Introduction to Machine Learning (Basic Concepts)

CS5330, Huaizu Jiang

Fall 2021, Northeastern University

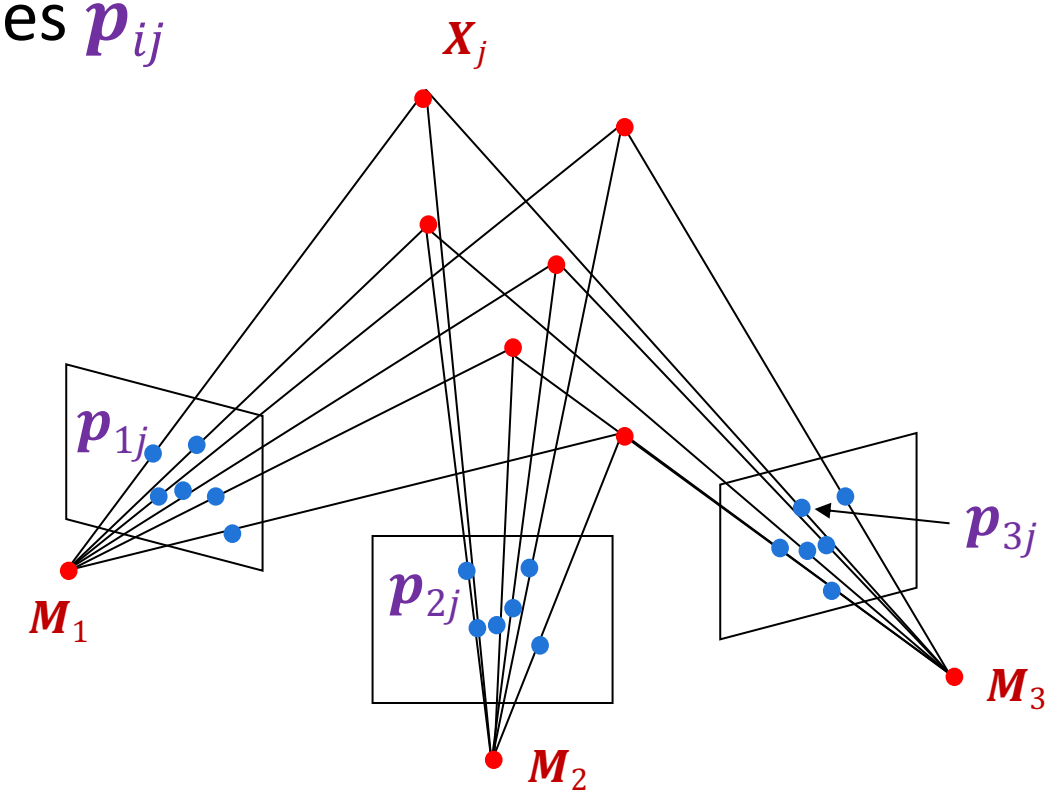
Before the class

- GPU resources for the final project
  - Discovery (you should have received notifications)
  - AWS
  - Google could
- Difficulty of pa3 vs pa2

Recap

Structure from motion: Problem formulation

- Given: m images of n fixed 3D points such that (ignoring visibility)
  - $\boldsymbol{p}_{ij} \equiv \boldsymbol{M}_i \boldsymbol{X}_j, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n$
- **Problem**: estimate m projection matrices $\boldsymbol{M}_i$ and n 3D points $\boldsymbol{X}_j$ from the mn correspondences $\boldsymbol{p}_{ij}$



Slide credit: S. Lazebnik

Structure from motion ambiguities

Not just limited to scale. Given:

$$\boldsymbol{p}_{ij} \equiv \boldsymbol{M}_i \boldsymbol{X}_j$$

Can insert any global transform **H**

$$\boldsymbol{p}_{ij} \equiv \boldsymbol{M}_i \boldsymbol{X}_j = \boldsymbol{M}_i \boldsymbol{H}^{-1} \boldsymbol{H} \boldsymbol{X}_j$$

H is a 3D homography / perspective transform / projective transform

Slide credit: D Fouhey & J Johnson

Affine structure from motion

General structure from motion:

$$\underset{3x1}{\boldsymbol{p}_{ij}} \equiv \underset{3x4}{\boldsymbol{M}_i} \underset{4x1}{\boldsymbol{X}_j}$$

Assume M is affine camera:

$$\underset{2x1}{\boldsymbol{p}_{ij}} = \underset{2x3}{\boldsymbol{A}_i} \underset{3x1}{\boldsymbol{X}_j} + \underset{2x1}{\boldsymbol{b}_i}$$

- **Problem**: use the mn 2D points $\boldsymbol{p}_{ij}$ to estimate m projection matrices $\boldsymbol{A}_i$ and translation vectors $\boldsymbol{b}_i$, and n 3D points $\boldsymbol{X}_j$

Using Normalized 2D Points

- After centering, each normalized 2D point $\hat{\boldsymbol{p}}_{ij}$ is related to the normalized 3D point by

$$\hat{\boldsymbol{p}}_{ij} = \boldsymbol{A}_i \hat{\boldsymbol{X}}_j$$

- We can get rid of the need to center the 3D data (and the translation ambiguity) by defining the origin of the world coordinate system as the centroid of the 3D points

Slide credit: S. Lazebnik

Affine structure from motion

Then, write all the equations in one in terms of product of cameras and points.

$$\widehat{p_{ij}} = A_i \hat{X}_j \qquad D = \begin{bmatrix} \widehat{p_{11}} & \cdots & \widehat{p_{1n}} \\ \vdots & \ddots & \vdots \\ \widehat{p_{m1}} & \cdots & \widehat{p_{mn}} \end{bmatrix} = \begin{bmatrix} A_1 \\ \vdots \\ A_m \end{bmatrix} \begin{bmatrix} X_1 & \cdots & X_n \end{bmatrix}$$

2m x n	2mx3	3xn
D	**M**	**S**

What's the rank of D?

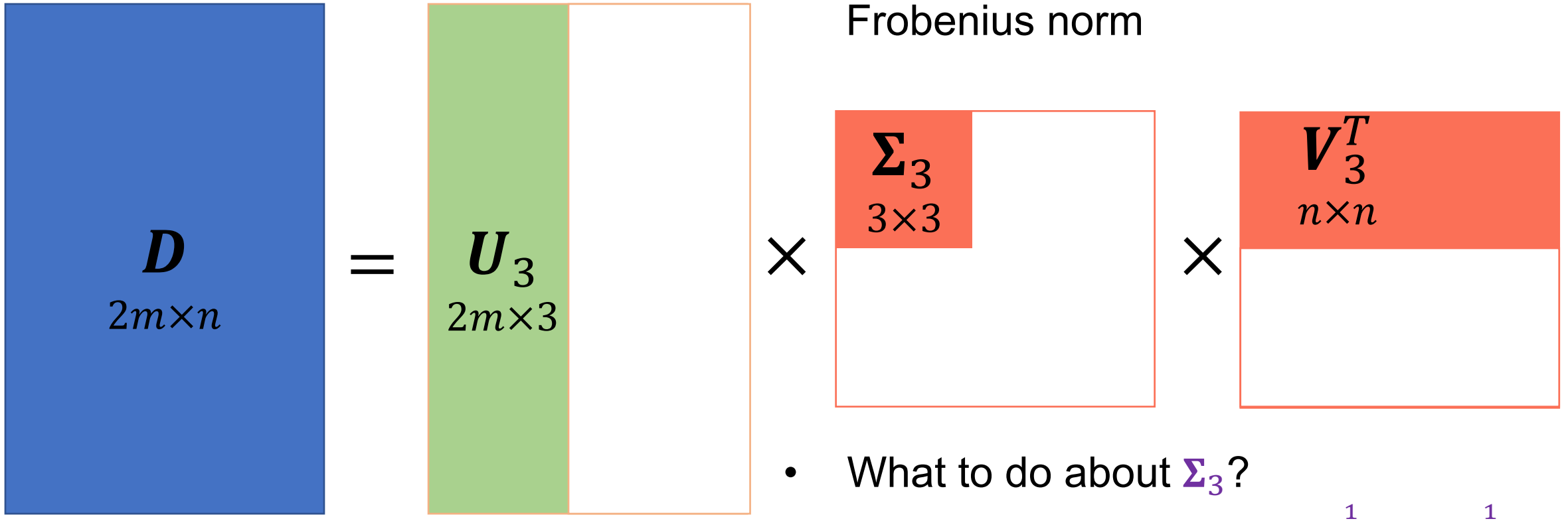
3!

C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. *IJCV*, 9(2):137-154, November 1992.

Slide credit: D Fouhey & J Johnson

Factorizing the measurement matrix

- Keep top 3 singular values:

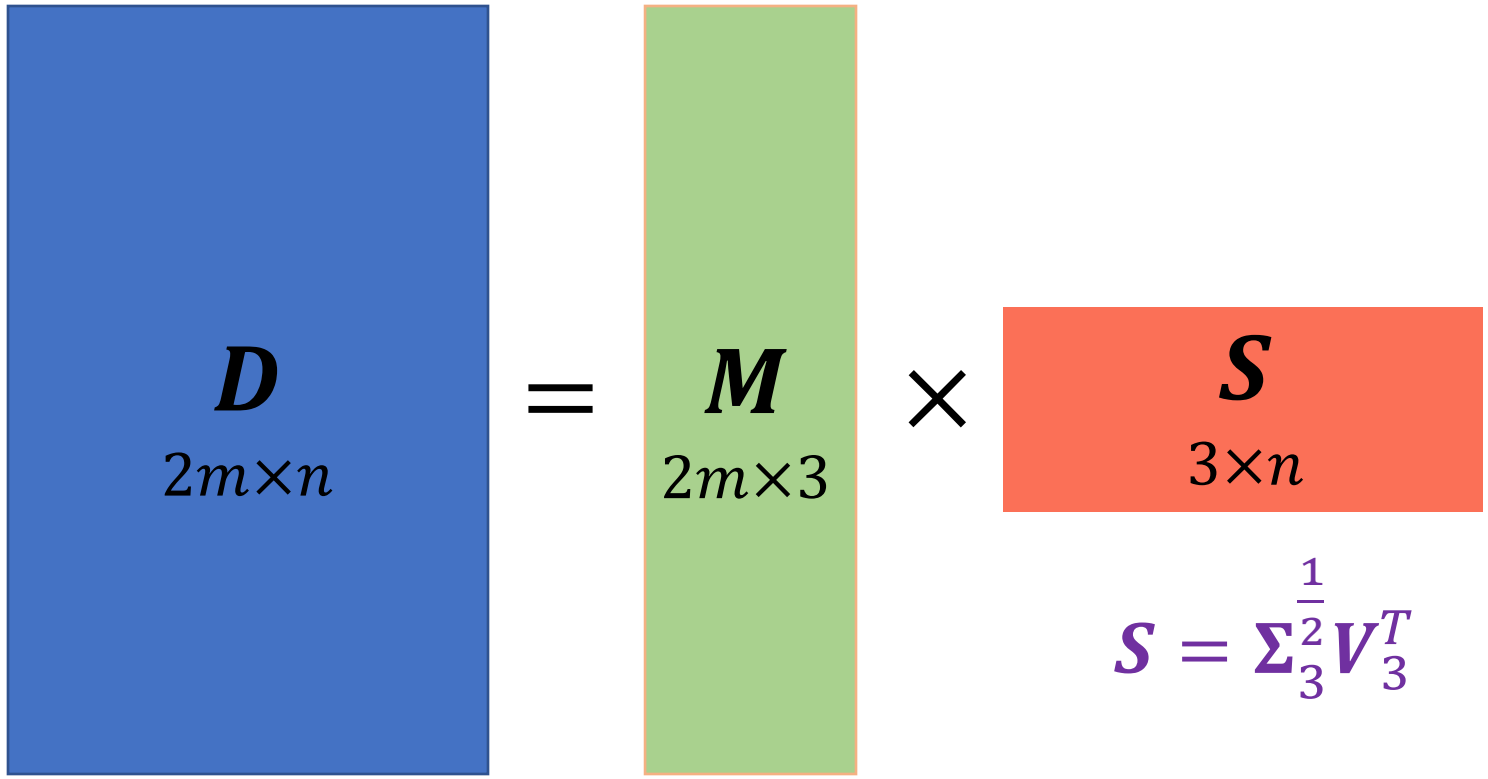


- This is the closest approximation of $\boldsymbol{D}$ with a rank-3 matrix in terms of Frobenius norm
- What to do about $\boldsymbol{\Sigma}_3$?
  - One solution: $\boldsymbol{M} = \boldsymbol{U}_3\boldsymbol{\Sigma}_3^{\frac{1}{2}}, \boldsymbol{S} = \boldsymbol{\Sigma}_3^{\frac{1}{2}}\boldsymbol{V}_3^T$

Slide credit: S. Lazebnik

Factorizing the measurement matrix

- One possible solution:



$$\boldsymbol{S} = \boldsymbol{\Sigma}_3^{\frac{1}{2}} \boldsymbol{V}_3^T$$

$$\boldsymbol{M} = \boldsymbol{U}_3 \boldsymbol{\Sigma}_3^{\frac{1}{2}}$$

- Are there other solutions?

Slide credit: S. Lazebnik

Projective structure from motion

- **Given**: m images of n fixed 3D points such that (ignoring visibility):

$$\bullet\ \boldsymbol{p}_{ij} \equiv \boldsymbol{M}_i \boldsymbol{X}_j, \qquad i = 1, \ldots, m, \quad j = 1, \ldots, n$$

- **Problem**: estimate m projection matrices $\boldsymbol{M}_i$ and n 3D points $\boldsymbol{X}_j$ from the mn correspondences $\boldsymbol{p}_{ij}$
- With no calibration info, cameras and points can only be recovered up to a 4×4 projective transformation $\boldsymbol{Q}$:

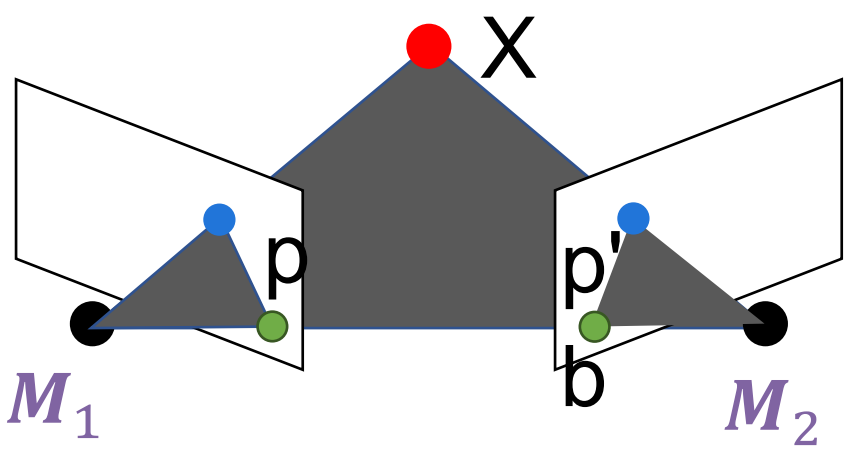
$$\bullet\ \boldsymbol{X} \rightarrow \boldsymbol{Q}\boldsymbol{X}, \ \boldsymbol{M} \rightarrow \boldsymbol{M}\boldsymbol{Q}^{-1}$$

Slide credit: S. Lazebnik

Two Camera Case

For two cameras, we need 7 points.

What else (in theory) requires 7 points?



