(*Mainly*)

Linear Models

EECS 442 – David Fouhey

Fall 2019, University of Michigan

http://web.eecs.umich.edu/~fouhey/teaching/EECS442_F19/

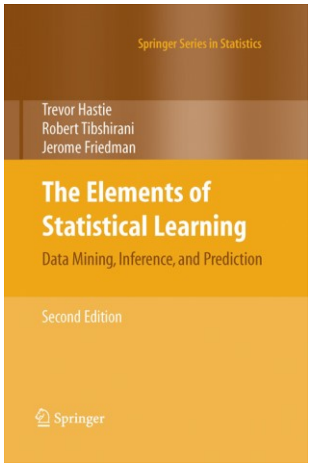
Next Few Classes

- Machine Learning (ML) Crash Course
- I can’t cover everything
- If you can, take a ML course or *learn online*
- ML really won’t solve all problems and is incredibly dangerous if misused
- But ML is a powerful tool and not going away

Terminology

- ML is incredibly messy terminology-wise.
- Most things have at lots of names.
- I will try to write down multiple of them so if you see it later you'll know what it is.

Pointers



Useful book (Free too!):
The Elements of Statistical Learning
Hastie, Tibshirani, Friedman
https://web.stanford.edu/~hastie/ElemStatLearn/

Useful set of data:
UCI ML Repository
https://archive.ics.uci.edu/ml/datasets.html

A lot of important and hard lessons summarized:
https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf

Machine Learning (ML)

- Goal: make "sense" of data
- Overly simplified version: transform vector **x** into vector **y**=T**(x)** that's somehow better
- Potentially you fit T using pairs of datapoints and desired outputs ($\mathbf{x}_i$,$\mathbf{y}_i$), or just using a set of datapoints ($\mathbf{x}_i$)
- Always are trying to find some transformation that minimizes or maximizes some **objective function** or goal.

Machine Learning

Input: **x**

Feature vector/Data point:
Vector representation of datapoint. Each dimension or "**feature**" represents some aspect of the data.

Output: **y**

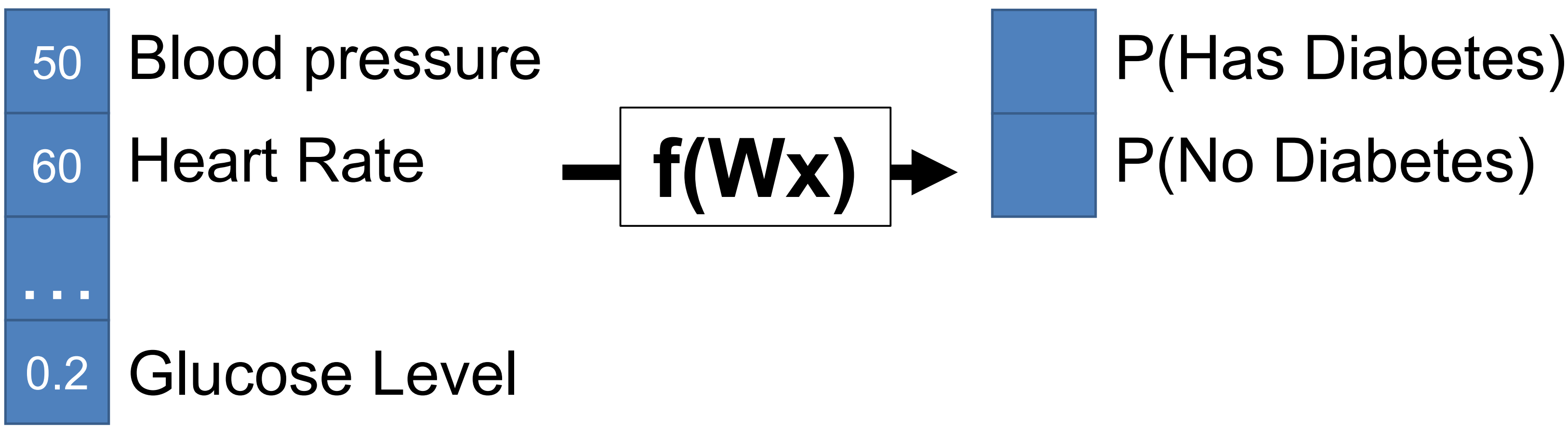
Label / target:
Fixed length vector of desired output. Each dimension represents some aspect of the output data

Supervised: we are given y.
Unsupervised: we are not, and make our own ys.

Example – Health

Input: **x** in R^N

Output: **y**

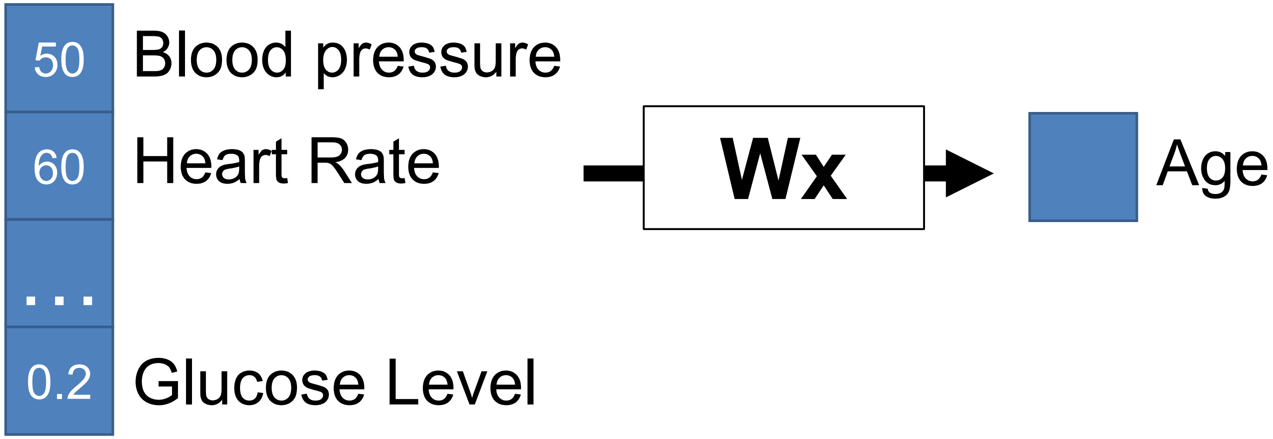


Intuitive objective function: Want correct category to be likely with our model.

