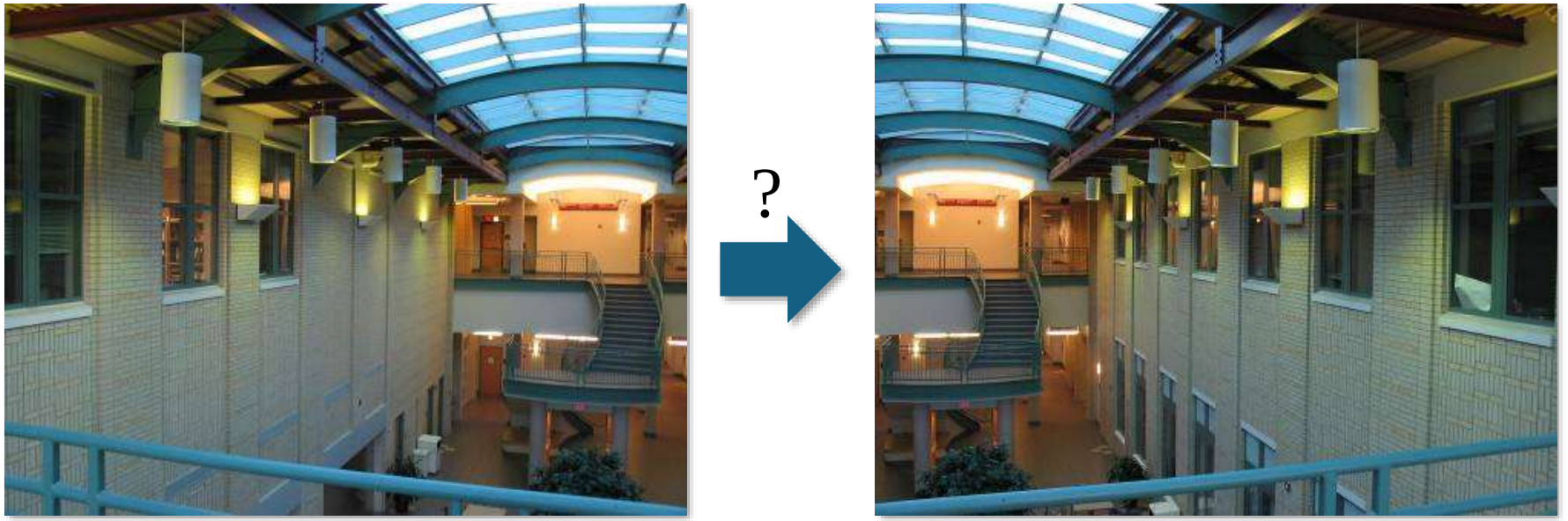
匹配的图像之间的几何关系是什么？



?

答案：相似性变换（平移、旋转、均匀缩放）

这两张图像之间的几何关系是什么？



这两张图像之间的几何关系是什么？

对制作全景图很重要!

首先，我们需要知道这种转换是什么。

其次，我们需要弄清楚如何使用特征匹配来计算它。

图像变换

图像滤波：更改图像**值域**

$$g(x) = T(f(x))$$

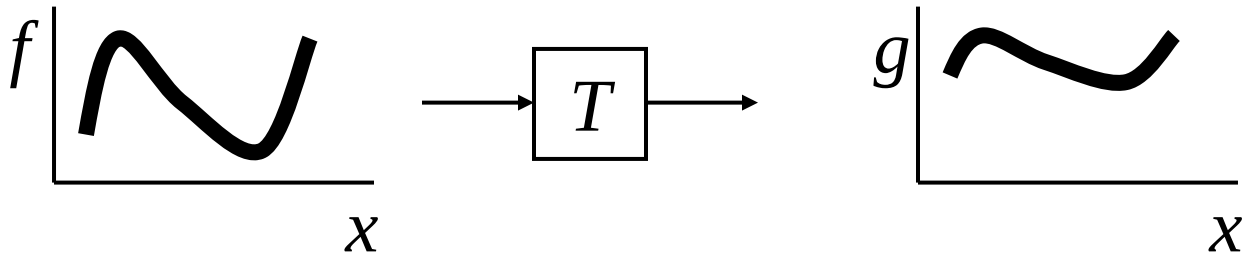
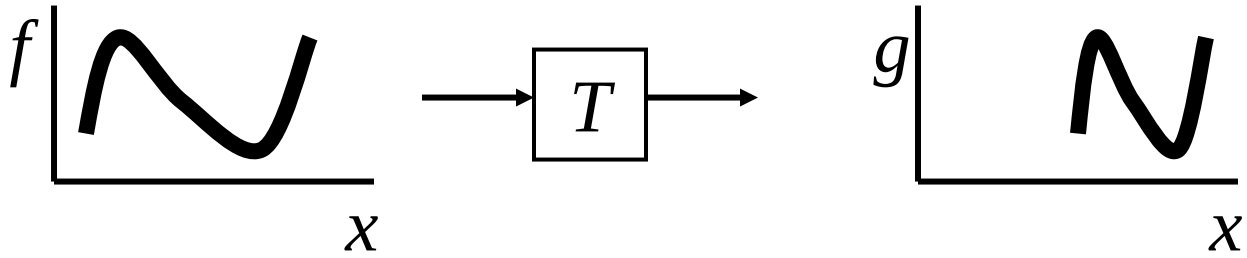