Compute fundamental matrix **F** and epipole **b** s.t. $\mathbf{F}^T\mathbf{b} = 0$. Then:

$$M_1 = [I, 0]$$

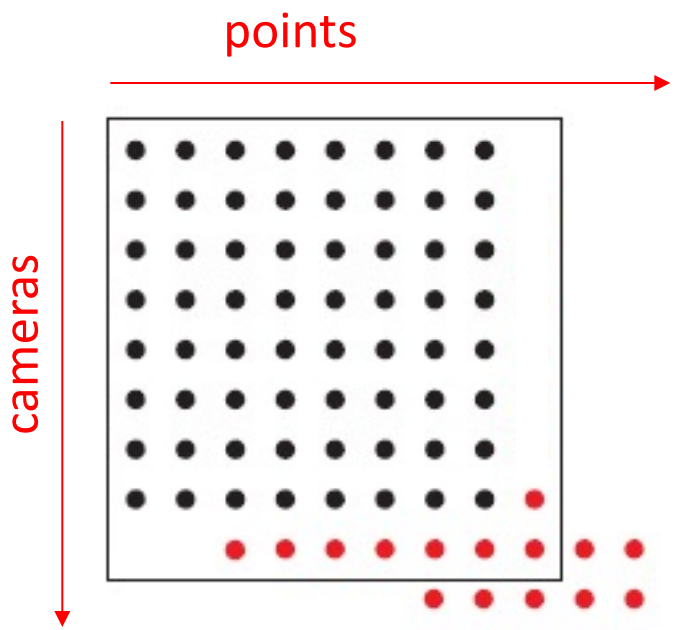
$$M_2 = [-[b_x]F, b]$$

Remember: this is up to a projective ambiguity!

Slide credit: D Fouhey & J Johnson

Incremental structure from motion

- Initialize motion from two images using fundamental matrix
- Initialize structure by triangulation
- For each additional view:
  - Determine projection matrix of new camera using all the known 3D points that are visible in its image – calibration
  - Refine and extend structure: compute new 3D points, re-optimize existing points that are also seen by this camera – triangulation
- Refine structure and motion: bundle adjustment



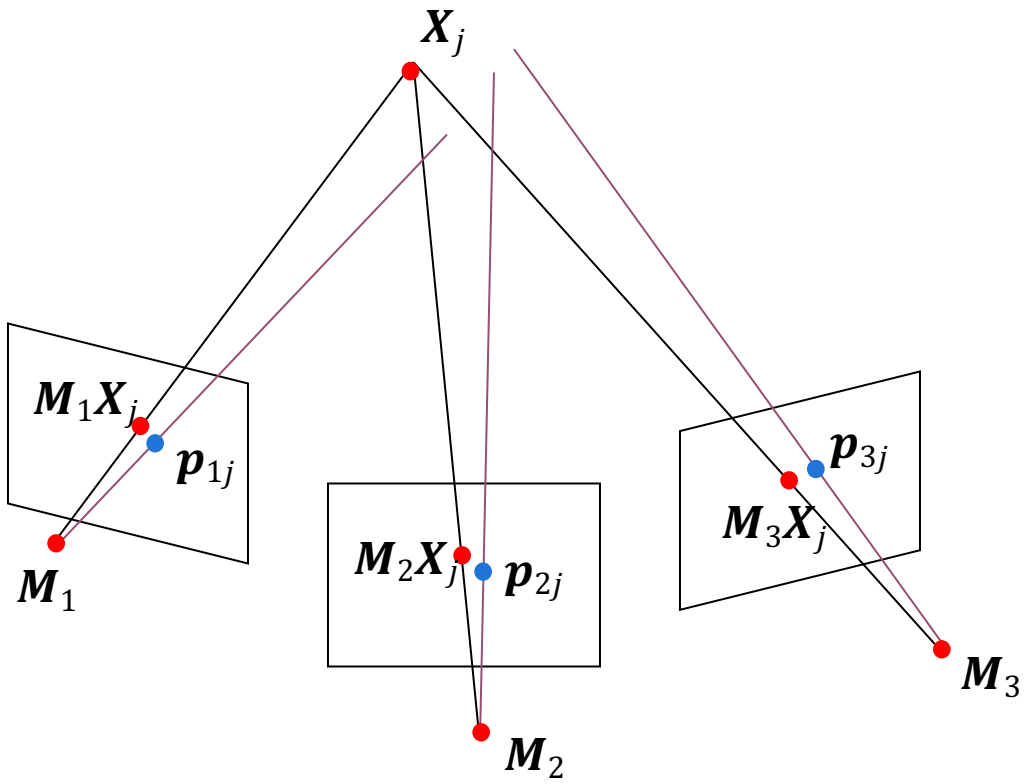
Slide credit: S. Lazebnik

Bundle adjustment

- Non-linear method for refining structure and motion
- Minimize reprojection error (with lots of bells and whistles):

$$\sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij}\, d\left(\boldsymbol{p}_{ij} - \boldsymbol{M}_i \boldsymbol{X}_j\right)^2$$

visibility flag: is point j visible in view i?



B. Triggs et al. Bundle adjustment – A modern synthesis. International Workshop on Vision Algorithms, 1999

Slide credit: S. Lazebnik

Today's class

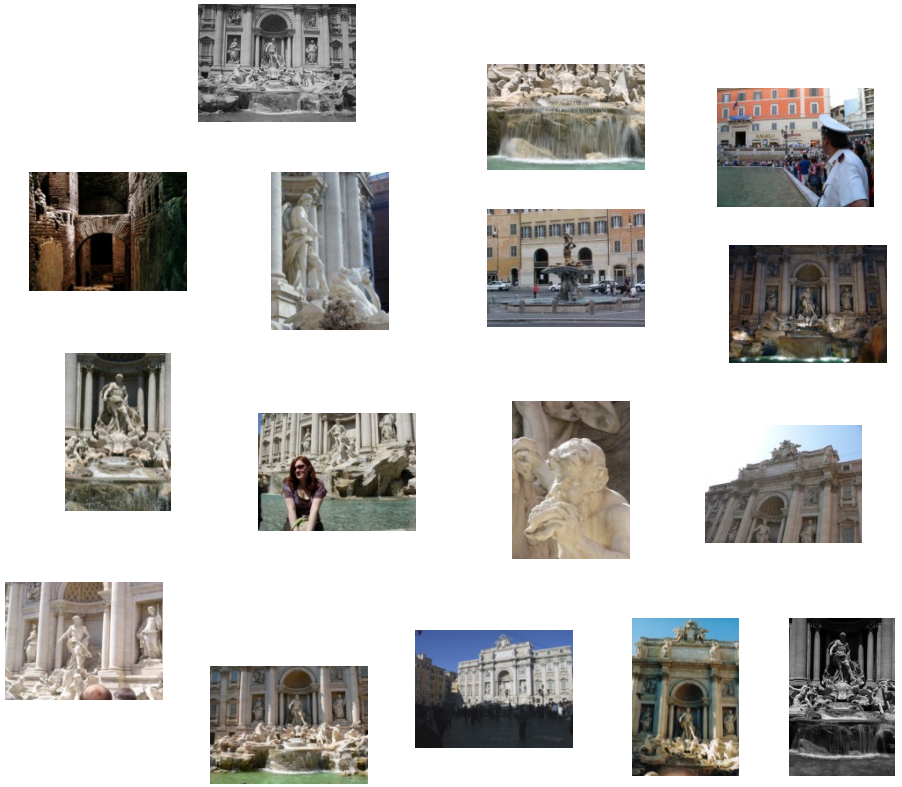
- A bit more about Structure from motion (SfM)
- Introduction to machine learning
  - What is machine learning
  - A simple machine learning example
  - Cross validation

Representative SFM pipeline

N. Snavely, S. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3D. SIGGRAPH 2006
http://phototour.cs.washington.edu/

Feature detection

Detect SIFT features

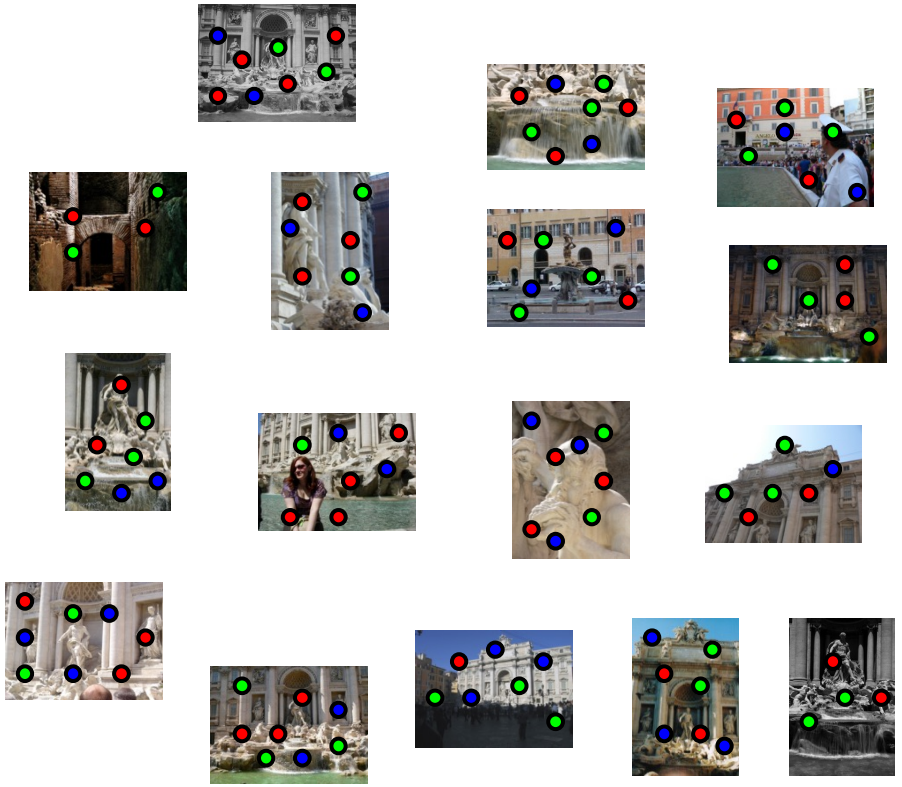


Source: N. Snavely

Feature detection

Detect SIFT features

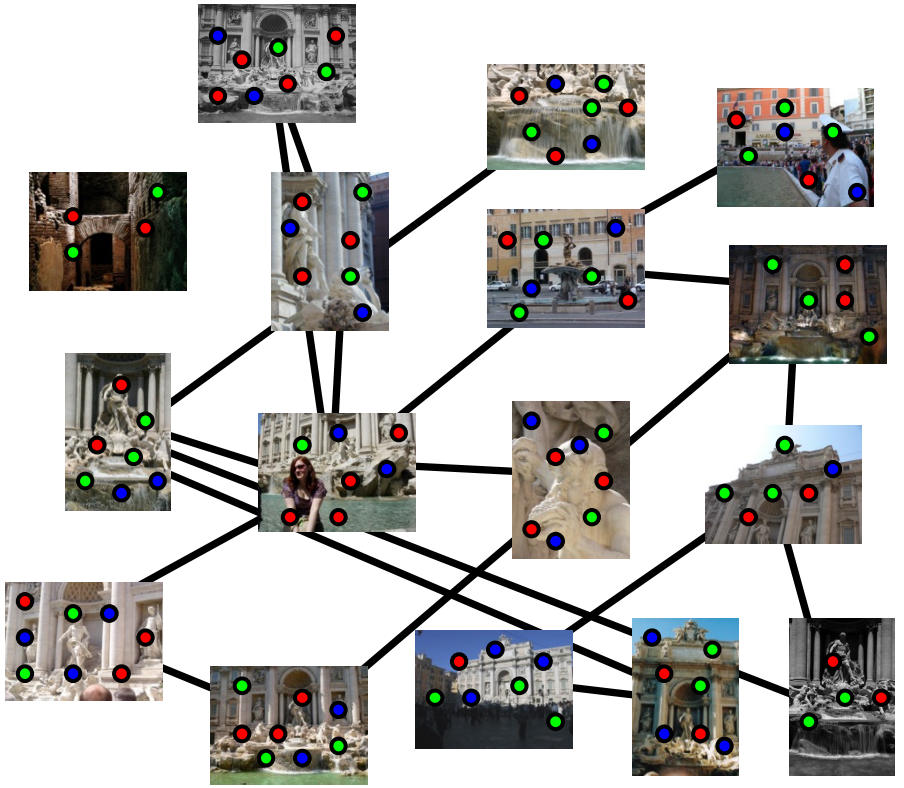
Other popular feature types: SURF, ORB, BRISK, …



Source: N. Snavely

Feature matching

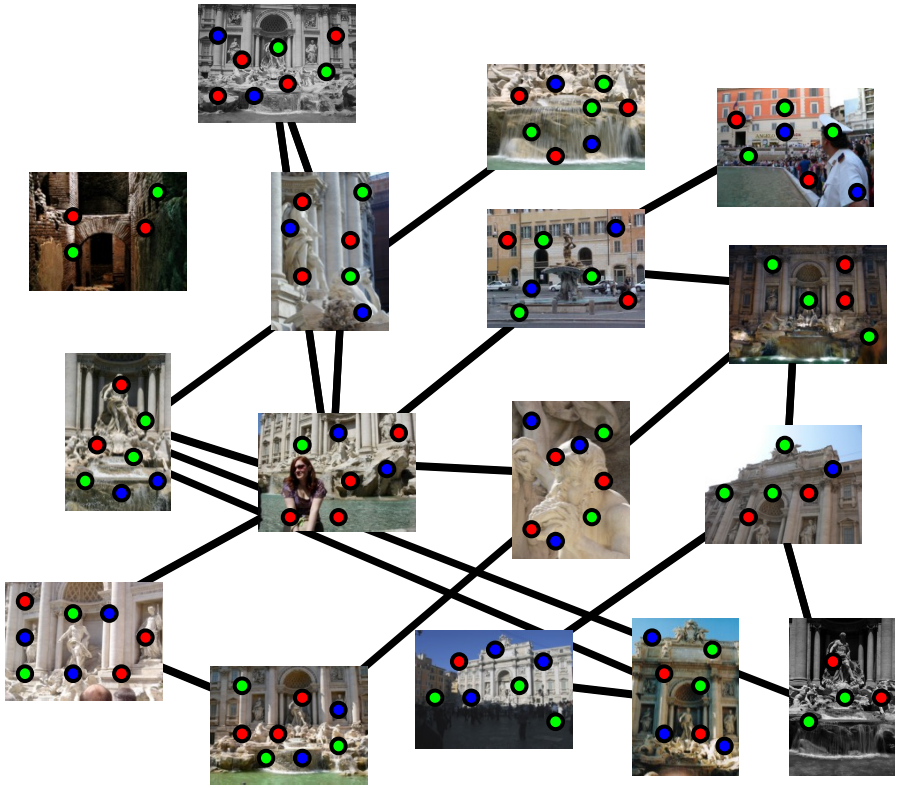
Match features between each pair of images