Example – Health

Input: **x** in R^N

Output: **y**

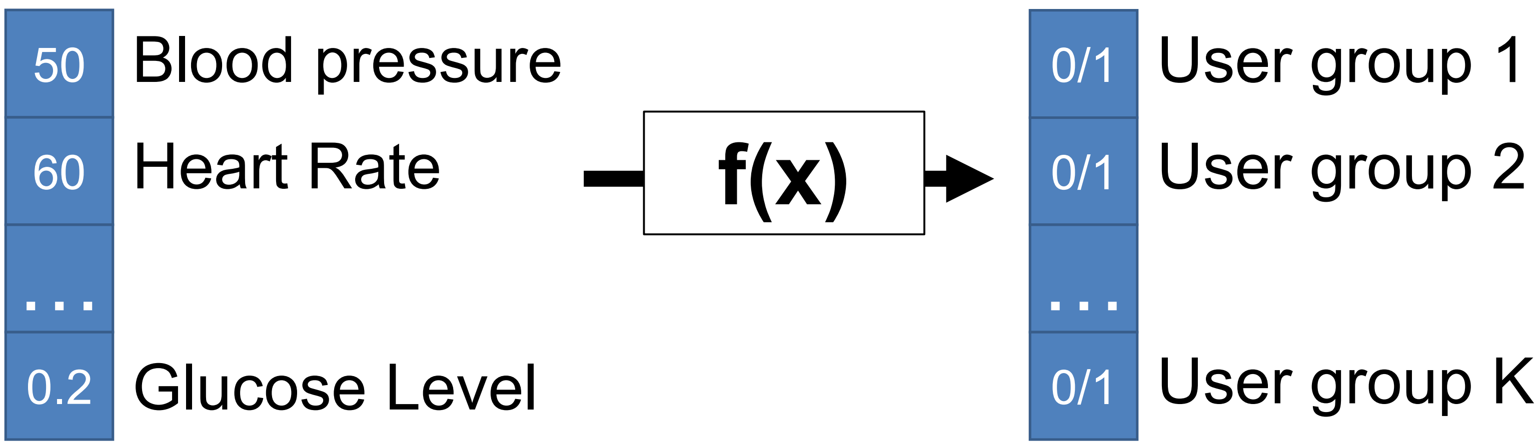


Intuitive objective function: Want our prediction of age to be "close" to true age.

Example – Health

Input: **x** in R^N

Output: **discrete y (unsupervised)**

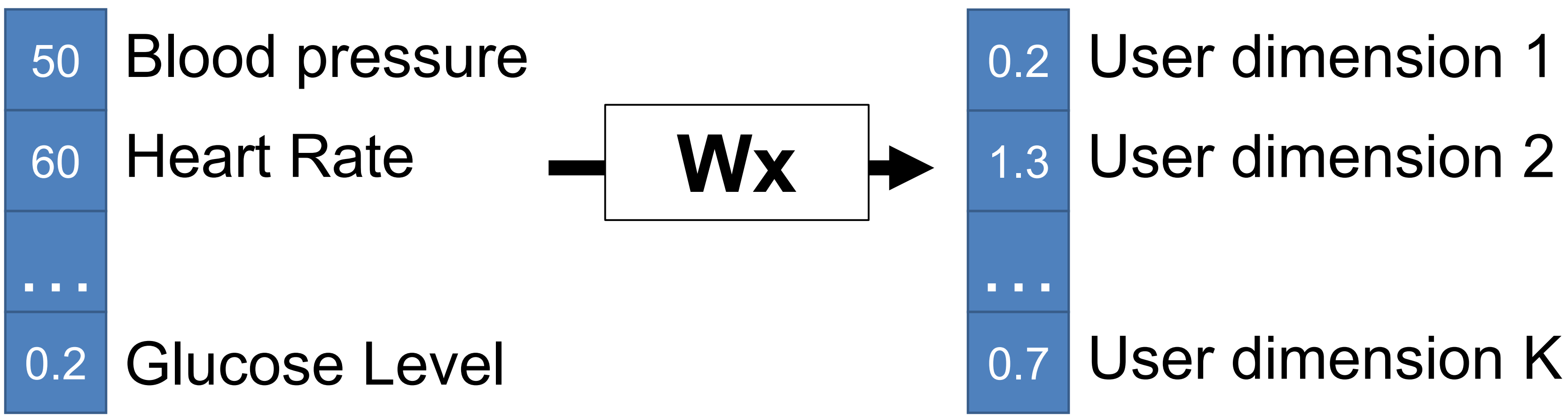


Intuitive objective function: Want to find K groups that explain the data we see.

Example – Health

Input: **x** in R^N

Output: **continuous y (discovered)**

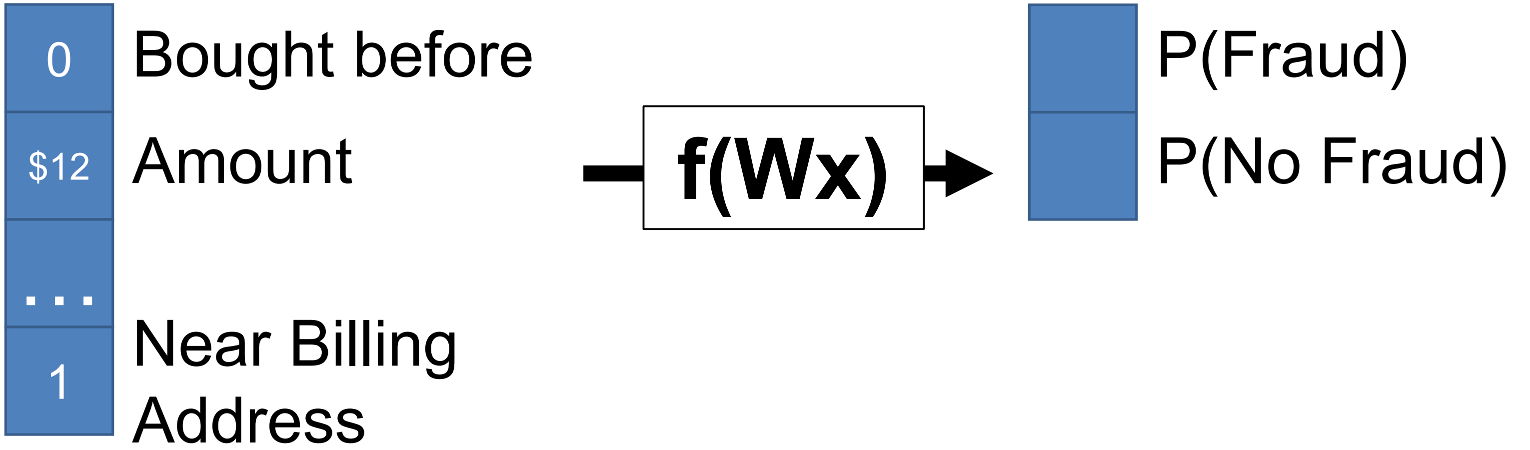


Intuitive objective function: Want to K dimensions (often two) that are easier to understand but capture the variance of the data.