Image warping **图像扭曲**：更改图像的**定义域**（**domain**）

$$g(x) = f(T(x))$$

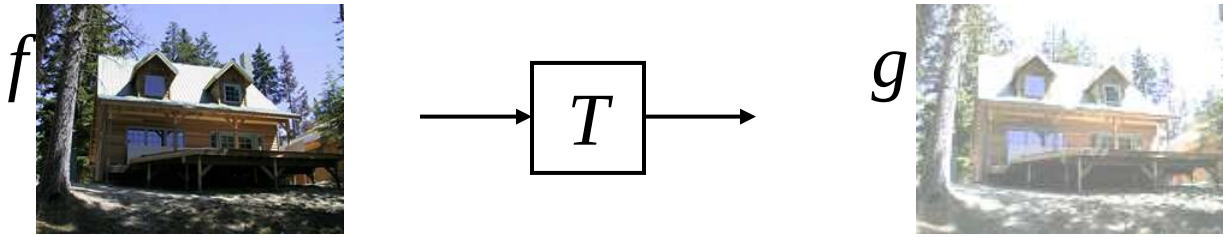


Slide credit: A. Efros

图像变换

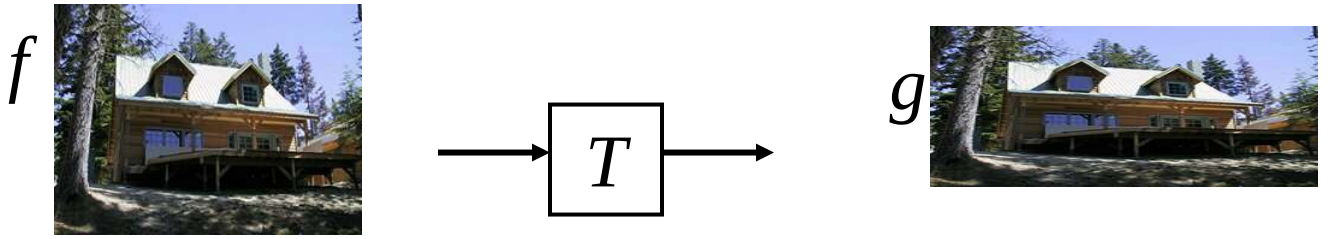
图像滤波：更改图像**值域**

$$g(x,y) = T(f(x,y))$$



图像扭曲：更改图像的**定义域**

$$g(x,y) = f(T(x,y))$$



Slide credit: A. Efros

参数化（全局）扭曲（Warping）

参数化扭曲的示例

Translation 平移

Rotation 旋转

Aspect 长宽比

Affine 仿射

Perspective 透视

Cylindrical 圆柱

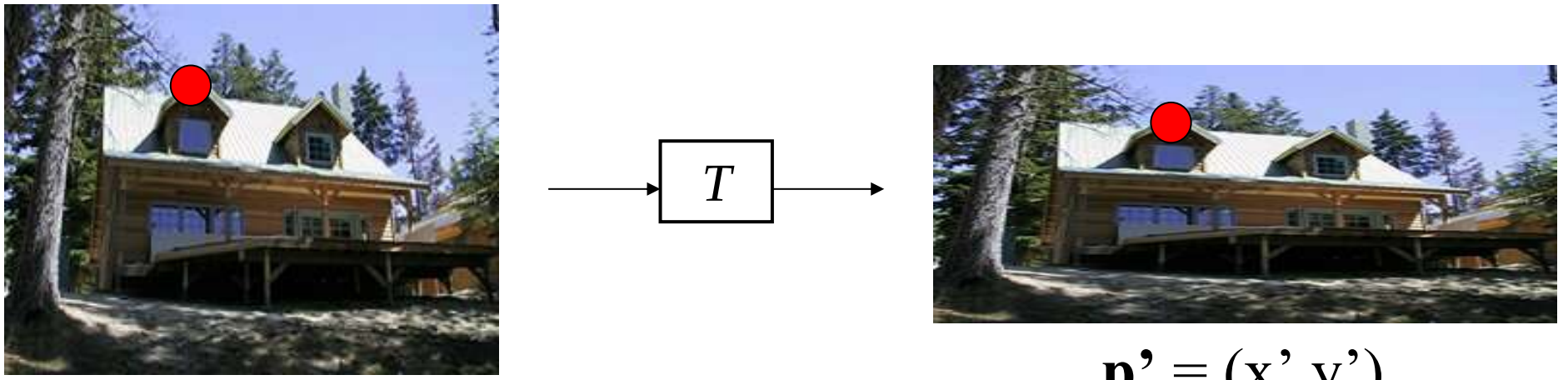
Slide credit: A. Efros

参数化（全局）扭曲

T 被描述为一个坐标变换机

$$\boldsymbol{p}' = T(\boldsymbol{p})$$

Note: 无论图像的内容是什么，T 都是相同的，且参数量很少，对图像的每个像素应用相同的变换



Slide credit: A. Efros

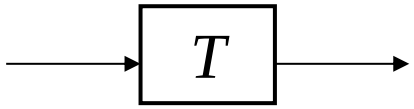
参数化（全局）扭曲

我们暂时只关注 线性变换

$$\boldsymbol{p' \equiv Tp}$$

T:矩阵; p, p’: 2D点。

p = (x,y)