Source: N. Snavely

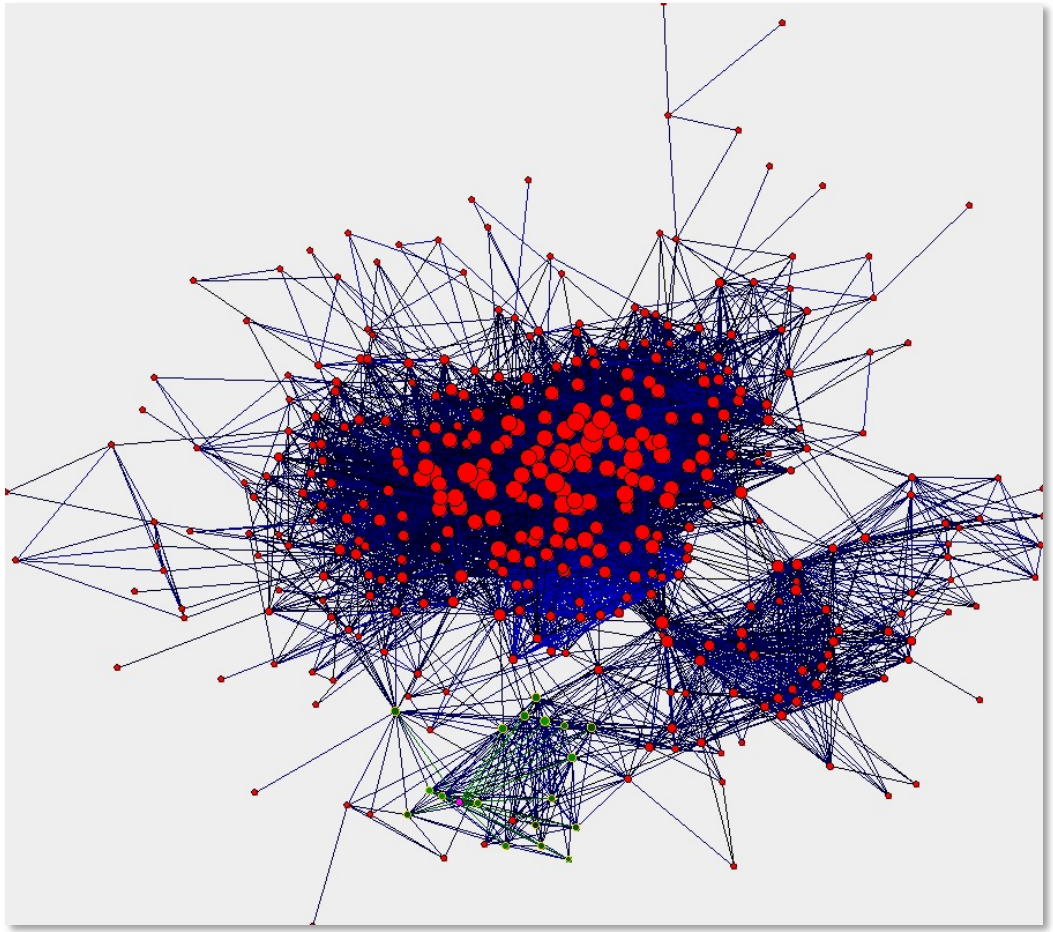
Feature matching

Use RANSAC to estimate fundamental matrix between each pair



Source: N. Snavely

Image connectivity graph



(graph layout produced using the Graphviz toolkit: http://www.graphviz.org/)

Source: N. Snavely

Incremental SFM

- Pick a pair of images with lots of inliers (and preferably, good EXIF data)
  - Initialize intrinsic parameters (focal length, principal point) from EXIF
  - Estimate extrinsic parameters ($\boldsymbol{R}$ and $\boldsymbol{t}$) using five-point algorithm
  - Use triangulation to initialize model points
- While remaining images exist
  - Find an image with many feature matches with images in the model
  - Run RANSAC on feature matches to register new image to model
  - Triangulate new points
  - Perform bundle adjustment to re-optimize everything
  - Optionally, align with GPS from EXIF data or ground control points

Slide credit: D Fouhey & J Johnson

SfM applications

- 3D modeling
- Surveying
- Robot navigation and mapmaking (including autonomous driving, SLAM)
- Virtual and augmented reality
- Visual effects (“Match moving”)
  - https://www.youtube.com/watch?v=RdYWp70P_kY

Slide credit: S. Lazebnik

Applications: Visual Reality & Augmented Reality & Metaverse

Oculus

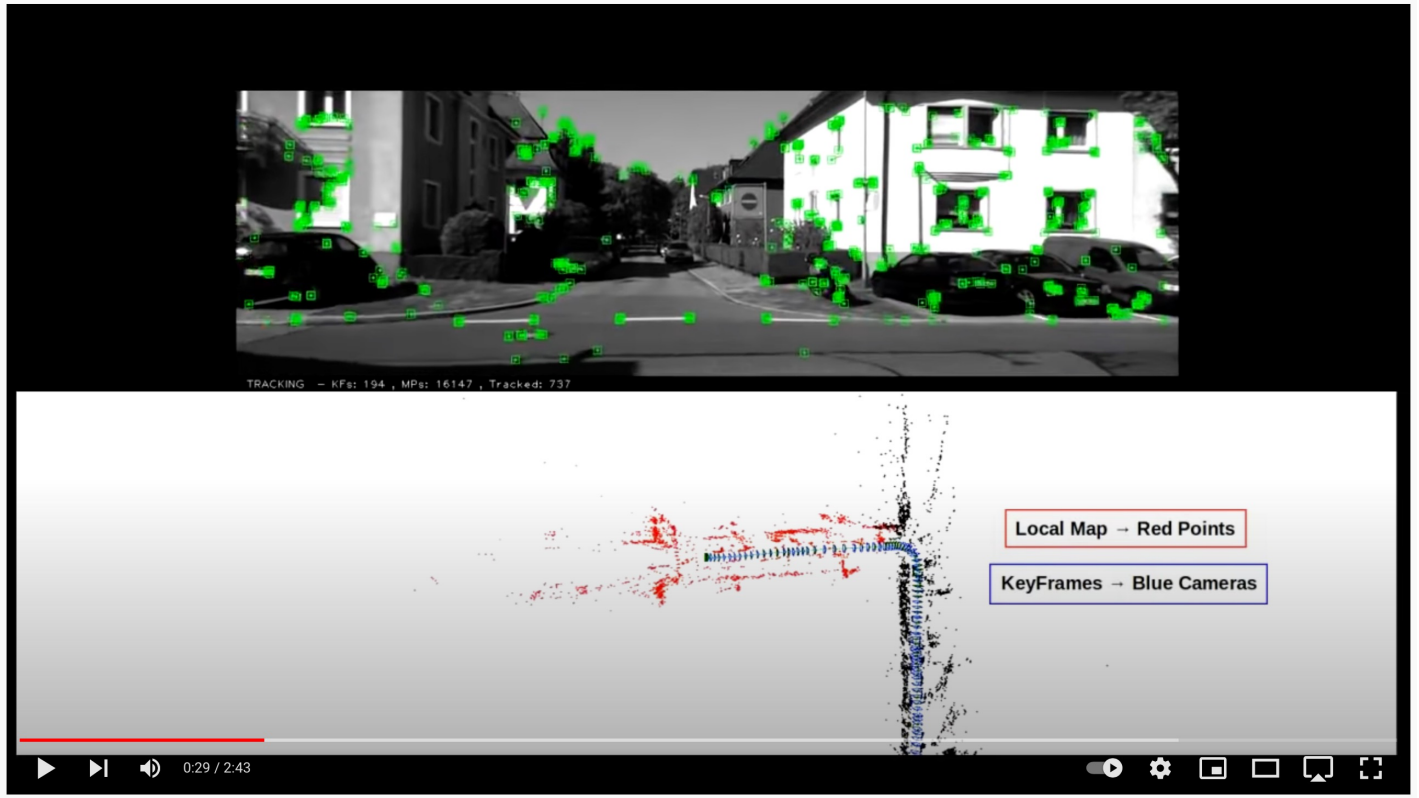
https://www.youtube.com/watch?v=KOG7yTz1iTA

Hololens

https://www.youtube.com/watch?v=FMtvrTGnP04

Slide credit: N. Snavely

Applications: Robot Navigation (Autonomous Driving)



ORBSLAM

https://www.youtube.com/watch?v=8DISRmsO2YQ

Slide credit: N. Snavely

Application: AR walking directions

https://www.theverge.com/2019/8/8/20776247/google-maps-live-view-ar-walking-directions-ios-android-feature

Slide credit: N. Snavely

The rest of this course

	Fundamentals of Deep Neural Networks	
	Mon Oct 25	Basic Concepts of Machine Learning
Week 7	Thu, Oct 28	Optimization: Stochastic Gradient Descent and Backpropagation
	Mon, Nov 1	Neural Networks 1
Week 8	Thu, Nov 4	Neural Networks 2
	Mon, Nov 8	Convolutional Neural Networks 1
Week 9	Thu, Nov 11	No class, Veterans' Day
	Mon, Nov 15	Convolutional Neural Networks 2
	Recognition, Detection and Segmentation via DNNs	
Week 10	Thur, Nov 18	Object Detection 1: R-CNN
	Mon, Nov 22	Object Detection 2: Fast and Faster R-CNN
Week 11	Thur, Nov 25	No class, Thanksgiving recess
	Mon, No 29	Segmentation 1: Mask R-CNN, FCN
Week 12	Thur, Dec 2	Segmentation 2: Panoptic Segmentation
	Advanced Topics	
	Mon, Dec 6	Novel view synthesis and video frame interpolation
Week 13	Thur, Dec 9	Vision and language (VQA, visual grounding, etc)
	Mon, Dec 13	Learning with unlabeled data
Week 14	Thur, Dec 16	Transformers in computer vision

guest lecture

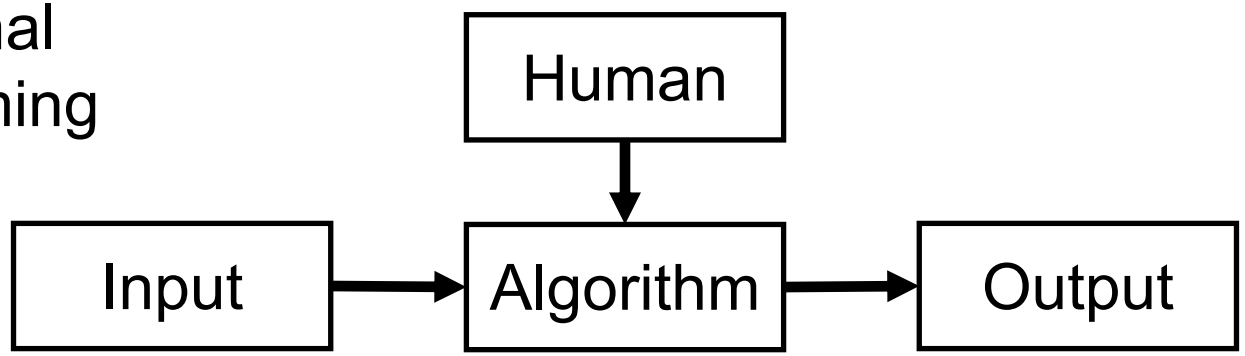
Machine Learning

Algorithms that learn from data

Slide credit: D Fouhey & J Johnson

Machine Learning vs Programming

Traditional Programming



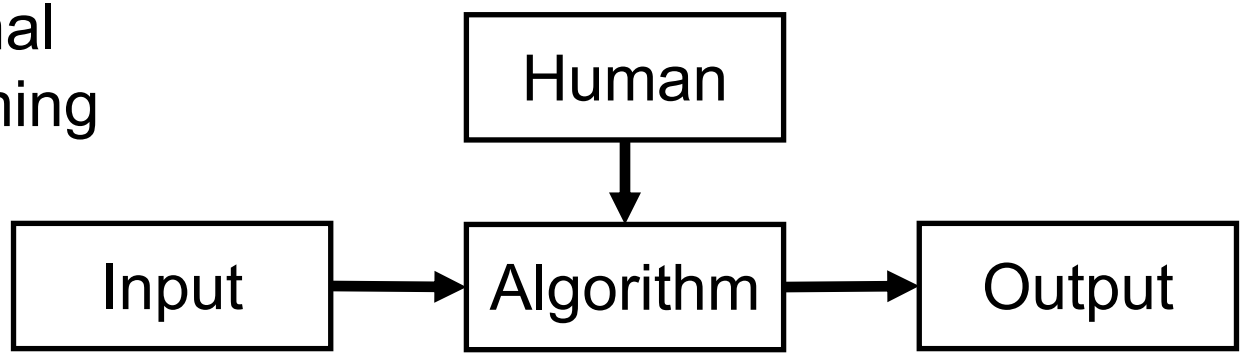
Works well for sorting numbers

```
def bubble_sort(arr):
  N = len(arr)
  for i in range(N - 1):
    for j in range(N - i - 1):
      if arr[j + 1] < arr[j]:
        temp = arr[j]
        arr[j] = arr[j + 1]
        arr[j + 1] = temp
  return arr
```

Slide credit: D Fouhey & J Johnson

Machine Learning vs Programming

Traditional Programming



Much harder for some problems

```
def cat_or_dog(image):
  if ????:
    return "cat"
  else:
    return "dog"
```

Slide credit: D Fouhey & J Johnson

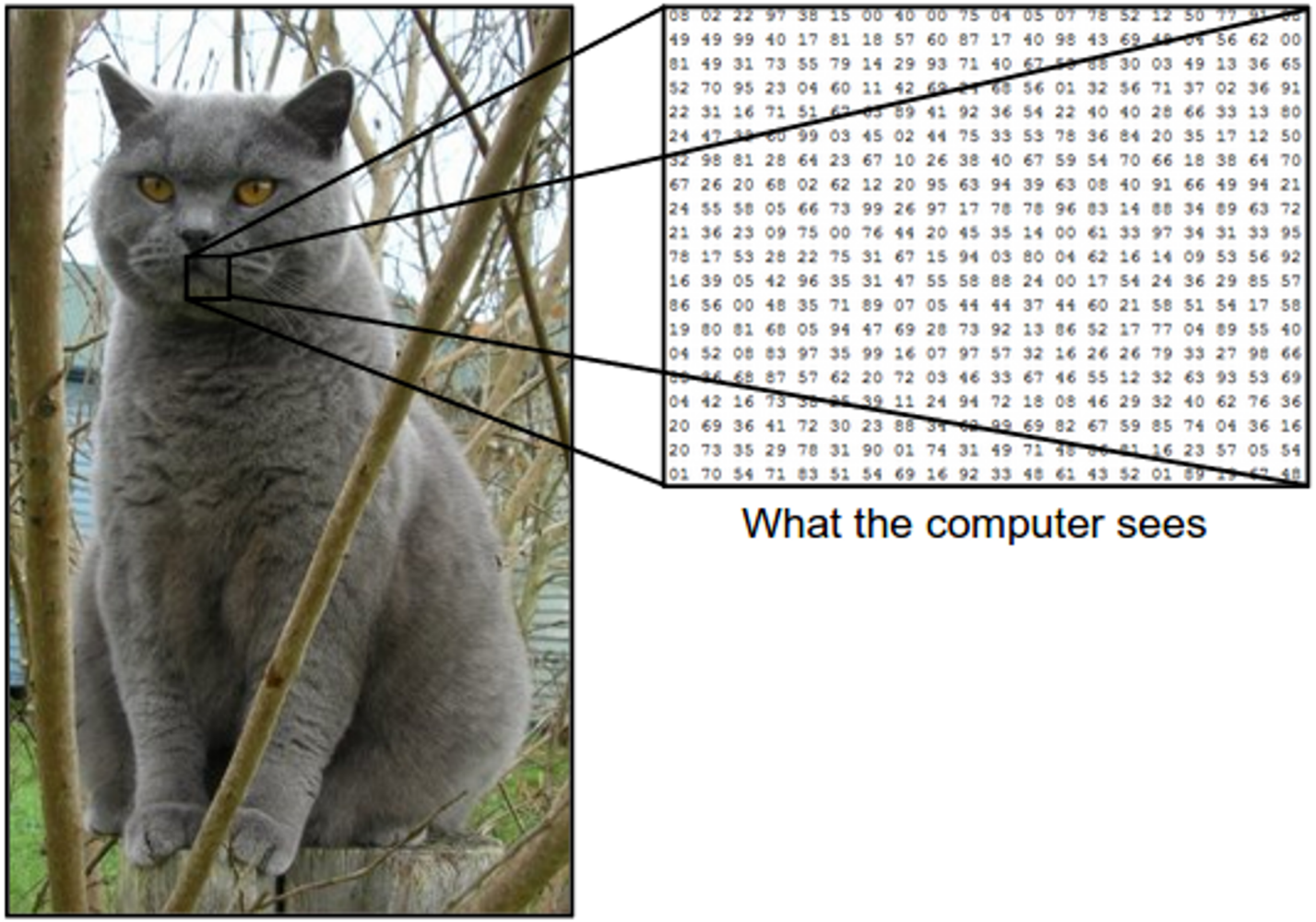
Image Classification: a core task in Computer Vision

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

cat

Slide credit: E Learned-Miller, originally developed by A Karpathy, J Johnson, Fei-Fei L