Example – Credit Card Fraud

Input: **x** in R^N

Output: **y**

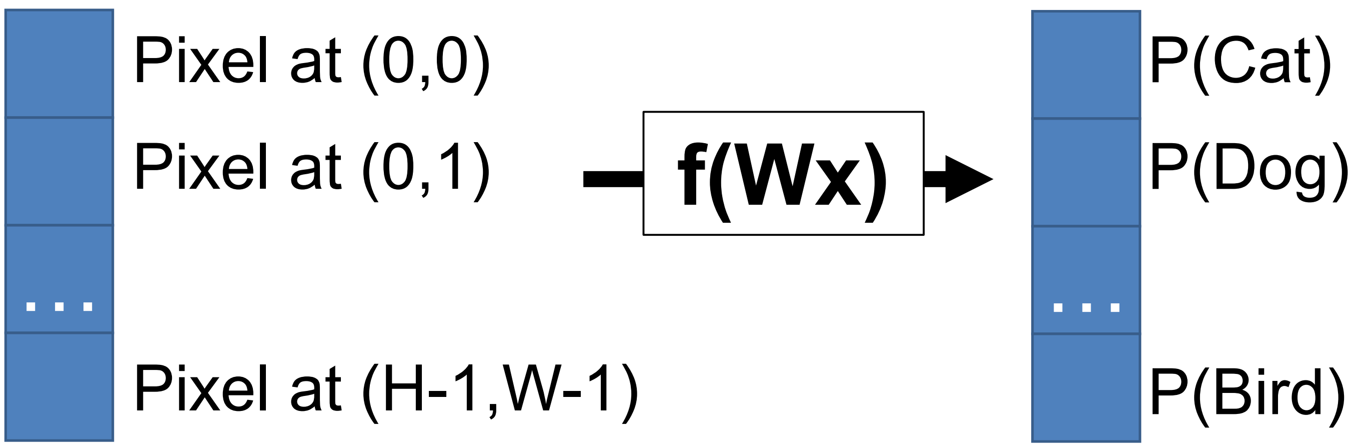


Intuitive objective function: Want correct category to be likely with our model.

Example – Computer Vision

Input: **x** in R^N

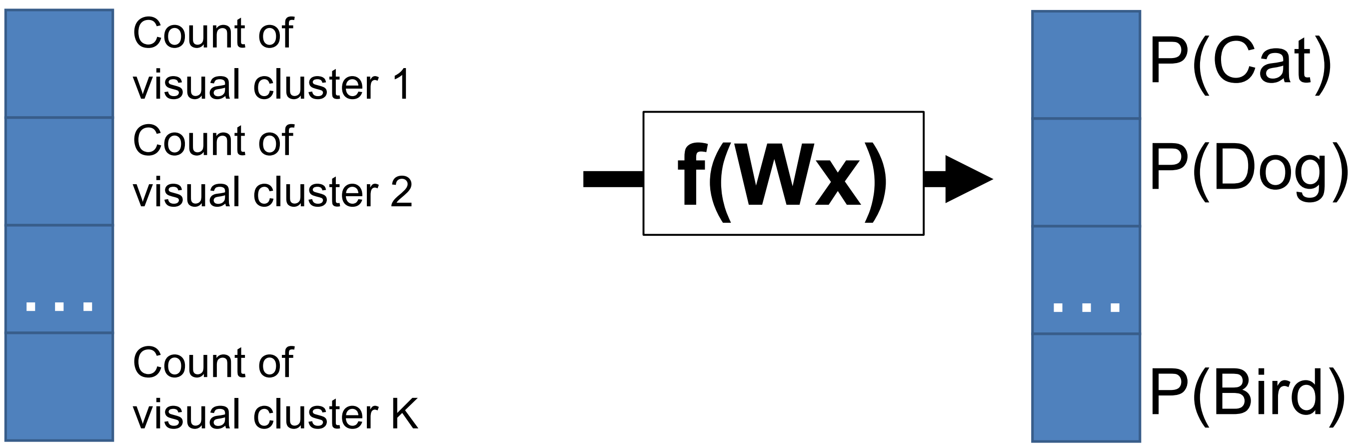
Output: **y**



Intuitive objective function: Want correct category to be likely with our model.

Example – Computer Vision

Input: **x** in R^N Output: **y**

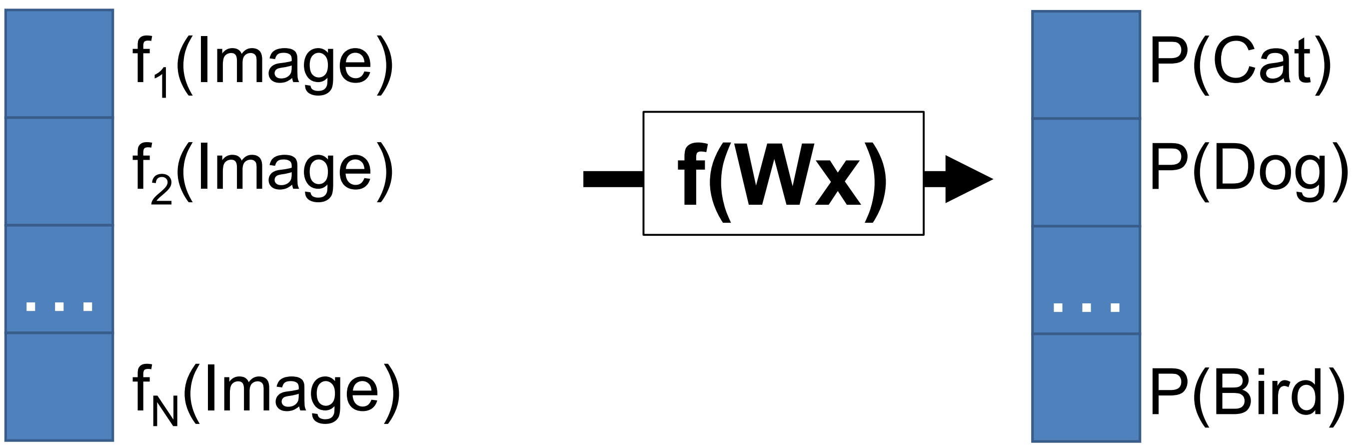


Intuitive objective function: Want correct category to be likely with our model.

Example – Computer Vision

Input: **x** in R^N

Output: **y**



Intuitive objective function: Want correct category to be likely with our model.

Abstractions

- Throughout, assume we've converted data into a fixed-length feature vector. There are well-designed ways for doing this.
- But remember it could be big!
    - Image (e.g., 224x224x3): 151K dimensions
    - Patch (e.g., 32x32x3) in image: 3072 dimensions

ML Problems in Vision

Image credit: Wikipedia

ML Problem Examples in Vision

	Supervised (Data+Labels)	Unsupervised (Just Data)
Discrete Output	Classification/ Categorization	
Continuous Output		