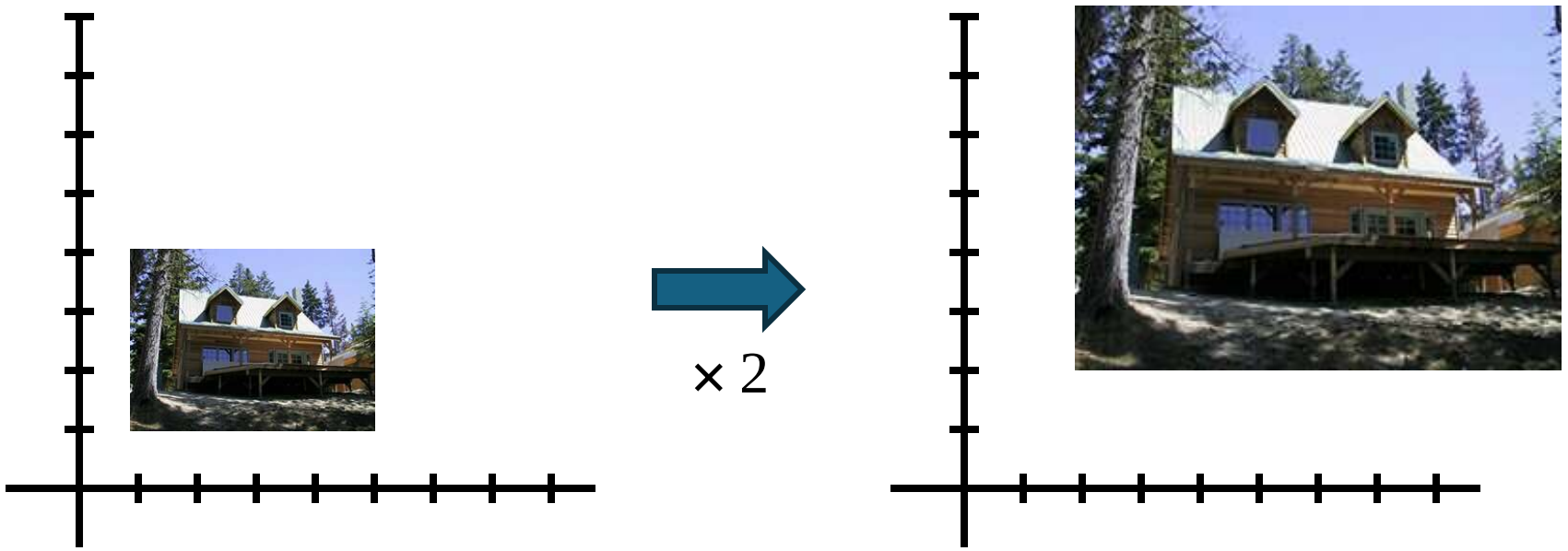


p’ = (x’,y’)