The problem:
semantic gap

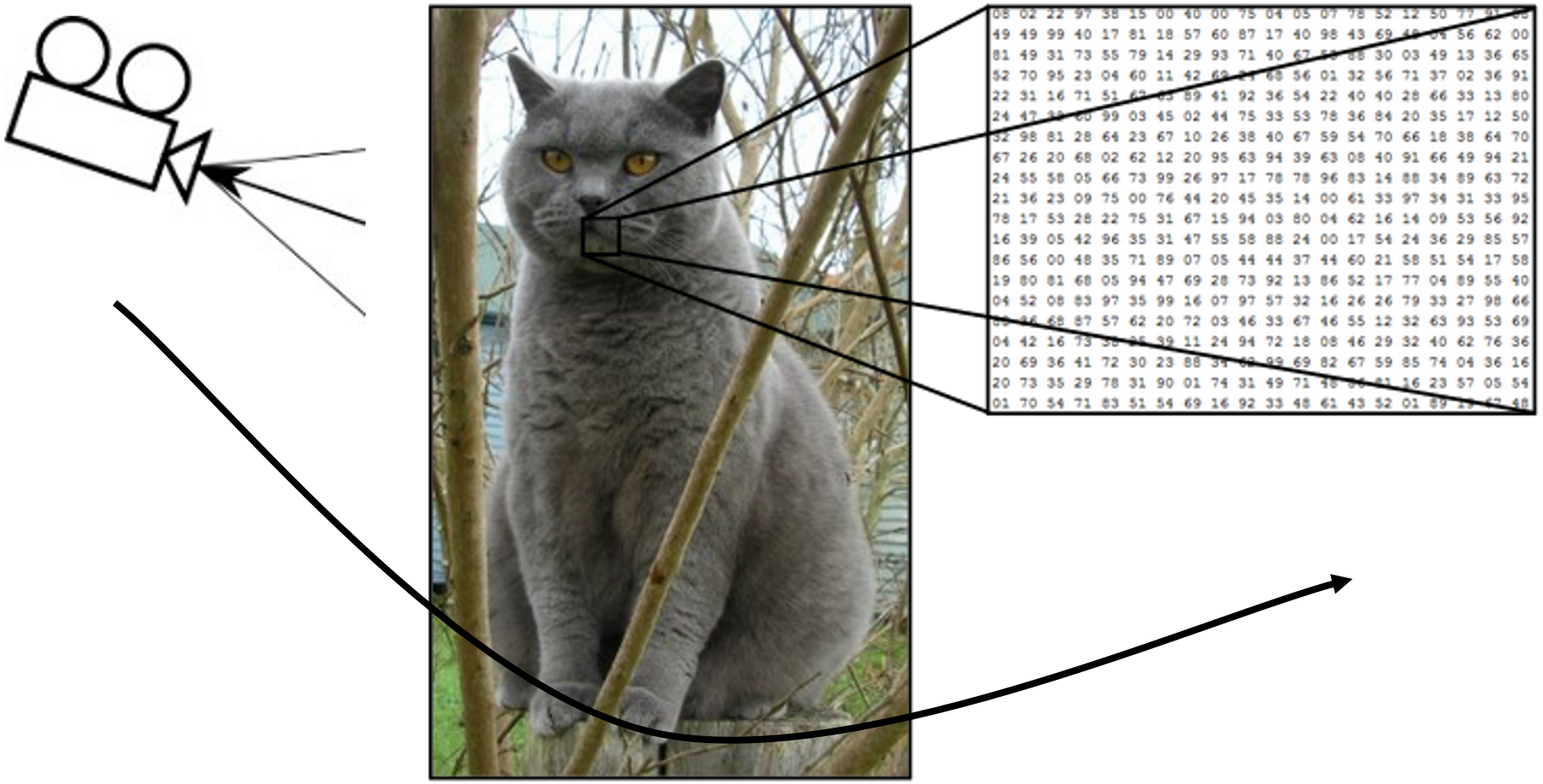


Images are represented as 3D arrays of numbers, with integers between [0, 255].

E.g.
300 x 100 x 3

Slide credit: E Learned-Miller, originally developed by A Karpathy, J Johnson, Fei-Fei L

Challenges: Viewpoint Variation

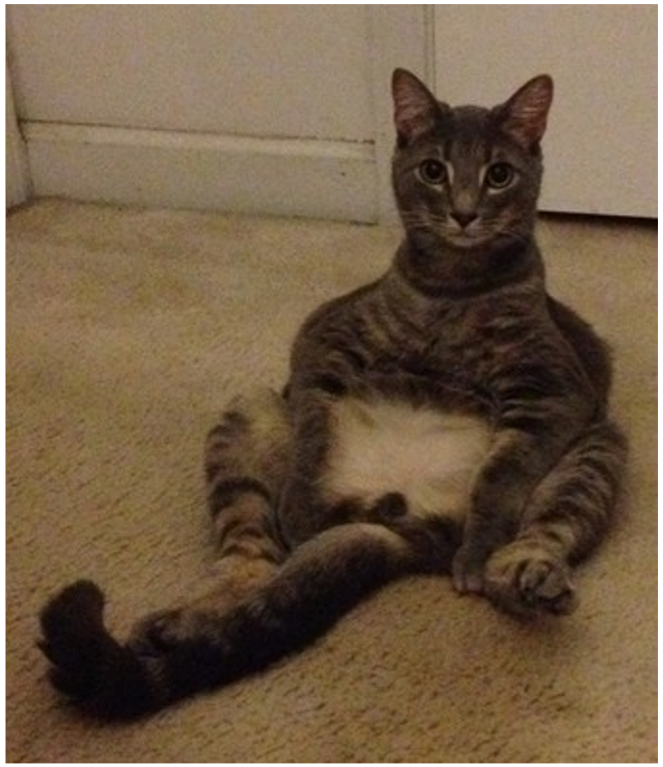


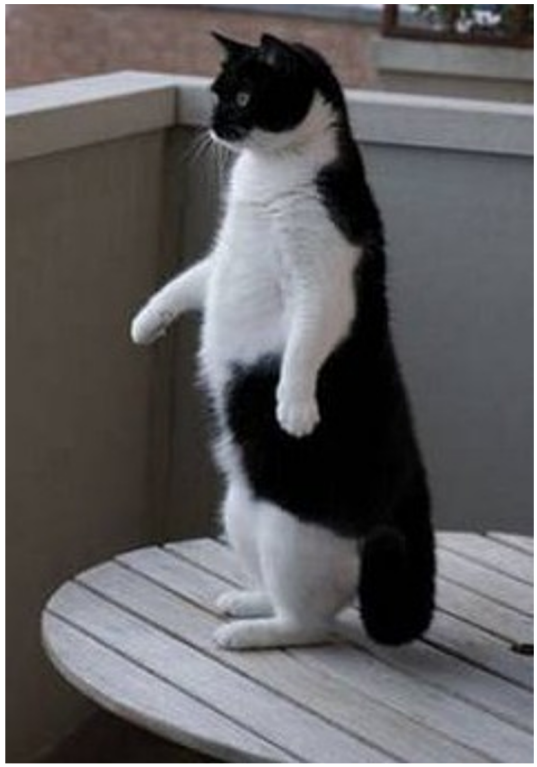
Slide credit: E Learned-Miller, originally developed by A Karpathy, J Johnson, Fei-Fei L

Challenges: Illumination

Slide credit: E Learned-Miller, originally developed by A Karpathy, J Johnson, Fei-Fei L

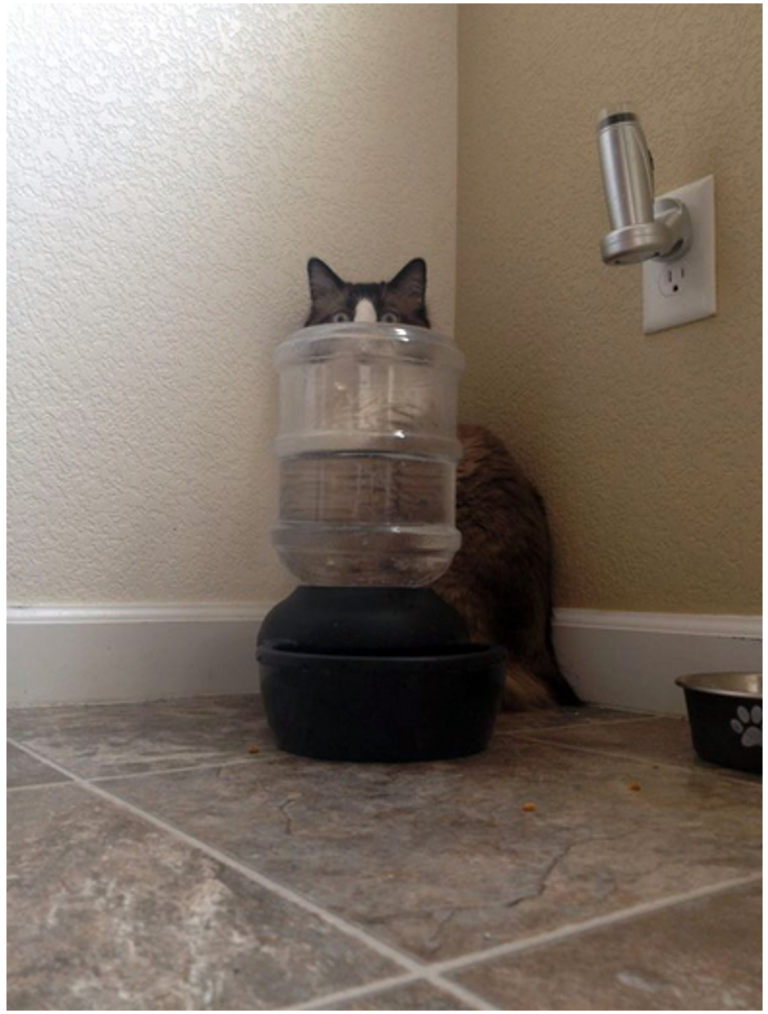
Challenges: Deformation



Slide credit: E Learned-Miller, originally developed by A Karpathy, J Johnson, Fei-Fei L

Challenges: Occlusion



Slide credit: E Learned-Miller, originally developed by A Karpathy, J Johnson, Fei-Fei L

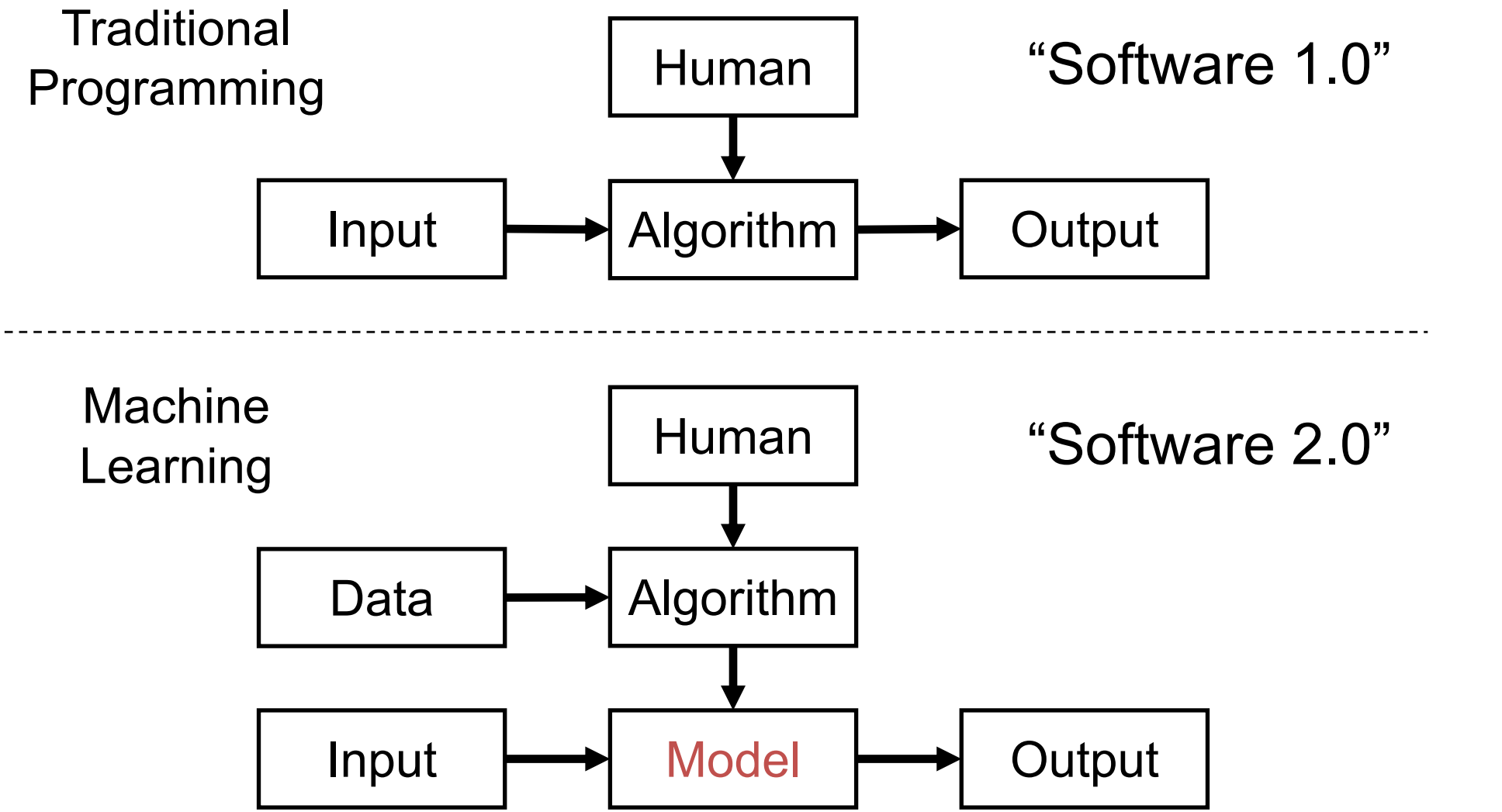
Challenges: Background clutter

Slide credit: E Learned-Miller, originally developed by A Karpathy, J Johnson, Fei-Fei L

Challenges: Intraclass variation

Slide credit: E Learned-Miller, originally developed by A Karpathy, J Johnson, Fei-Fei L

Machine Learning vs Programming



https://medium.com/@karpathy/software-2-0-a64152b37c35

Slide credit: D Fouhey & J Johnson

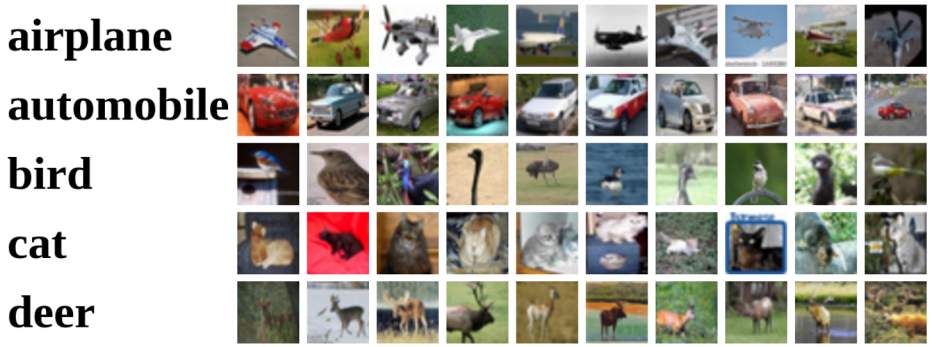
Machine Learning: Data-Driven Approach

1. Collect a large set of data
2. Use Machine Learning to train a model
3. Evaluate the model on new data

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

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

Example training set



Slide credit: D Fouhey & J Johnson

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)

x is input / feature

y is label / target

Goal: Learn a *function* to map x -> y

Slide credit: D Fouhey & J Johnson

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)

x is input / feature

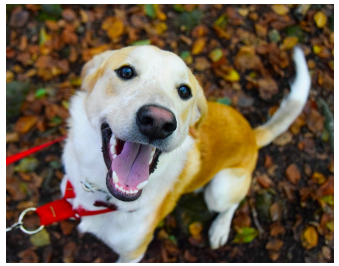
y is label / target

Goal: Learn a *function* to map x -> y

Image Classification:
Predict a discrete category

x y

→ Cat



→ Dog

→ Monkey

Slide credit: D Fouhey & J Johnson

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)

x is input / feature

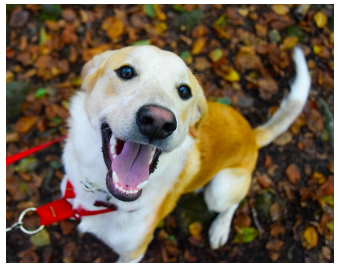
y is label / target

Goal: Learn a *function* to map x -> y

Image Regression:
Predict a continuous value

x y

→ 2 lbs



→ 25 lbs

→ 35 lbs

Slide credit: D Fouhey & J Johnson

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)

x is input / feature

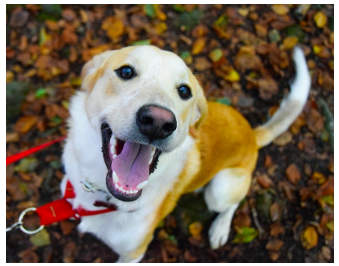
y is label / target

Goal: Learn a *function* to map x -> y

Image Captioning:
Predict a sequence of words

x y

“A white and gray kitten on grass”



“White and orange dog with a red leash in the woods”

“A monkey sitting in front of rocks”

Slide credit: D Fouhey & J Johnson

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)

x is input / feature

y is label / target

Goal: Learn a *function* to map x -> y

Unsupervised Learning

Data: x

Just data, no labels!