Slide adapted from J. Hays

ML Problem Examples in Vision

***Cat*egorization/Classification**

Binning into K mutually-exclusive categories

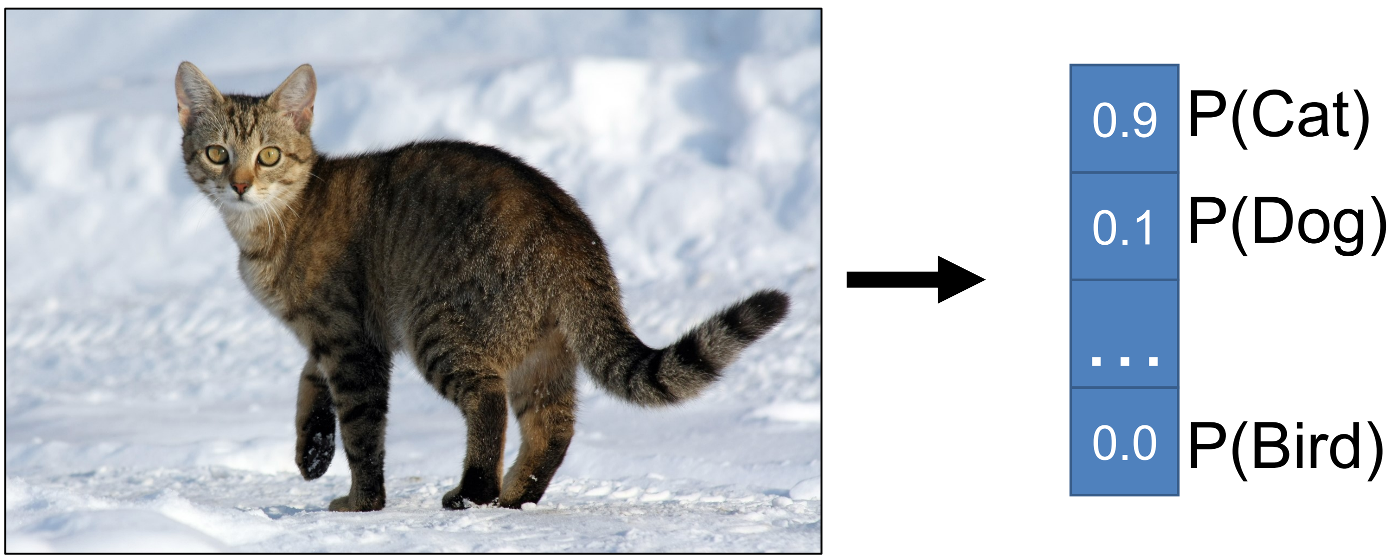


Image credit: Wikipedia

ML Problem Examples in Vision

	Supervised (Data+Labels)	**Unsupervised (Just Data)**
Discrete Output	Classification/ Categorization	
Continuous Output	**Regression**	

Slide adapted from J. Hays

ML Problem Examples in Vision

Regression

Estimating continuous variable(s)

→ 3.6 kg Cat weight

Image credit: Wikipedia

ML Problem Examples in Vision

	Supervised (Data+Labels)	Unsupervised (Just Data)
Discrete Output	Classification/ Categorization	**Clustering**
Continuous Output	Regression	

Slide adapted from J. Hays

ML Problem Examples in Vision

Clustering

Given a set of cats, automatically discover clusters or *cat*egories.

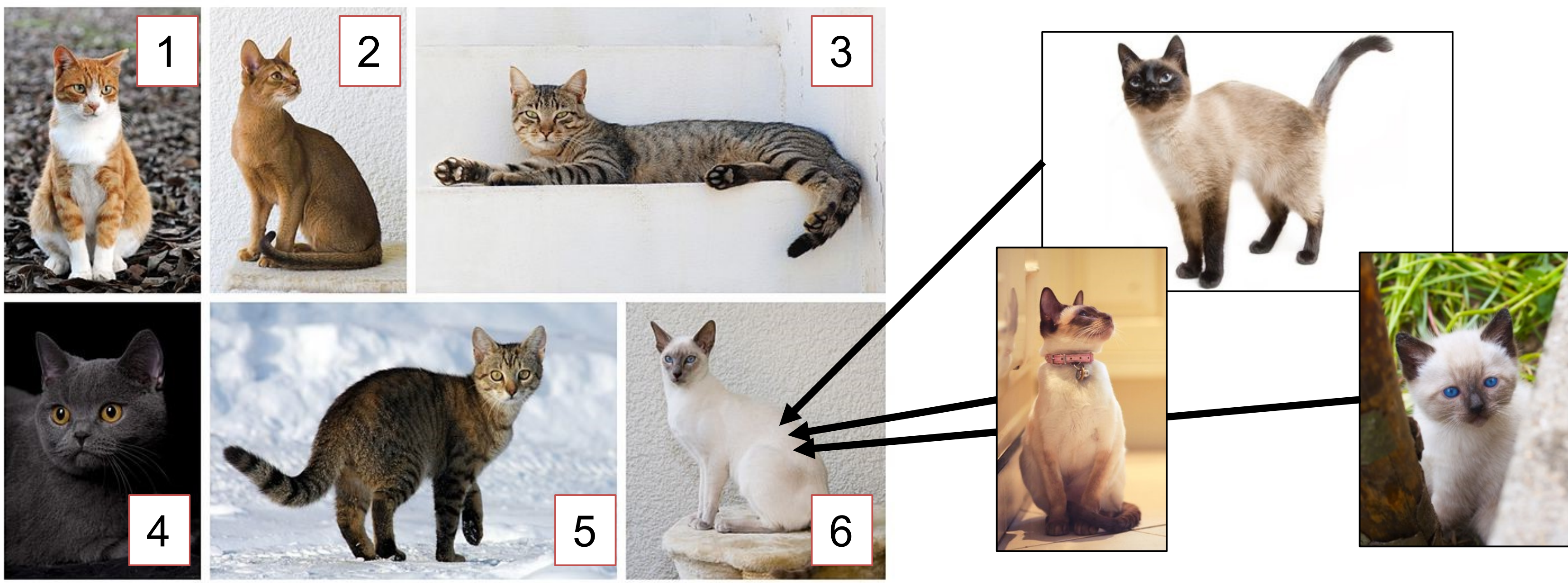


Image credit: Wikipedia, cattime.com

ML Problem Examples in Vision

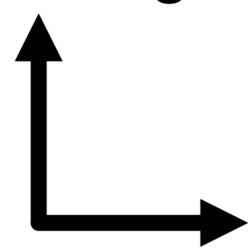
	Supervised (Data+Labels)	**Unsupervised (Just Data)**
Discrete Output	Classification/ Categorization	Clustering
Continuous Output	Regression	**Dimensionality Reduction**

Slide adapted from J. Hays

ML Problem Examples in Vision

Dimensionality Reduction

Find dimensions that best explain the whole image/input



For ordinary images, this is currently a totally hopeless task. For certain images (e.g., faces, this works reasonably well)

Image credit: Wikipedia

Practical Example

- ML has a tendency to be mysterious
- Let's start with:
  - A model you learned in middle/high school (a line)
  - Least-squares
- One thing to remember:
  - N eqns, <N vars = overdetermined (will have errors)
  - N eqns, N vars = exact solution
  - N eqns, >N vars = underdetermined (infinite solns)