Slide credit: A. Efros

放缩

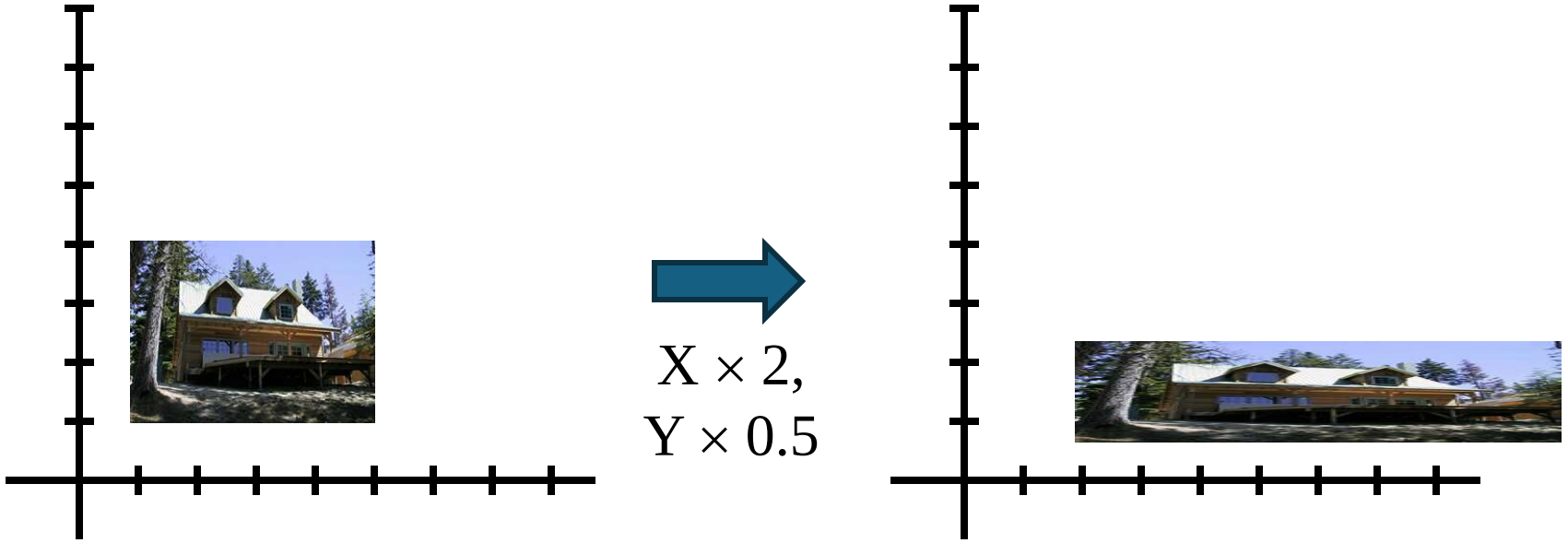
放缩 会把所有像素的坐标 (x,y) 乘以一个定值



Slide credit: A. Efros

放缩

放缩 会把所有像素的坐标 (x,y) 乘以一个定值



Slide credit: A. Efros

放缩

放缩变换的 T 是怎样的?

$$x' = ax$$

$$y' = by$$

矩阵形式:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\text{放缩矩阵 } S} \begin{bmatrix} x \\ y \end{bmatrix}$$

S的逆矩阵是?

Slide credit: A. Efros

2D 旋转

旋转矩阵

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

R_θ的逆矩阵是？ $\boldsymbol{I = R_\theta^T R_\theta}$

Slide credit: A. Efros

其他 T 为 2x2 矩阵能做的变换

Identity 等价变换

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Shear 倾斜

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & sh_x \\ sh_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Slide credit: A. Efros

其他 T 为 2x2 矩阵能做的变换

Before

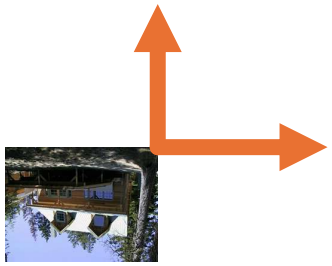
After

2D Y轴 镜像

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Before

After



2D 反转

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Slide credit: A. Efros

T 为 2x2 矩阵时

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = T \begin{bmatrix} x \\ y \end{bmatrix}$$