Goal: Learn underlying *structure* in the data

Slide credit: D Fouhey & J Johnson

Supervised vs Unsupervised Learning

Clustering:
Group similar images

Cat image is CC0 public domain
Dog image is CC0 Public Domain
Monkey image is CC0 Public Domain

Cat image is CC0 public domain
Monkey Image is CC0 public domain

Cat image is CC0 public domain
Dog image is CC0 public domain
Dog image is CC0 public domain

Unsupervised Learning

Data: x

Just data, no labels!

Goal: Learn underlying *structure* in the data

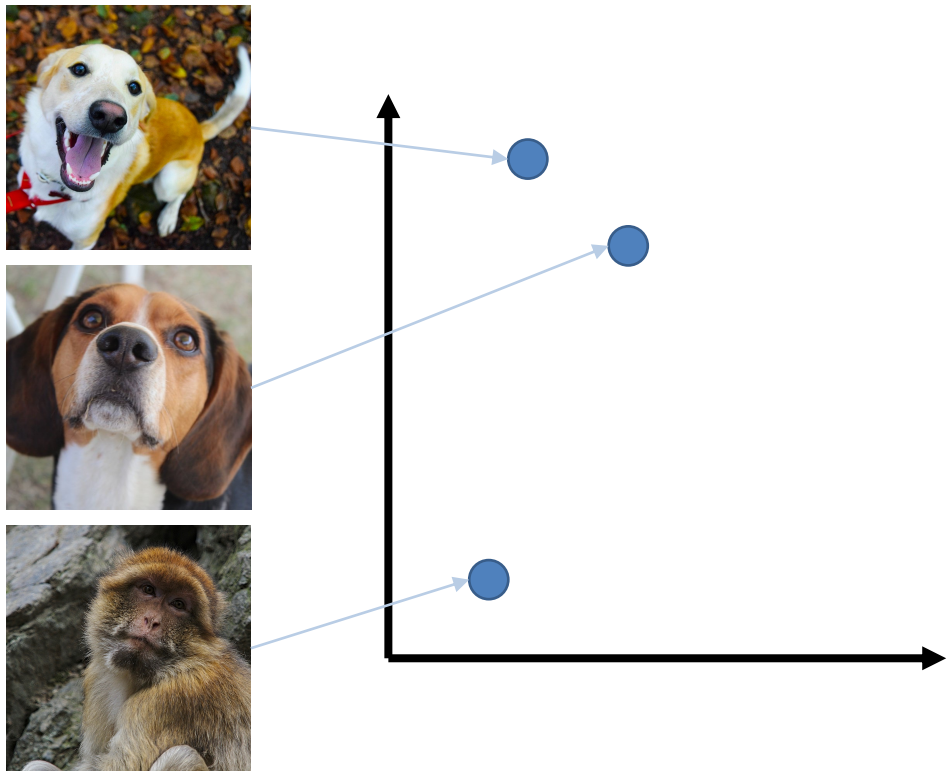
Slide credit: D Fouhey & J Johnson

Supervised vs Unsupervised Learning

Dimensionality Reduction:
Project to subspace

Images:
256x256x3

Projections:
2-dimensional



Unsupervised Learning

Data: x

Just data, no labels!

Goal: Learn underlying *structure* in the data

Slide credit: D Fouhey & J Johnson

ML Problems in Vision

	Supervised (Inputs+Labels)	**Unsupervised (Just Data)**
Discrete Output	Classification/ Categorization (Object detection, image segmentation)	Clustering
Continuous Output	Regression (stereo disparity, optical flow)	Dimensionality Reduction

Slide adapted from J. Hays

First Machine Learning Algorithm:

Least Squares Linear Regression

Slide credit: D Fouhey & J Johnson

Least Squares Linear Regression

Slide credit: D Fouhey & J Johnson

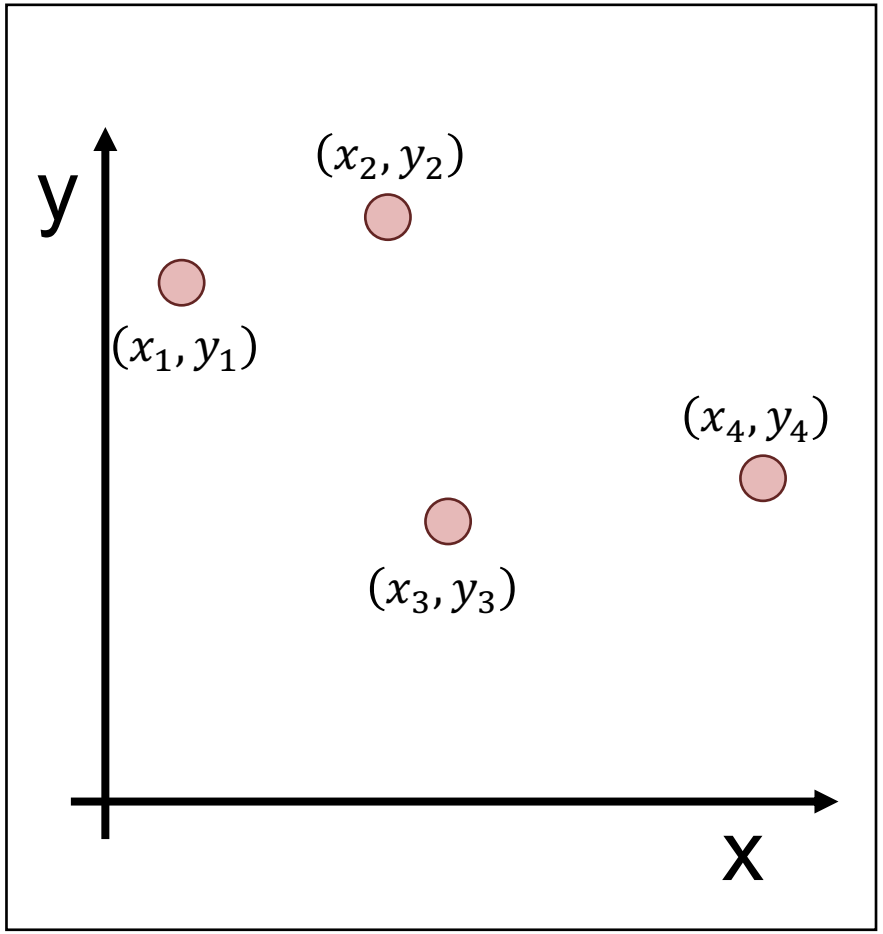
Least Squares Linear Regression

"Regression" = supervised learning with continuous outputs

Data:

$$(x_1, y_1), (x_2, y_2), \ldots (x_N, y_N)$$

$$x_i, y_i \in \mathbb{R}$$



Slide credit: D Fouhey & J Johnson

Least Squares Linear Regression

"Linear" = Our model is a line

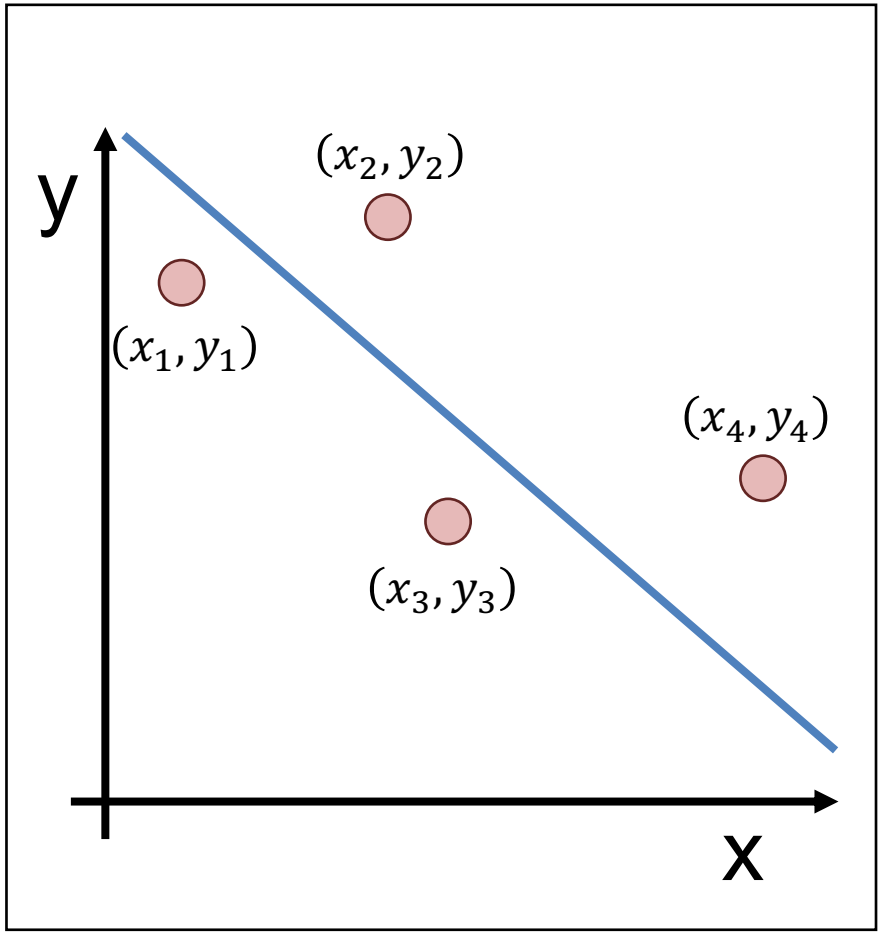
Data:

$$(x_1, y_1), (x_2, y_2), \ldots (x_N, y_N)$$
$$x_i, y_i \in \mathbb{R}$$

Model: $y = mx + b$

Or: $\boldsymbol{x} = (x, 1); \boldsymbol{w} = (m, b)$

$y = \boldsymbol{w} \cdot \boldsymbol{x}$



Slide credit: D Fouhey & J Johnson

Least Squares Linear Regression

"Least squares" = Find the line that minimizes squared error

Data:

$$(x_1, y_1), (x_2, y_2), \ldots (x_N, y_N)$$
$$x_i, y_i \in \mathbb{R}$$

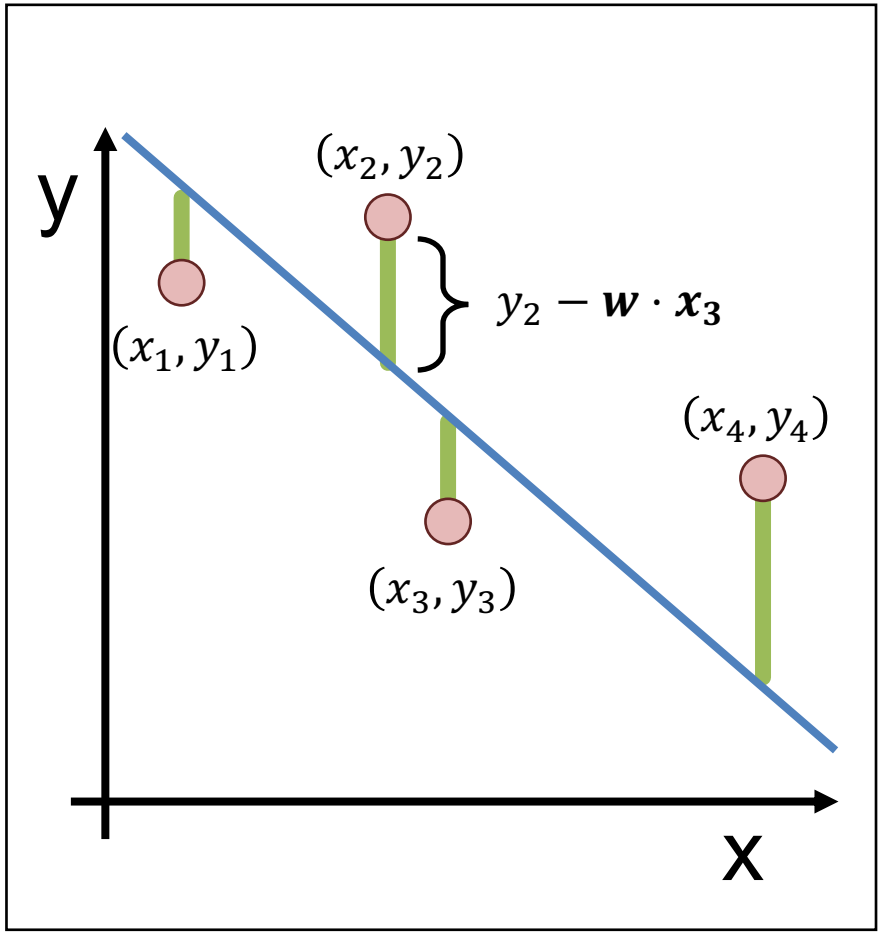
Model: $y = mx + b$

Or: $\boldsymbol{x} = (x, 1); \boldsymbol{w} = (m, b)$

$y = \boldsymbol{w} \cdot \boldsymbol{x}$

Training:

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \sum_{i=1}^{N} (y_i - \boldsymbol{w} \cdot \boldsymbol{x}_i)^2$$



Slide credit: D Fouhey & J Johnson

Solving Least Squares

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \sum_{i=1}^{N} (y_i - \boldsymbol{w} \cdot \boldsymbol{x}_i)^2$$
$$= \arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2$$

Output: Vector of shape (N,)

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$$

Inputs: Matrix of shape (N, 2)

$$\boldsymbol{X} = \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_N & 1 \end{bmatrix}$$

Weights: Vector of shape (2,)

$$\boldsymbol{w} = \begin{bmatrix} m \\ b \end{bmatrix}$$

Solution: $\boldsymbol{w}^* = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$

Slide credit: D Fouhey & J Johnson

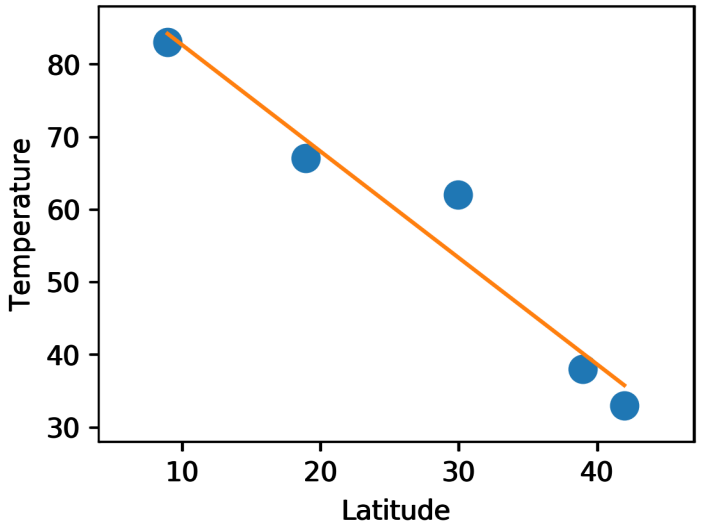
Example: A Bad Weather Model

Given latitude, predict temperature by fitting a line

City	*Latitude (°)*	*Temp (F)*
Ann Arbor	42	33
Washington, DC	39	38
Austin, TX	30	62
Mexico City	19	67
Panama City	9	83