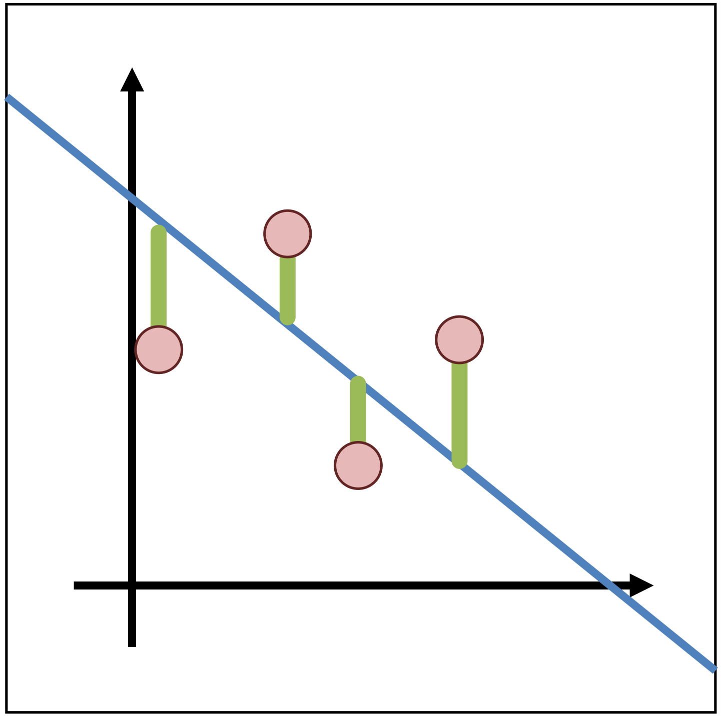
Example – Least Squares

Let's make the world's **worst** weather model

Data: (x_1,y_1), (x_2,y_2), …, (x_k,y_k)

Model: (m,b) $y_i=mx_i+b$
Or $(\mathbf{w})$ $y_i = \mathbf{w}^T\mathbf{x}_i$

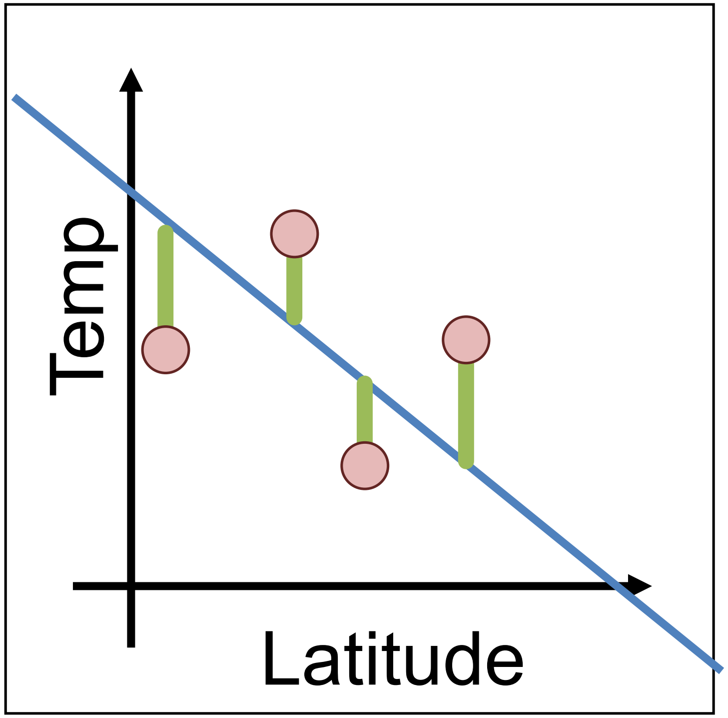
Objective function:
$(y_i - \mathbf{w}^T\mathbf{x}_i)^2$



World's Worst Weather Model

Given latitude (distance above equator), 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



Example – Least Squares

$$\sum_{i=1}^{k} (y_i - \boldsymbol{w}^T \boldsymbol{x_i})^2 \quad \longrightarrow \quad \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2$$

Output:
Temperature

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

Inputs:
Latitude, 1

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

Model/Weights:
Latitude, “Bias”

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

Example – Least Squares

$$\sum_{i=1}^{k}(y_i - \boldsymbol{w}^T\boldsymbol{x_i})^2 \quad \longrightarrow \quad \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2$$

Output:
Temperature

$$\boldsymbol{y} = \begin{bmatrix} 33 \\ \vdots \\ 83 \end{bmatrix}$$

Inputs:
Latitude, 1

$$\boldsymbol{X} = \begin{bmatrix} 42 & 1 \\ \vdots & \vdots \\ 9 & 1 \end{bmatrix}$$

Model/Weights:
Latitude, "Bias"

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

Intuitively why do we add a one to the inputs?

Example – Least Squares

Training ($\mathbf{x}_i$,y_i):

$$\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 \quad \textbf{or}$$

$$\arg\min_{\boldsymbol{w}} \sum_{i=1}^{n} \|\boldsymbol{w}^T \boldsymbol{x_i} - y_i\|^2$$

Loss function/objective: evaluates correctness.
Here: Squared L2 norm / Sum of Squared Errors

Training/Learning/Fitting: try to find model that *optimizes*/*minimizes* an objective / loss function

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

Example – Least Squares

Training ($\mathbf{x}_i$,y_i):

$$\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 \quad \textbf{or}$$

$$\arg\min_{\boldsymbol{w}} \sum_{i=1}^{n} \|\boldsymbol{w}^T \boldsymbol{x_i} - y_i\|^2$$

Inference (x):

$$\boldsymbol{w}^T \boldsymbol{x} = w_1 x_1 + \cdots + w_F x_F$$

Testing/Inference: Given a new output, what's the prediction?

Least Squares: Learning

Data

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

Model

Temp =
-1.47*Lat + 97

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

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

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

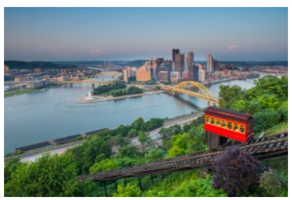
Let's Predict The Weather

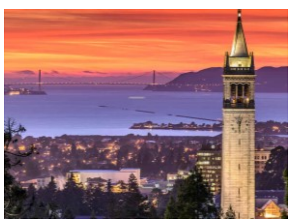
The EECS 442 Weather Channel

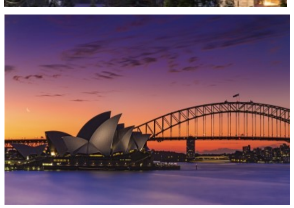
City	*Latitude*	*Temp*	*Temp*	*Error*
Ann Arbor	42	33	35.3	2.3
Washington, DC	39	38	39.7	1.7
Austin, TX	30	62	52.9	10.9
Mexico City	19	67	69.1	2.1
Panama City	9	83	83.8	0.8

Is This a Minimum Viable Product?

The EECS 442 Weather Channel	The Weather Channel
Pittsburgh: Temp = -1.47*40 + 97 = 38	*Actual Pittsburgh:* *45*
Berkeley: Temp = -1.47*38 + 97 = 41	*Actual Berkeley:* *53*
Sydney: Temp = -1.47*-33 + 97 = **146**	*Actual Sydney:* *74*







Won’t do so well in the Australian market…

Where Can This Go Wrong?