T 变换后

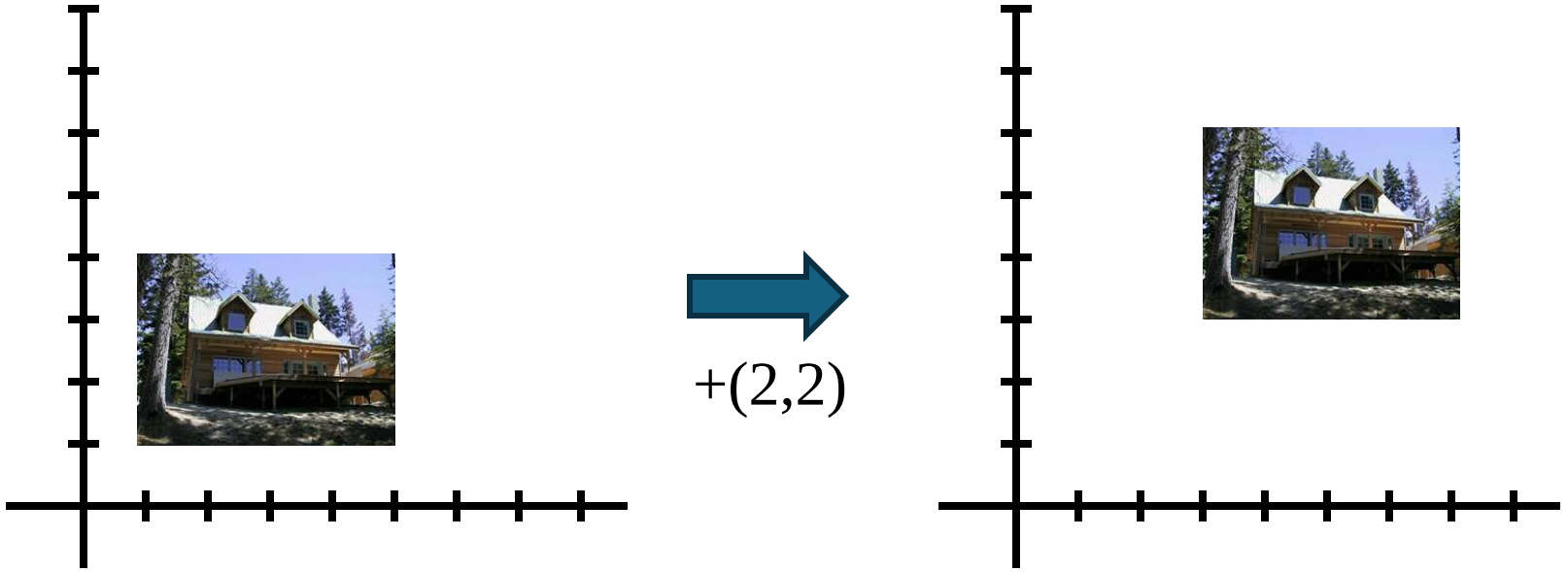
- 原点不变: **0 = T0**
- 线还是线
- 平行线仍然平行

T 为 2x2 矩阵不能做的变换

平移怎么做?

$x' = x + t_x, y' = y+t_y$

怎样用线性变换表示?

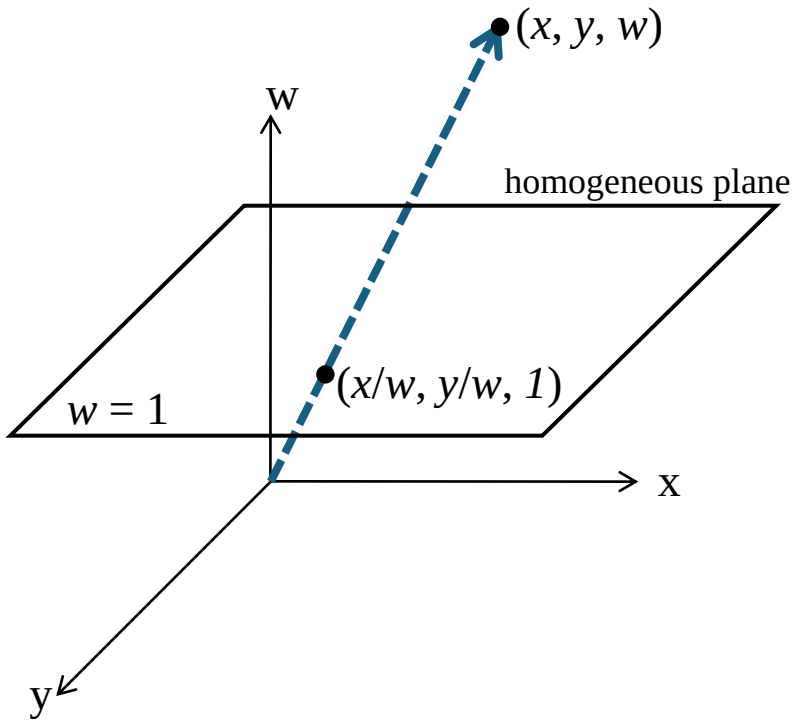


齐次坐标

Trick: 增加一维坐标:

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

齐次图像坐标



齐次坐标到图像坐标的转换

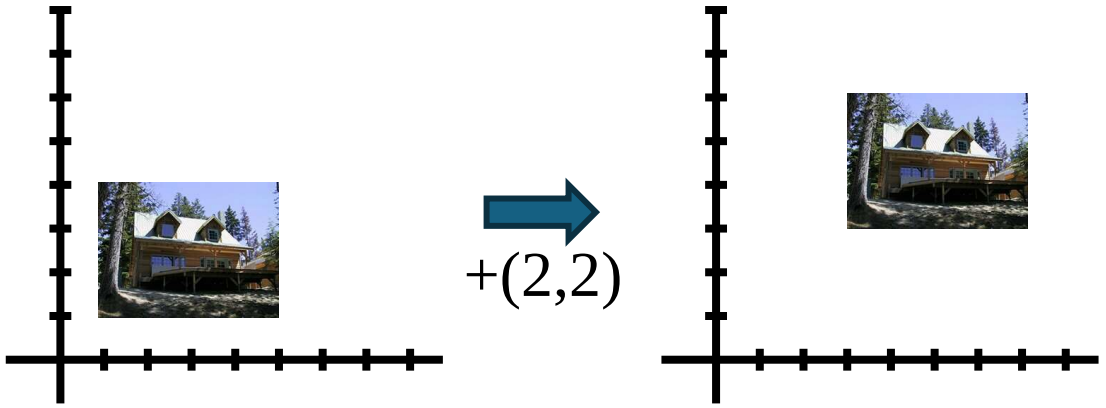
$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

图像齐次坐标

平移怎么做?

$x' = x + t_x$, $y' = y + t_y$

$$\begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix} \equiv \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \equiv \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



Slide credit: A. Efros

Affine transformations：仿射变换

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

任何 由 3x3 矩阵表示的变换，其最后一行为 [0 0 1]我们称之为仿射变换。

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

平移

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{s}_x & 0 & 0 \\ 0 & \boldsymbol{s}_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

放缩

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

2D旋转

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \boldsymbol{sh}_x & 0 \\ \boldsymbol{sh}_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

倾斜

仿射变换

- 仿射变换可以认为是以下变换的组合 …
  - 线性变换
  - 平移

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- 仿射变换的性质:
  - 原点变换后不一定在原点
  - 线仍然是线
  - 平行线仍然平行
  - 长度相对比例会保留
  - 仿射变换的组合还是仿射变换

这是仿射变换吗?