Training

$$\boldsymbol{X}_{5x2} = \begin{bmatrix} 42 & 1 \\ 39 & 1 \\ 30 & 1 \\ 19 & 1 \\ 9 & 1 \end{bmatrix} \quad \boldsymbol{y}_{5x1} = \begin{bmatrix} 33 \\ 38 \\ 62 \\ 67 \\ 83 \end{bmatrix}$$



$$(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

$$w_{2x1} = \begin{bmatrix} -1.47 \\ 97 \end{bmatrix}$$

Temp = -1.47*Lat + 97

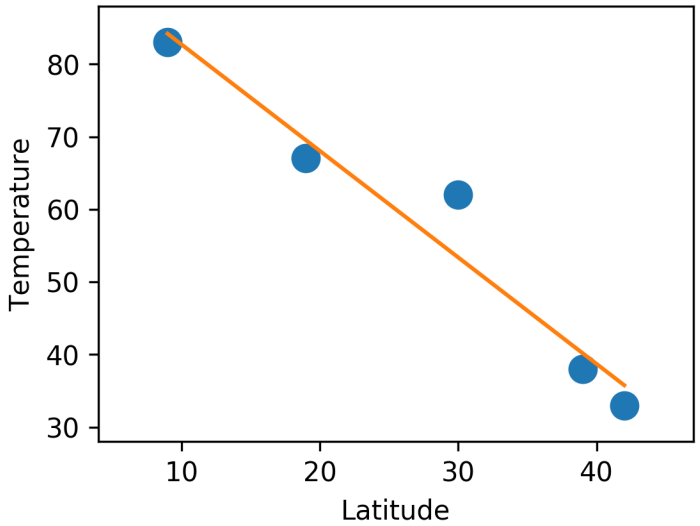
Slide credit: D Fouhey & J Johnson

Example: A Bad Weather Model

Given latitude, predict temperature by fitting a line

City	*Latitude (°)*	*Temp (F)*	*Prediction*	*Error*
Ann Arbor	42	33	35.3	2.3
Washington, DC	39	38	39.7	1.7
Austin, TX	30	62	52.9	10.9
Mexico City	19	67	69.1	2.1
Panama City	9	83	83.8	0.8

Seems good!



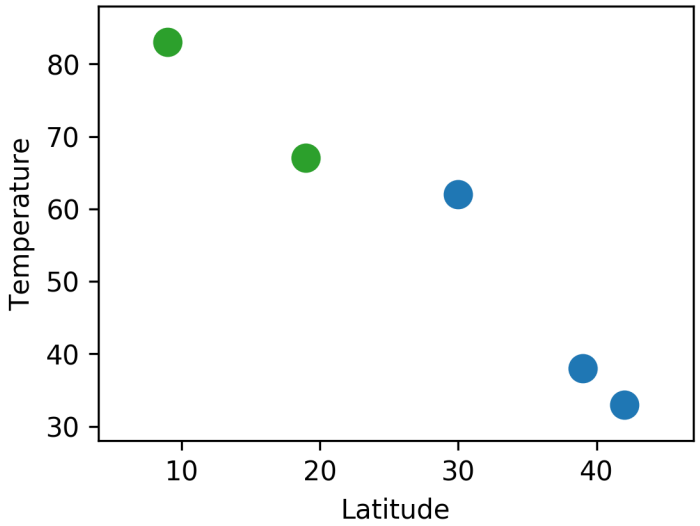
Problem: In ML we don't care about training set performance; we want models that **generalize** to new data

Slide credit: D Fouhey & J Johnson

Example: A Bad Weather Model

Given latitude, predict temperature by fitting a line

	City	*Latitude (°)*	*Temp (F)*
Train	Ann Arbor	42	33
	Washington, DC	39	38
	Austin, TX	30	62
Test	Mexico City	19	67
	Panama City	9	83



Problem: In ML we don't care about training set performance; we want models that **generalize** to new data

Solution: Split dataset into train and test sets

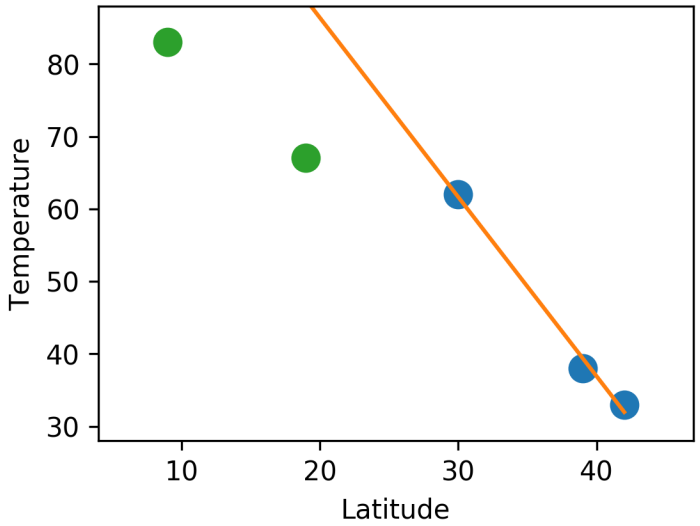
Slide credit: D Fouhey & J Johnson

Example: A Bad Weather Model

Given latitude, predict temperature by fitting a line

	City	Latitude (°)	Temp (F)	Prediction	Error
Train	Ann Arbor	42	33	31.9	1.0
	Washington, DC	39	38	39.4	1.4
	Austin, TX	30	62	61.7	0.3
Test	Mexico City	19	67	88.9	21.9
	Panama City	9	83	113.6	30.6

Problem: Low error on train, high error on test



Problem: In ML we don't care about training set performance; we want models that **generalize** to new data

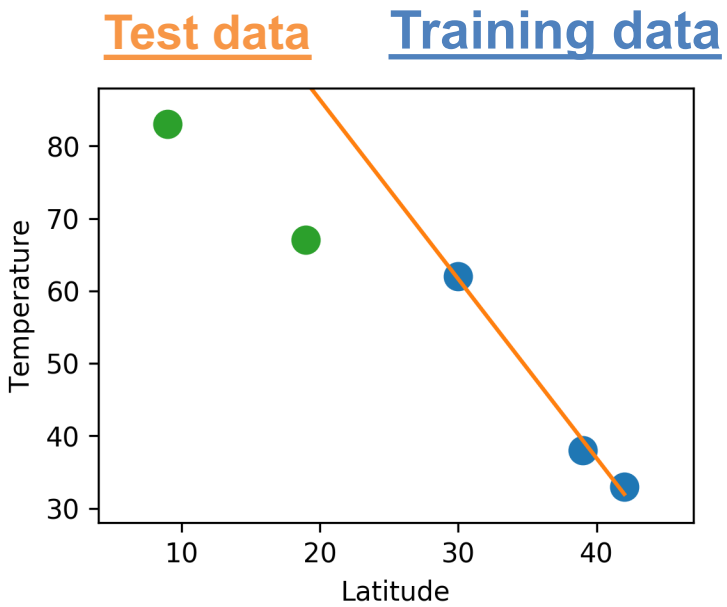
Solution: Split dataset into train and test sets

Fit on train set, evaluate on both

Slide credit: D Fouhey & J Johnson

Overfitting,

Overfitting: Model fits noise in the training set and doesn't generalize well to new data
Model is too flexible

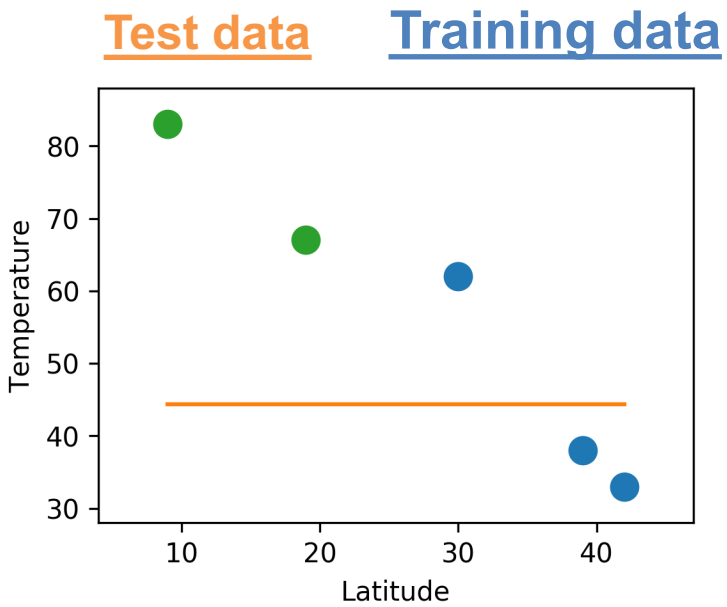


Slide credit: D Fouhey & J Johnson

Overfitting, Underfitting,

Overfitting: Model fits noise in the training set and doesn't generalize well to new data
Model is too flexible

Underfitting: Model doesn't fit the training data well, high error on both train and test
Model is not flexible enough



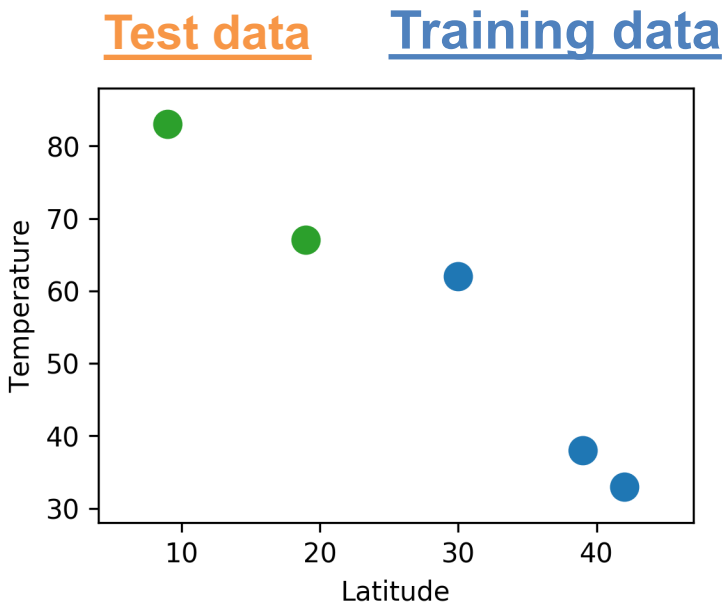
Slide credit: D Fouhey & J Johnson

Overfitting, Underfitting, Regularization

Overfitting: Model fits noise in the training set and doesn't generalize well to new data
Model is too flexible

Underfitting: Model doesn't fit the training data well, high error on both train and test
Model is not flexible enough

Regularization: Penalize model complexity to prevent overfitting and improve generalization



Least Squares

$$\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2$$

L2-Regularized Least Squares

$$\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 + \lambda\|\boldsymbol{w}\|^2$$

$$\boldsymbol{w}^* = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

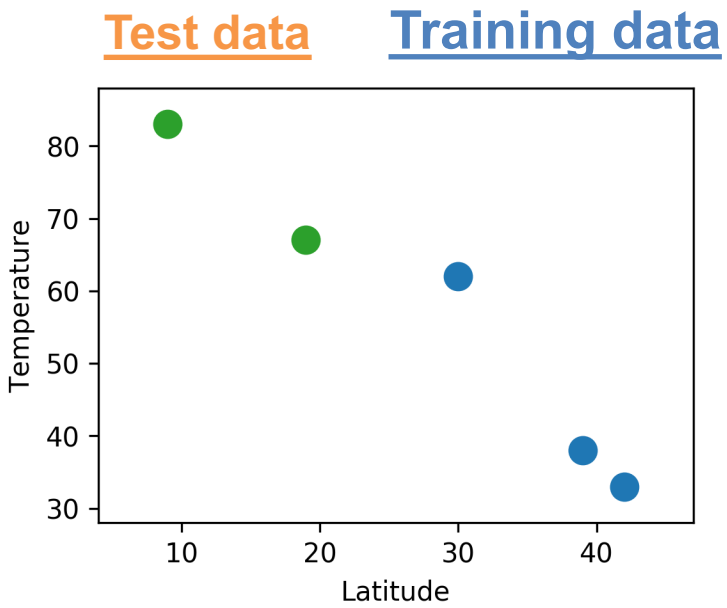
Slide credit: D Fouhey & J Johnson

Overfitting, Underfitting, Regularization

Overfitting: Model fits noise in the training set and doesn't generalize well to new data
Model is too flexible

Underfitting: Model doesn't fit the training data well, high error on both train and test
Model is not flexible enough

Regularization: Penalize model complexity to prevent overfitting and improve generalization



Least Squares

$$\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2$$

L2-Regularized Least Squares

$$\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 + \lambda\|\boldsymbol{w}\|^2$$

Fit training data

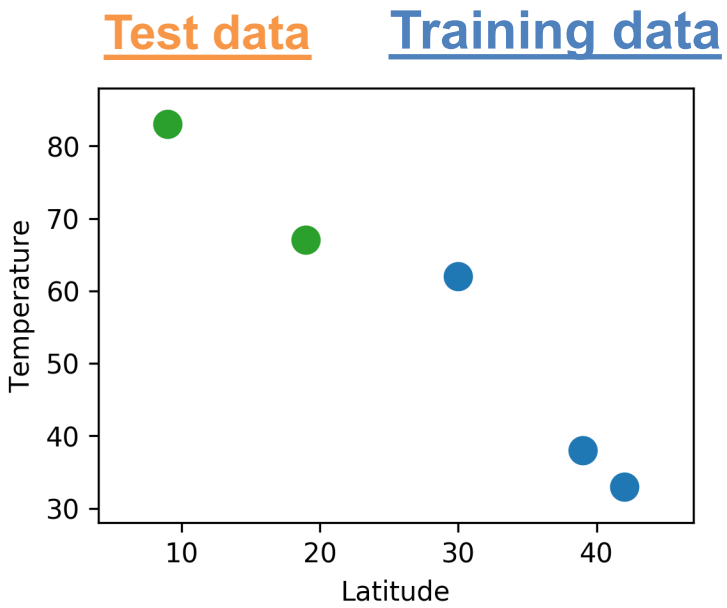
Slide credit: D Fouhey & J Johnson

Overfitting, Underfitting, Regularization

Overfitting: Model fits noise in the training set and doesn't generalize well to new data
Model is too flexible

Underfitting: Model doesn't fit the training data well, high error on both train and test
Model is not flexible enough

Regularization: Penalize model complexity to prevent overfitting and improve generalization



Least Squares

$$\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2$$

L2-Regularized Least Squares

$$\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 + \lambda\|\boldsymbol{w}\|^2$$

Fit training data

Penalize complexity

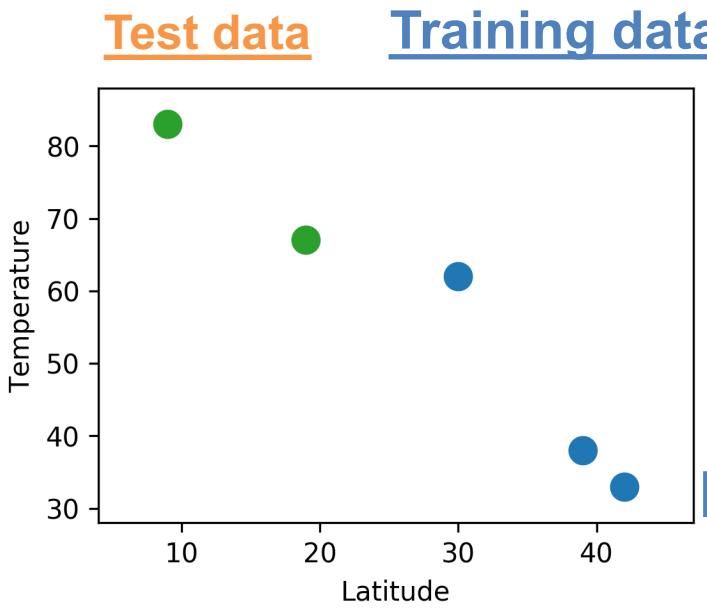
Slide credit: D Fouhey & J Johnson

Overfitting, Underfitting, Regularization

Overfitting: Model fits noise in the training set and doesn't generalize well to new data
Model is too flexible