Where Can This Go Wrong?

Data

City	*Latitude*	*Temp*
Ann Arbor	42	33
Washington, DC	39	38

Model

Temp =
-1.66*Lat + 103

How well can we predict Ann Arbor and DC and why?

Always Need Separated Testing

Model might be fit data too precisely "*overfitting*"
Remember: #datapoints = #params = perfect fit

Model may only work under some conditions (e.g., trained on northern hemisphere).

Sydney:
Temp = -1.47*-33 + 97 = **146**

Training and Testing

Fit model parameters on **training** set;
evaluate on *entirely unseen* **test** set.

"It's tough to make predictions, especially about the future"
-Yogi Berra

Nearly any model can predict data it's seen. If your model can't accurately interpret "unseen" data, it's probably useless. We have no clue whether it has just memorized.

Let's Improve Things

If one feature does ok, what about more features!?

City Name	*Latitude (deg)*	*Avg July High (F)*	*Avg Snowfall*	*Temp (F)*
Ann Arbor	42	83	58	33
Washington, DC	39	88	15	38
Austin, TX	30	95	0.6	62
Mexico City	19	74	0	67
Panama City	9	93	0	83

$\boldsymbol{X}_{5x4}$ 4 features + a feature of 1s for intercept/bias

$\boldsymbol{y}_{5x1}$

Let's Improve Things

All the math works out!

Data

$\boldsymbol{X}_{5x4}$ $\boldsymbol{y}_{5x1}$

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

Model

$\boldsymbol{w}_{4x1}$

New EECS 442 Weather Rule:

w_1*latitude + w_2*(avg July high) +
w_3*(avg snowfall) + w_4*1

In general called linear regression

Let's Improve Things More

If one feature does ok, what about **LOTS** of features!?

City Name	*Latitude (deg)*	*Avg July High (F)*	*Avg Snowfall*	*Day of Year*	*Elevation (ft)*	*% Letter M*	*Temp (F)*
Ann Arbor	42	83	58	45	840	100	33
Washington, DC	39	88	15	45	409	3	38
Austin, TX	30	95	0.6	45	489	2	62
Mexico City	19	74	0	45	7200	4	67
Panama City	9	93	0	45	7	1	83

$\boldsymbol{X}_{5x7}$ 6 features + a feature of 1s for intercept/bias $\boldsymbol{y}_{5x1}$

Let's Improve Things More

Data

$\boldsymbol{X}_{5x7}$ $\boldsymbol{y}_{5x1}$

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

Model

$\boldsymbol{w}_{7x1}$

$$\boldsymbol{w}^* = \underbrace{\left(\boldsymbol{X}^T\boldsymbol{X}\right)}^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

$\mathbf{X}^\mathbf{T}\mathbf{X}$ is a 7x7 matrix but is **rank deficient** (rank 5) *and has no inverse. There are an infinite number of solutions.*

Have to express some preference for which of the infinite solutions we want.

Exercise for the mathematically-inclined folks: derive what the space of solutions looks like.

The Fix – Regularized Least Squares

Add **regularization** to objective that prefers some solutions:

Before: $\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2$ → Loss

After: $\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda \|\boldsymbol{w}\|_2^2$

Loss   Trade-off   Regularization

Want model "smaller": pay a penalty for **w** with big norm

Intuitive Objective: accurate model (low loss) but not too complex (low regularization). λ controls how much of each.

The Fix – Regularized Least Squares

Objective: $\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda\|\boldsymbol{w}\|_2^2$

Loss — Trade-off — Regularization

Take $\frac{\partial}{\partial \boldsymbol{w}}$, set to **0**, solve

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

$\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I}$ is full-rank (and thus invertible) for $\lambda>0$

Called *lots of things:* regularized least-squares, Tikhonov regularization (after Andrey Tikhonov), ridge regression, Bayesian linear regression with a multivariate normal prior.

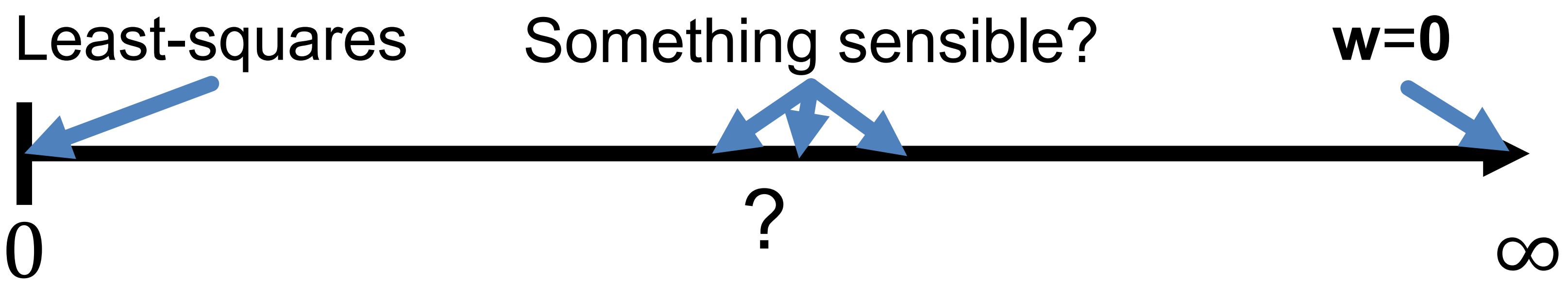
The Fix – Regularized Least Squares

Objective: $\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda \|\boldsymbol{w}\|_2^2$

Loss | Trade-off | Regularization

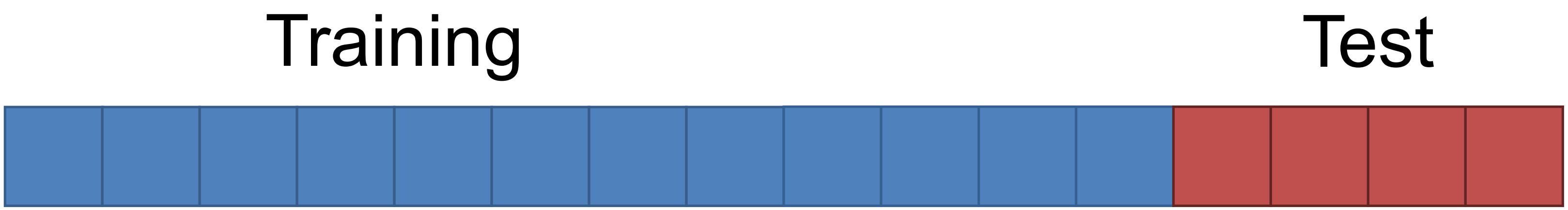
What happens (and why) if:

- $\lambda=0$
- $\lambda=\infty$



Training and Testing

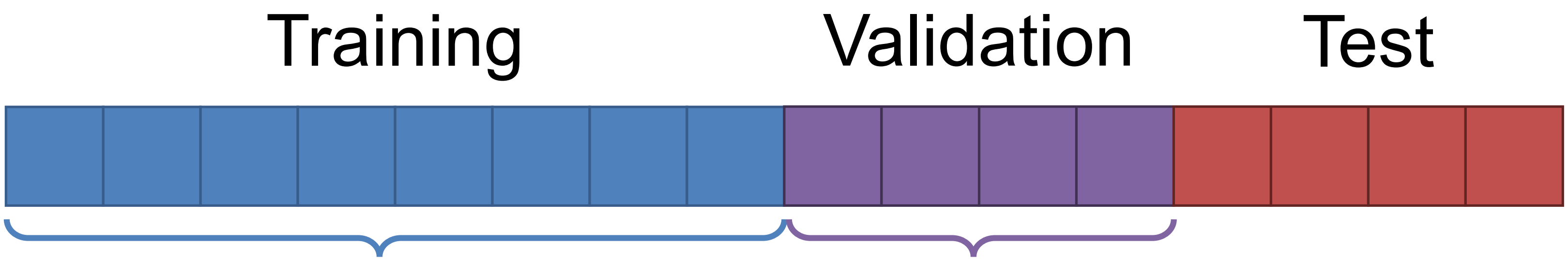
Fit model parameters on training set;
evaluate on *entirely unseen* test set.



How do we pick λ?

Training and Testing

Fit model parameters on training set;
find *hyperparameters* by testing on validation set;
evaluate on *entirely unseen* test set.

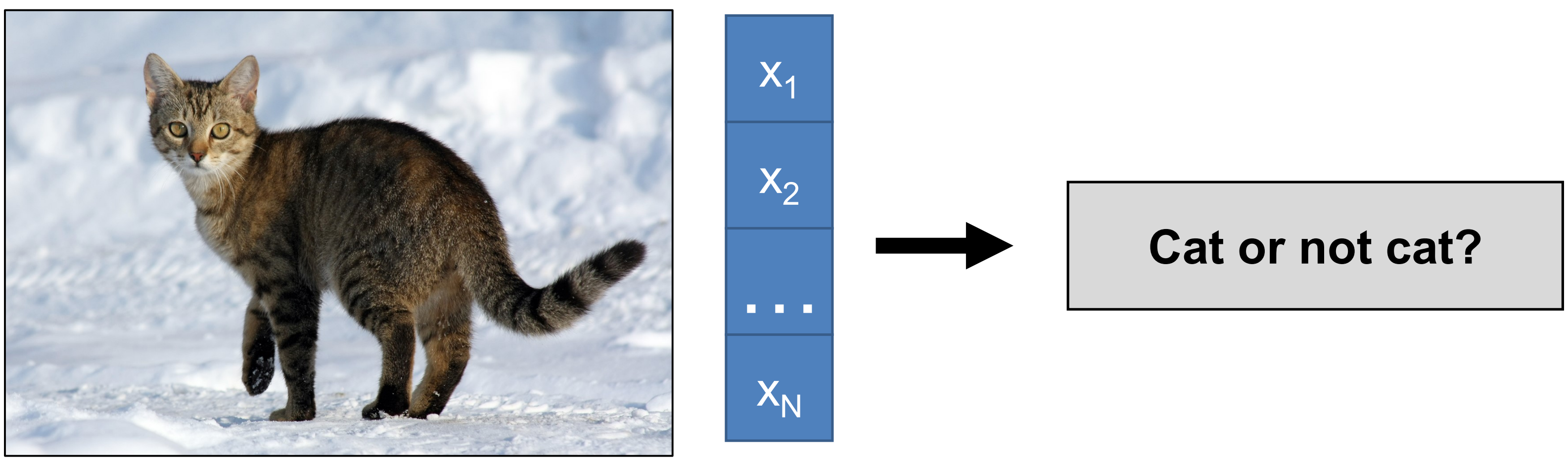


Use these data points to fit $\mathbf{w}^*=(\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$

Evaluate on these points for different λ, pick the best

Classification

Start with simplest example: binary classification



Actually: a feature vector representing the image

Classification by Least-Squares

Treat as regression: x_i is image feature; y_i is 1 if it's a cat, 0 if it's not a cat. Minimize least-squares loss.

Training ($\mathbf{x}_i$,y_i):

$$\arg\min_{\boldsymbol{w}} \sum_{i=1}^{n} \|\boldsymbol{w}^T \boldsymbol{x}_{\boldsymbol{i}} - y_i\|^2$$

Inference (x):

$$\boldsymbol{w}^T \boldsymbol{x} > t$$

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.

Easiest Form of Classification

Just **memorize** (as in a Python dictionary)
Consider cat/dog/hippo classification.

If this:
cat.

If this:
dog.

If this:
hippo.