Where do we go from here?

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$$

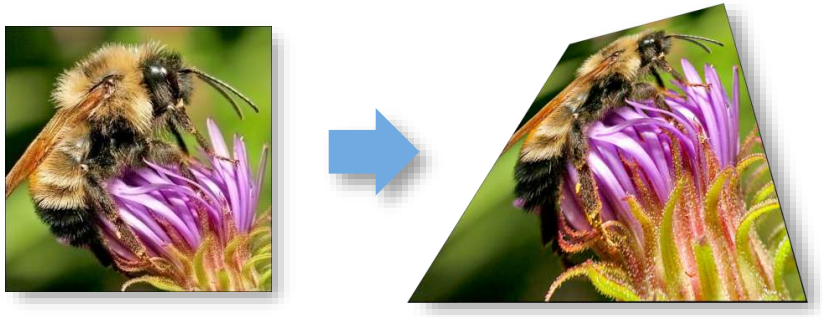
what happens when we mess with this row?

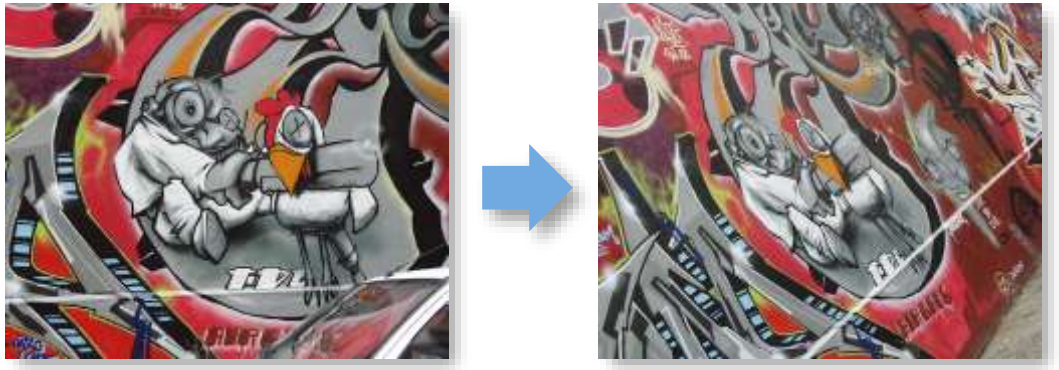
affine transformation

Projective Transformations 投影变换
aka Homographies 同构变换
aka Planar Perspective Maps 平面透视映射