Underfitting: Model doesn't fit the training data well, high error on both train and test
Model is not flexible enough

Regularization: Penalize model complexity to prevent overfitting and improve generalization



Least Squares

$$\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2$$

L2-Regularized Least Squares

$$\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 + \lambda\|\boldsymbol{w}\|^2$$

Fit training data

Regularization Strength

Penalize complexity

Slide credit: D Fouhey & J Johnson

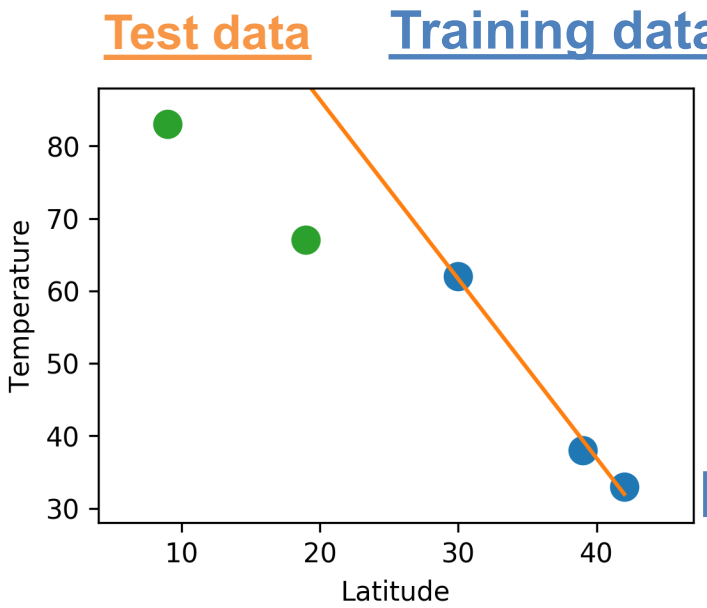
Overfitting, Underfitting, Regularization

Overfitting: Model fits noise in the training set and doesn't generalize well to new data
Model is too flexible

Underfitting: Model doesn't fit the training data well, high error on both train and test
Model is not flexible enough

Regularization: Penalize model complexity to prevent overfitting and improve generalization

$\lambda = 0.00$: No regularization; model overfits



L2-Regularized Least Squares

$$\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 + \lambda\|\boldsymbol{w}\|^2$$

Fit training data

Regularization Strength

Penalize complexity

Slide credit: D Fouhey & J Johnson

Overfitting, Underfitting, Regularization

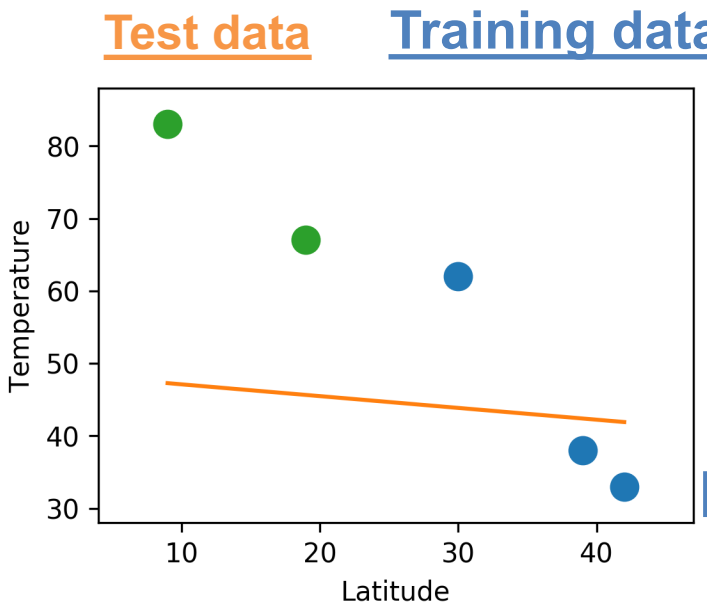
Overfitting: Model fits noise in the training set and doesn't generalize well to new data
Model is too flexible

Underfitting: Model doesn't fit the training data well, high error on both train and test
Model is not flexible enough

Regularization: Penalize model complexity to prevent overfitting and improve generalization

$\lambda = 0.00$: No regularization; model overfits

$\lambda = 0.10$: Too much regularization, underfitting



L2-Regularized Least Squares

$$\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 + \lambda\|\boldsymbol{w}\|^2$$

Fit training data — **Regularization Strength** — **Penalize complexity**

Slide credit: D Fouhey & J Johnson

Overfitting, Underfitting, Regularization

Overfitting: Model fits noise in the training set and doesn't generalize well to new data
Model is too flexible

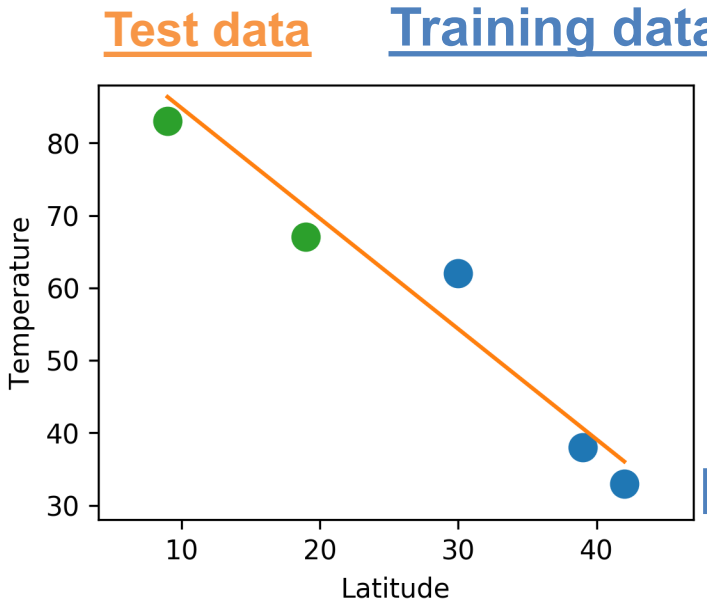
Underfitting: Model doesn't fit the training data well, high error on both train and test
Model is not flexible enough

Regularization: Penalize model complexity to prevent overfitting and improve generalization

$\lambda = 0.00$: No regularization; model overfits

$\lambda = 0.10$: Too much regularization, underfitting

$\lambda = 0.02$: Just right! Good fit and generalization



L2-Regularized Least Squares

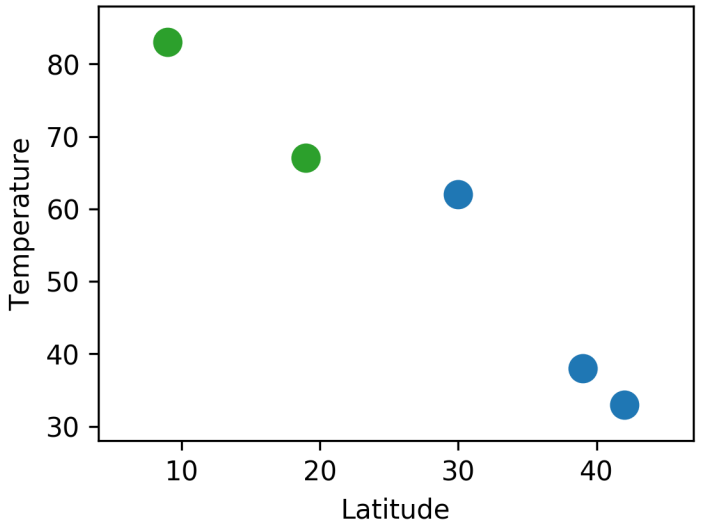
$$\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 + \lambda\|\boldsymbol{w}\|^2$$

Fit training data — **Regularization Strength** — **Penalize complexity**

Slide credit: D Fouhey & J Johnson

Regularization

	City	Latitude (°)	Temp (F)
Train	Ann Arbor	42	33
	Washington, DC	39	38
	Austin, TX	30	62
Test	Mexico City	19	67
	Panama City	9	83



L2-Regularized Least Squares

$$\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 + \lambda\|\boldsymbol{w}\|^2$$

Fit training data

Regularization Strength

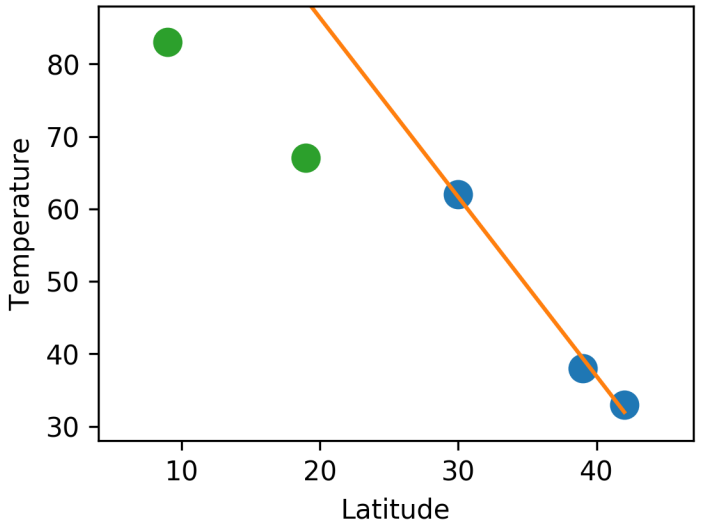
Penalize complexity

Slide credit: D Fouhey & J Johnson

Regularization

No regularization ($\lambda = 0$)

	City	Latitude (°)	Temp (F)	Prediction	Error
Train	Ann Arbor	42	33	31.9	1.0
	Washington, DC	39	38	39.4	1.4
	Austin, TX	30	62	61.7	0.3
Test	Mexico City	19	67	88.9	21.9
	Panama City	9	83	113.6	30.6



L2-Regularized Least Squares

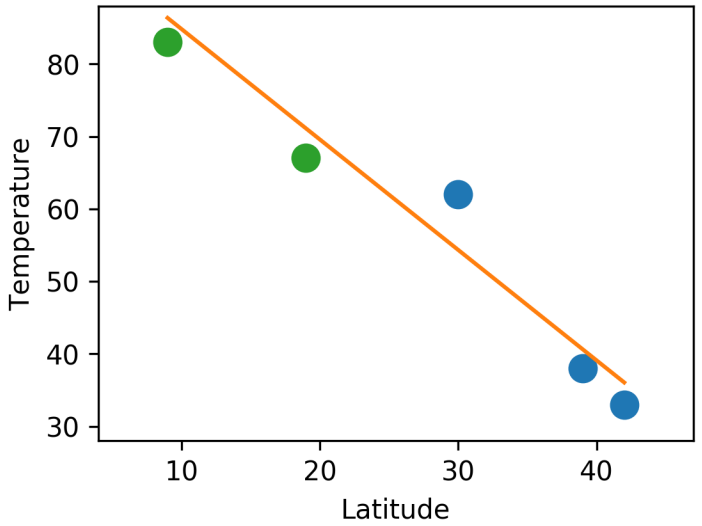
$$\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 + \lambda\|\boldsymbol{w}\|^2$$

Fit training data — **Regularization Strength** — **Penalize complexity**

Slide credit: D Fouhey & J Johnson

Regularization

	City	Latitude (°)	Temp (F)	No regularization ($\lambda = 0$) Prediction	No regularization ($\lambda = 0$) Error	Regularized ($\lambda = 0.02$) Prediction	Regularized ($\lambda = 0.02$) Error
Train	Ann Arbor	42	33	31.9	1.0	36.0	3.0
	Washington, DC	39	38	39.4	1.4	40.6	2.6
	Austin, TX	30	62	61.7	0.3	54.3	7.7
Test	Mexico City	19	67	88.9	21.9	71.1	4.1
	Panama City	9	83	113.6	30.6	86.4	3.4



L2-Regularized Least Squares

$$\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 + \lambda\|\boldsymbol{w}\|^2$$

Fit training data — **Regularization Strength** — **Penalize complexity**

Slide credit: D Fouhey & J Johnson

Parameters and Hyperparameters

L2-Regularized Least Squares

$$\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 + \lambda\|\boldsymbol{w}\|^2$$

Parameter ($\boldsymbol{w}$): (Automatically) selected during training by fitting to training data

Hyperparameter (λ): (Manually) chosen before training, does not depend on training data

Question: How to choose hyperparameters?

Slide credit: D Fouhey & J Johnson

Choosing Hyperparameters

Idea #1: Choose hyperparameters that work best on the data

BAD: λ =0 always works best on training data

Idea #2: Split data into **train** and **test**, choose hyperparameters that work best on test data

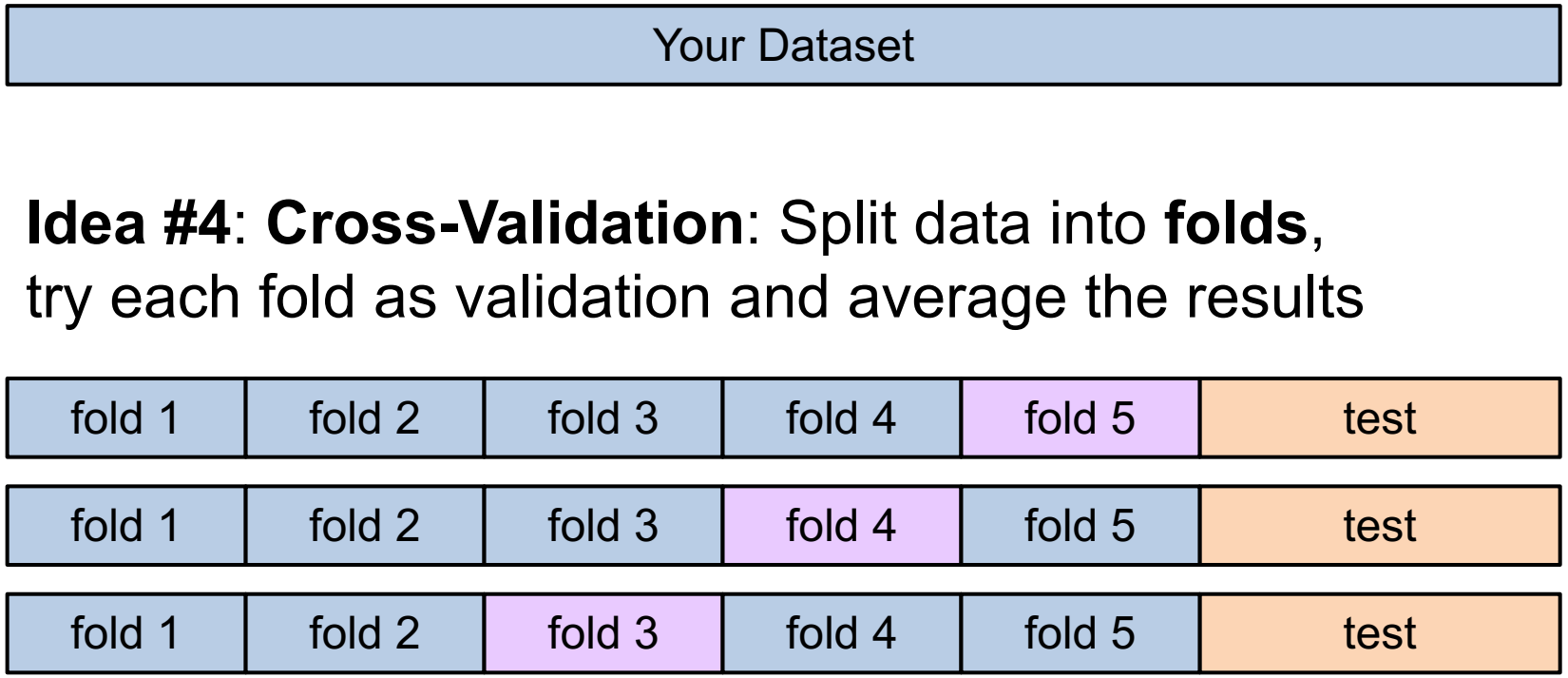
BAD: No idea how we will perform on new data

Idea #3: Split data into **train**, **val**, and **test**; choose hyperparameters on val and evaluate on test

Better!