$$\mathbf{H} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}$$

这种变换称为同构变换





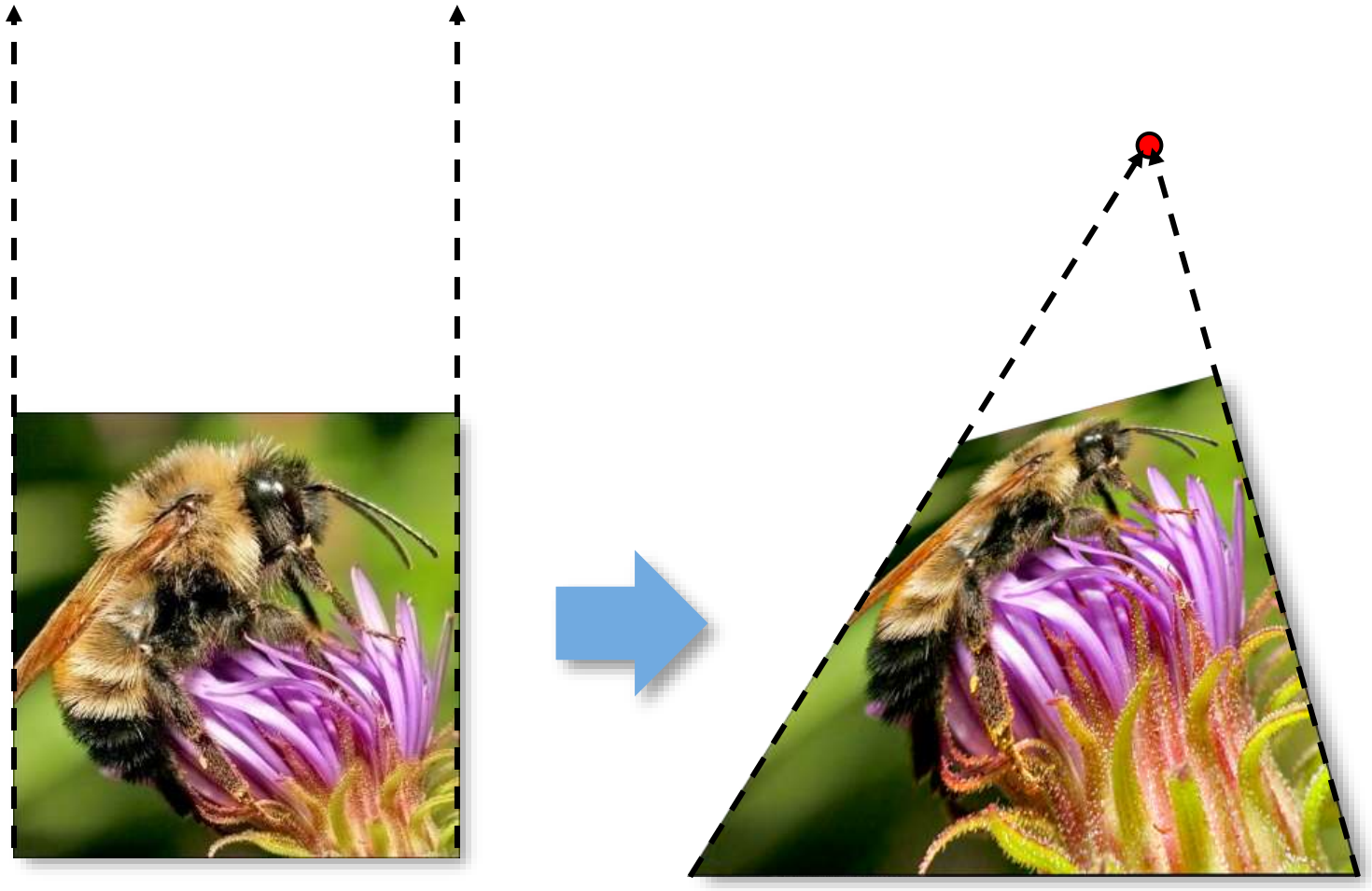
Homographies

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

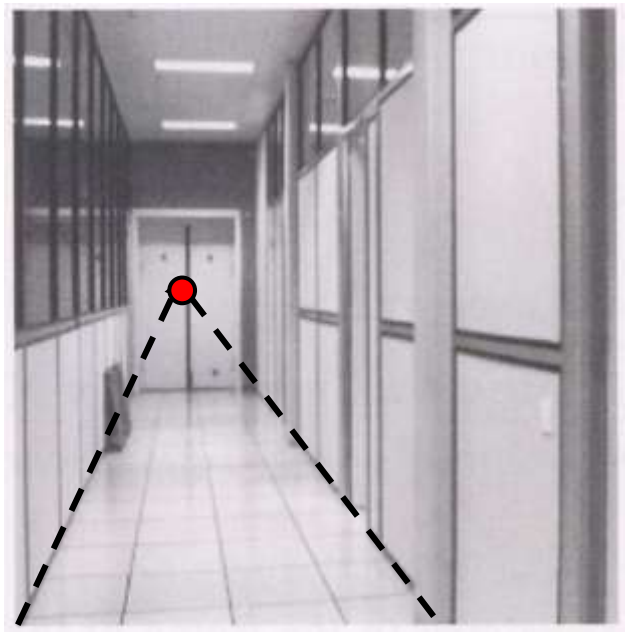
What happens when the denominator is 0?

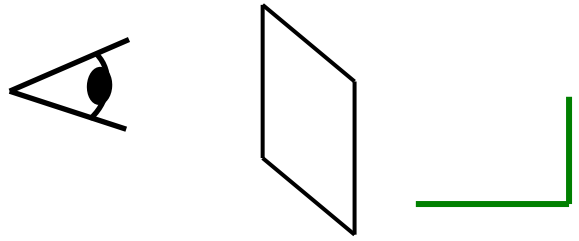
$$\sim \begin{bmatrix} \frac{ax+by+c}{gx+hy+1} \\ \frac{dx+ey+f}{gx+hy+1} \\ 1 \end{bmatrix}$$

Points at infinity



同构变换

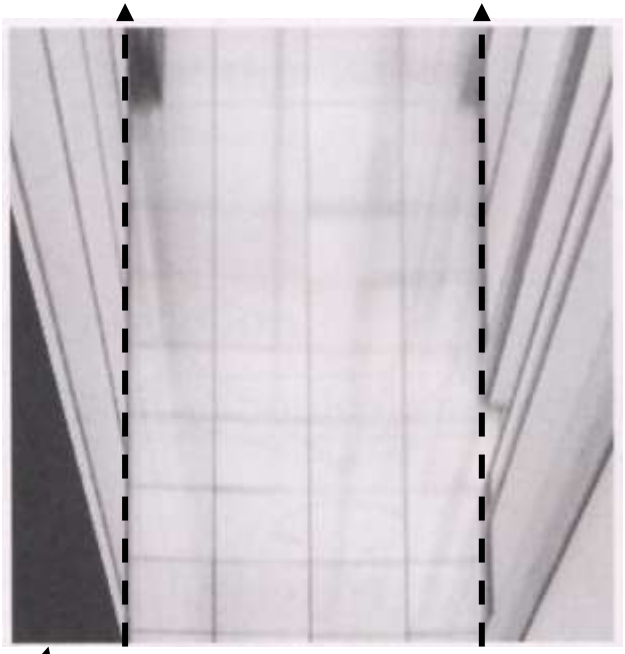




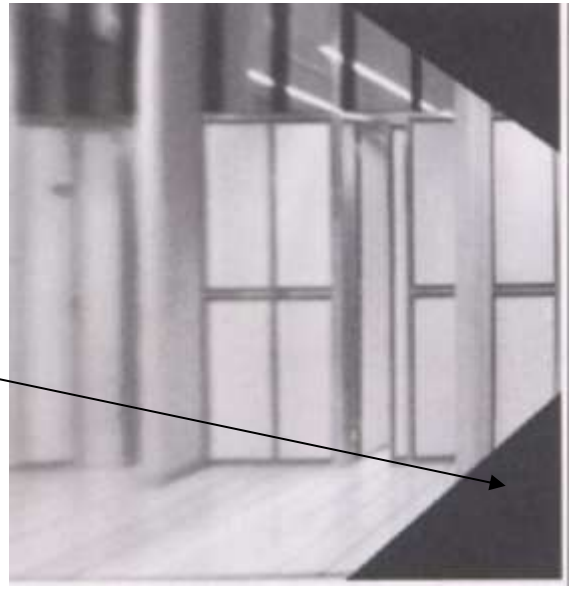
正面视角的图像

H_1

H_2



黑色区域
代表没有
像素



同构变换: 全景图是从不同视角采集的

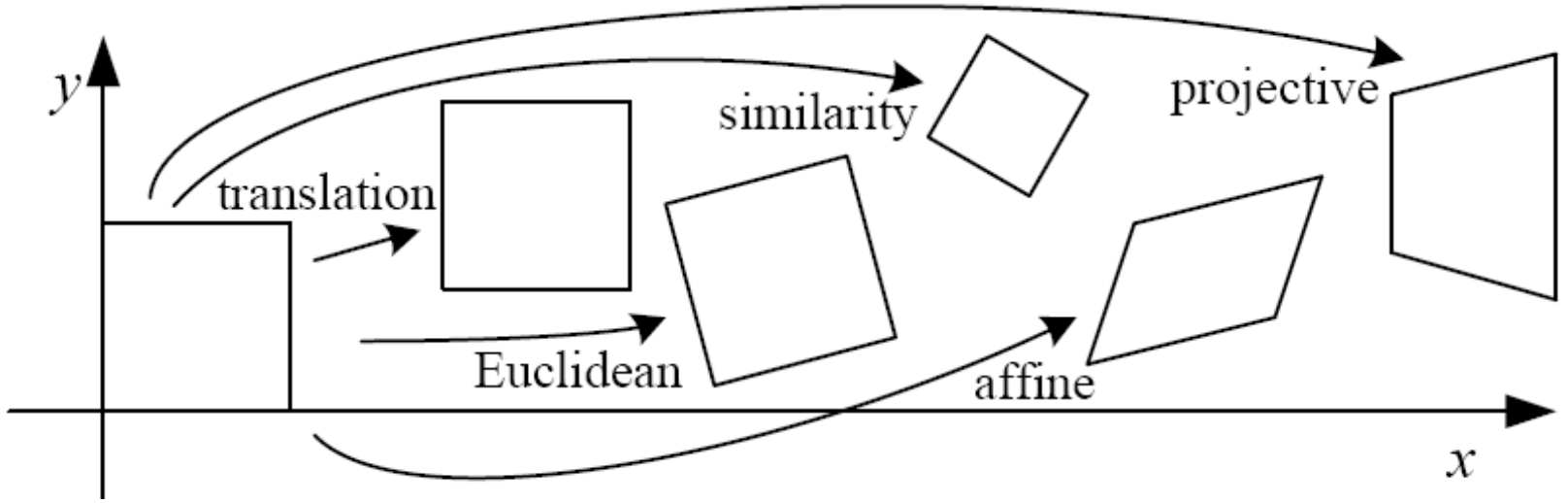
同构变换

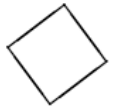
- 同构变换包括...
  - 仿射变换
  - 透视变换

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- 性质:
  - 原点变换后不一定是原点
  - 线仍然是线
  - 平行线不一定平行
  - 比例可能变化
  - 同构变换的组合还是同构变换

2D 图像变换



Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\left[\begin{array}{c\|c} \boldsymbol{I} & \boldsymbol{t} \end{array}\right]_{2\times3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\left[\begin{array}{c\|c} \boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times3}$	3	lengths $+\cdots$	
similarity	$\left[\begin{array}{c\|c} s\boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times3}$	4	angles $+\cdots$	
affine	$\left[\begin{array}{c} \boldsymbol{A} \end{array}\right]_{2\times3}$	6	parallelism $+\cdots$	
projective	$\left[\begin{array}{c} \tilde{\boldsymbol{H}} \end{array}\right]_{3\times3}$	8	straight lines