Slide credit: D Fouhey & J Johnson

Choosing Hyperparameters



Idea #4: **Cross-Validation**: Split data into **folds**, try each fold as validation and average the results

Useful for small datasets, but (unfortunately) not used too frequently in deep learning

Slide credit: D Fouhey & J Johnson

Image Classification: Core Vision Task

Input: image

Output: Assign image to one of a fixed set of categories

cat

bird

deer

dog

truck

Slide credit: D Fouhey & J Johnson

Classification with Least Squares

$\boldsymbol{x}_i \in \mathbb{R}^D$ is image feature

$\boldsymbol{y}_i \in \mathbb{R}^C$ is **one-hot** label

$y_{i,c} = 1$ if $\boldsymbol{x}_i$ has category c, 0 otherwise

Training ($\mathbf{x}_i$,$\mathbf{y}_i$): $\arg\min_{\boldsymbol{W}} \sum_{i=1}^{n} \|\boldsymbol{W}\boldsymbol{x}_i - \boldsymbol{y}_i\|^2$

Inference (x): $\boldsymbol{W}\boldsymbol{x} > t$

Unprincipled in theory, but often effective in practice

The reverse (regression via discrete bins) is also common

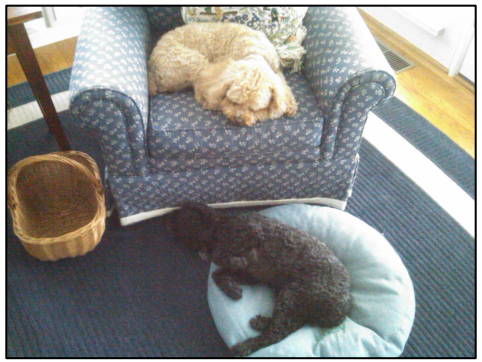
Rifkin, Yeo, Poggio. *Regularized Least Squares Classification* (http://cbcl.mit.edu/publications/ps/rlsc.pdf). 2003

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

Slide credit: D Fouhey & J Johnson

Linear Classifiers

$\boldsymbol{y_i = wx_i + b,}$ $\boldsymbol{x_i \in \mathbb{R}^D}$ is image feature


Model – one weight per class: $\boldsymbol{w_0, w_1, w_2}$

$\boldsymbol{w_0^T x}$ big if cat

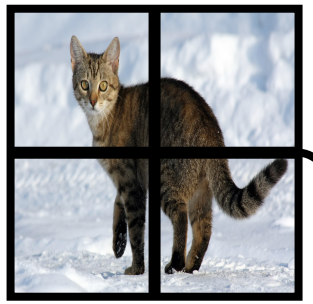
$\boldsymbol{w_1^T x}$ big if dog

$\boldsymbol{w_2^T x}$ big if hippo

Stack together: $\boldsymbol{W_{3xD}}$

Slide credit: D Fouhey & J Johnson

Linear Classifiers



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

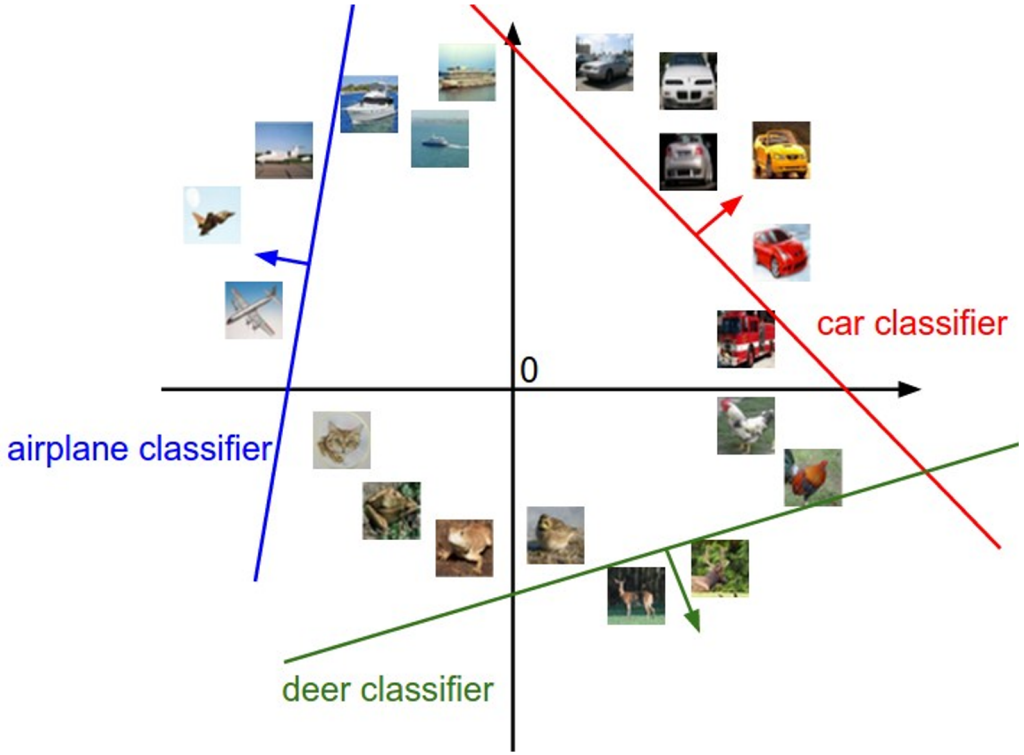
Weight matrix a collection of scoring functions, one per class

Prediction is vector where jth component is “score” for jth class.

Slide credit: D Fouhey & J Johnson Diagram by: A Karpathy, Fei-Fei L

Linear Classifiers: Geometric Intuition

What does a linear classifier look like in 2D?

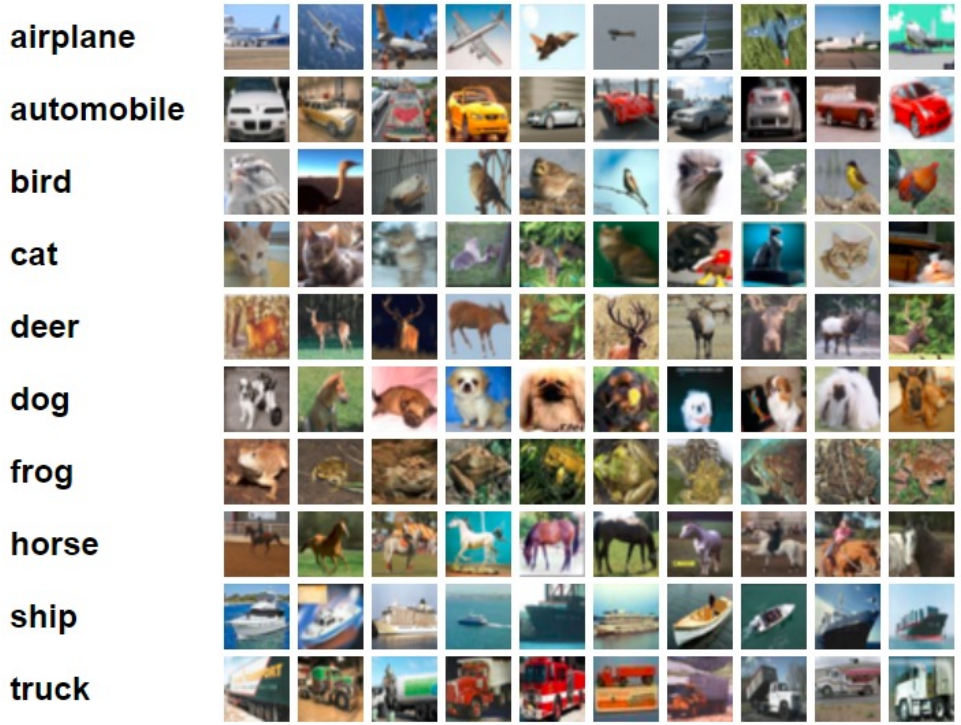


Be aware: Intuition from 2D doesn't always carry over into high-dimensional spaces. See: *On the Surprising Behavior of Distance Metrics in High Dimensional Space.* Charu, Hinneburg, Keim. ICDT 2001

Slide credit: D Fouhey & J Johnson Diagram by: A Karpathy, Fei-Fei L

Linear Classifiers: Visual Intuition

CIFAR 10:
32x32x3 Images, 10 Classes



- Turn each image into feature by unrolling all pixels
- Train a linear model to recognize 10 classes

Slide credit: D Fouhey & J Johnson

Linear Classifiers: Visual Intuition

Decision rule is $\mathbf{w}^T\mathbf{x}$. If $\mathbf{w}_i$ is big, then big values of x_i are indicative of the class.

Deer or Plane?

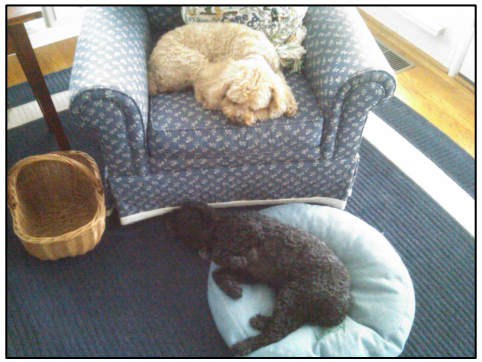
Slide credit: D Fouhey & J Johnson

Linear Classifiers: Visual Intuition

Decision rule is $\mathbf{w}^T\mathbf{x}$. If $\mathbf{w}_i$ is big, then big values of x_i are indicative of the class.

Slide credit: D Fouhey & J Johnson

So Far: Linear Score Function


Model – one weight per class: $\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{w}_2$

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

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

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

Stack together: $\boldsymbol{W}_{\boldsymbol{3xF}}$ where $\mathbf{x}$ is in R^F

How do we know which W is best?

Slide credit: D Fouhey & J Johnson

Choosing W: Loss Function

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

Low loss = good classifier
High loss = bad classifier

(Also called: **objective function**; **cost function**)

Given a dataset

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

of images x_i and labels y_i,

Loss for a single example is:

$$L_i(f(x_i, W), y_i)$$

Loss for the dataset is

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

Slide credit: D Fouhey & J Johnson

Cross-Entropy Loss

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

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

cat	**3.2**
car	5.1
frog	-1.7

Slide credit: D Fouhey & J Johnson

Cross-Entropy Loss

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

Classifier scores

$s = f(x_i, W)$

Softmax function

$$p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)}$$

cat	**3.2**
car	5.1
frog	-1.7

Slide credit: D Fouhey & J Johnson

Cross-Entropy Loss

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

Classifier scores

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

Softmax function

$$p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)}$$

cat	**3.2**
car	5.1
frog	-1.7

Unnormalized log-probabilities / logits

Slide credit: D Fouhey & J Johnson

Cross-Entropy Loss

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

Classifier scores

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

Softmax function

$$p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)}$$

Probabilities must be >= 0

cat	**3.2**	exp →	**24.5**
car	5.1		164
frog	-1.7		0.18

Unnormalized log-probabilities / logits

unnormalized probabilities

Slide credit: D Fouhey & J Johnson

Cross-Entropy Loss

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

Classifier scores

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

Softmax function

$$p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)}$$

	Unnormalized log-probabilities / logits	exp	Probabilities must be >= 0 unnormalized probabilities	normalize	Probabilities must sum to 1 probabilities
cat	**3.2**		**24.5**		**0.13**
car	5.1		164		0.87
frog	-1.7		0.18		0.00

Slide credit: D Fouhey & J Johnson

Cross-Entropy Loss

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

Classifier scores

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

Softmax function

$$p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)}$$

Loss

$$L_i = -\log(p_{y_i})$$

	Unnormalized log-probabilities / logits	exp	unnormalized probabilities (Probabilities must be >= 0)	normalize	probabilities (Probabilities must sum to 1)
cat	**3.2**		**24.5**		**0.13**
car	5.1		164		0.87
frog	-1.7		0.18		0.00

$L_i = -\log(0.13)$

$= \mathbf{2.04}$

Slide credit: D Fouhey & J Johnson

Cross-Entropy Loss

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

Classifier scores

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

Softmax function

$$p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)}$$

Loss

$$L_i = -\log(p_{y_i})$$

	Unnormalized log-probabilities / logits	exp	Probabilities must be >= 0	normalize	Probabilities must sum to 1
cat	**3.2**		**24.5**		**0.13**
car	5.1		164		0.87
frog	-1.7		0.18		0.00
			unnormalized probabilities		probabilities

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

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

Slide credit: D Fouhey & J Johnson

Cross-Entropy Loss

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

Classifier scores

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

Softmax function

$$p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)}$$

Loss

$$L_i = -\log(p_{y_i})$$

	Unnormalized log-probabilities / logits	exp	Probabilities must be >= 0 unnormalized probabilities	normalize	Probabilities must sum to 1 probabilities	Compare	Correct probs
cat	**3.2**		**24.5**		**0.13**		**1.00**
car	5.1		164		0.87		0.00
frog	-1.7		0.18		0.00		0.00

Slide credit: D Fouhey & J Johnson

Cross-Entropy Loss

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

Classifier scores

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

Softmax function

$$p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)}$$

Loss

$$L_i = -\log(p_{y_i})$$

	Unnormalized log-probabilities / logits	exp	Probabilities must be >= 0 unnormalized probabilities	normalize	Probabilities must sum to 1 probabilities	Compare	Correct probs
cat	**3.2**		**24.5**		**0.13**		**1.00**
car	5.1		164		0.87		0.00
frog	-1.7		0.18		0.00		0.00

Kullback-Leibler Divergence

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

Slide credit: D Fouhey & J Johnson

Cross-Entropy Loss

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

Classifier scores

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

Softmax function

$$p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)}$$

Loss

$$L_i = -\log(p_{y_i})$$

	Unnormalized log-probabilities / logits	exp	Probabilities must be >= 0 unnormalized probabilities	normalize	Probabilities must sum to 1 probabilities	Compare	Correct probs
cat	**3.2**		**24.5**		**0.13**		**1.00**
car	5.1		164		0.87		0.00
frog	-1.7		0.18		0.00		0.00

Cross-Entropy:

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

Slide credit: D Fouhey & J Johnson

Cross-Entropy Loss

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

Classifier scores

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

Softmax function

$$p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)}$$

Loss

$$L_i = -\log(p_{y_i})$$

cat	**3.2**
car	5.1
frog	-1.7

Putting it all together:

$$L_i = -\log \frac{\exp(s_{y_i})}{\sum_j \exp(s_j)}$$

Slide credit: D Fouhey & J Johnson

Cross-Entropy Loss

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

Classifier scores

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

Softmax function

$$p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)}$$

Loss

$$L_i = -\log(p_{y_i})$$

cat	**3.2**
car	5.1
frog	-1.7

Putting it all together:

$$L_i = -\log \frac{\exp(s_{y_i})}{\sum_j \exp(s_j)}$$

Q: What is the min / max possible loss L_i?

Slide credit: D Fouhey & J Johnson

Cross-Entropy Loss

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

Classifier scores

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

Softmax function

$$p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)}$$

Loss

$$L_i = -\log(p_{y_i})$$

cat	**3.2**
car	5.1
frog	-1.7

Putting it all together:

$$L_i = -\log \frac{\exp(s_{y_i})}{\sum_j \exp(s_j)}$$

Q: If all scores are small random values, what is the loss?

Slide credit: D Fouhey & J Johnson

Next Class

- A bit more about loss function for classification
  - SVM (hinge) loss
- Optimization
  - How to choose the optimal W (W^*)