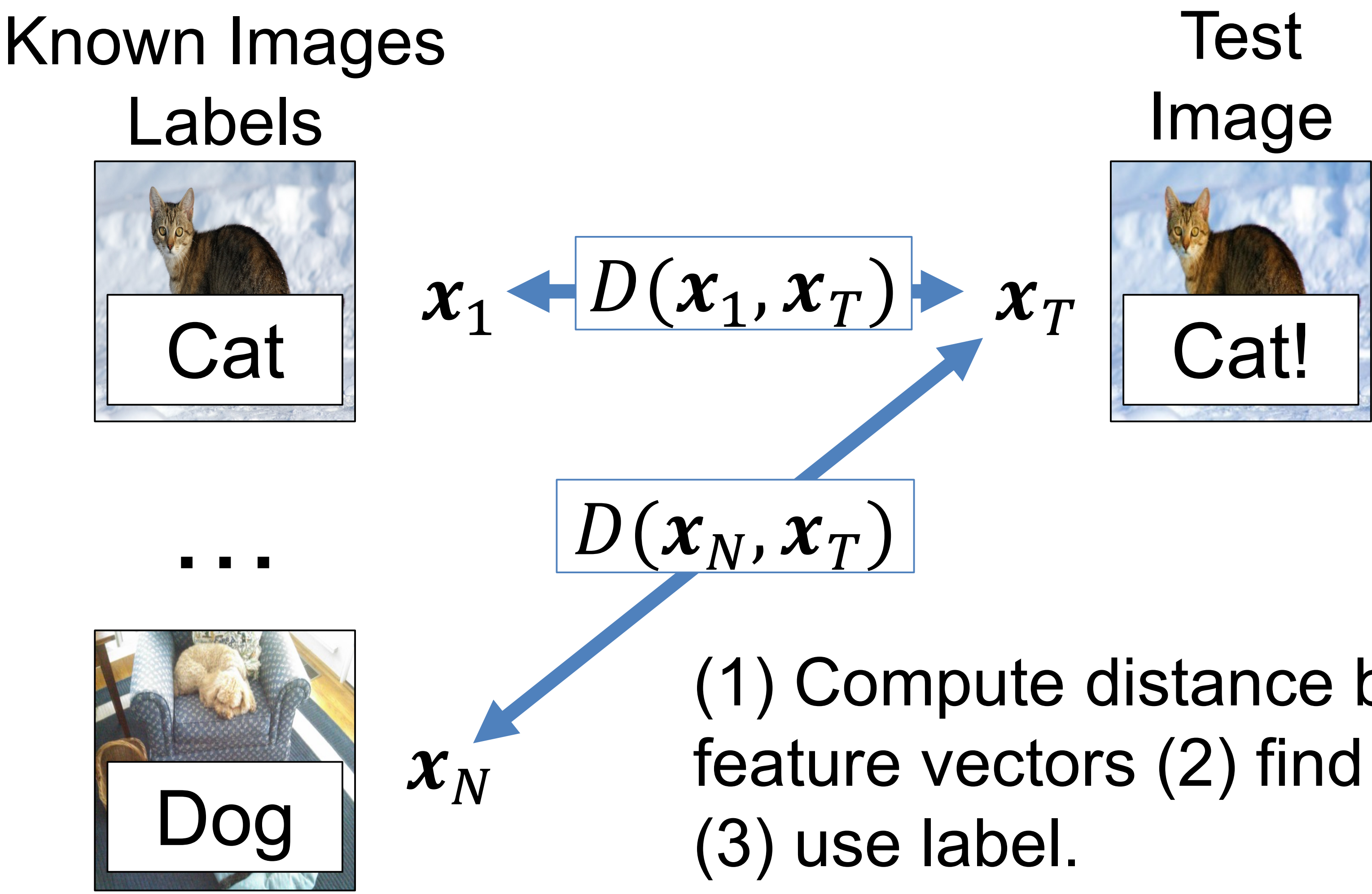
Easiest Form of Classification

Where does this go wrong?

Rule: if this, then cat

Hmmm. Not quite the same.

Easiest Form of Classification



(1) Compute distance between feature vectors (2) find nearest (3) use label.

Nearest Neighbor

"Algorithm"

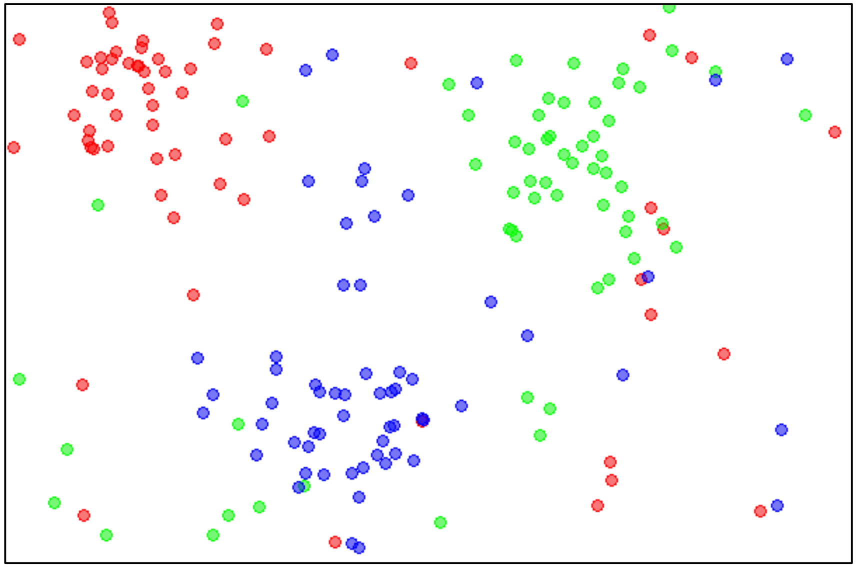
Training ($\mathbf{x}_i$,y_i): Memorize training set

Inference (x):

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

Nearest Neighbor

2D Datapoints
(colors = labels)



2D Predictions
(colors = labels)

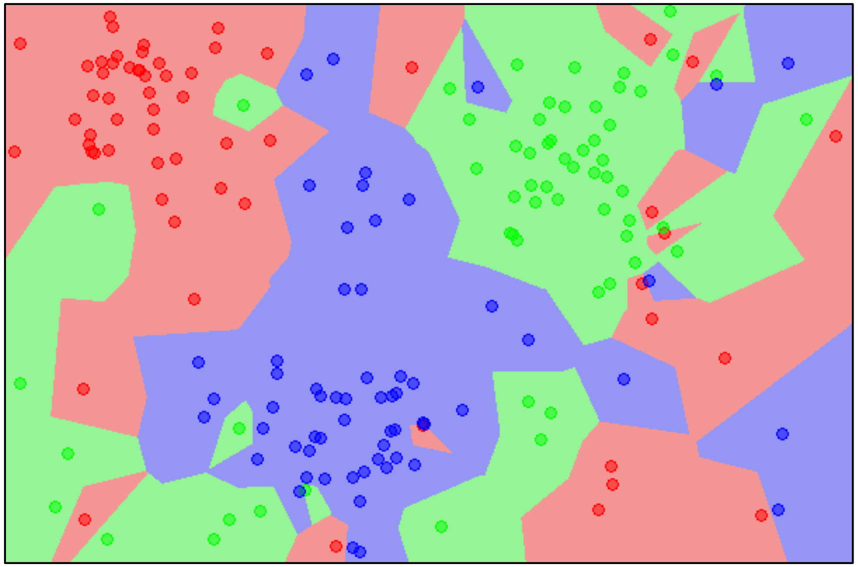
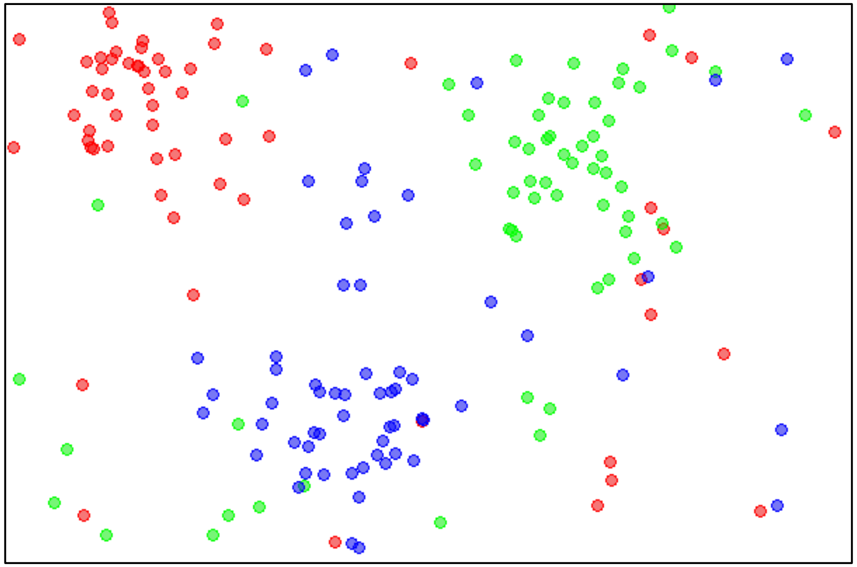


Diagram Credit: Wikipedia

K-Nearest Neighbors

Take top K-closest points, vote

2D Datapoints (colors = labels)



2D Predictions (colors = labels)

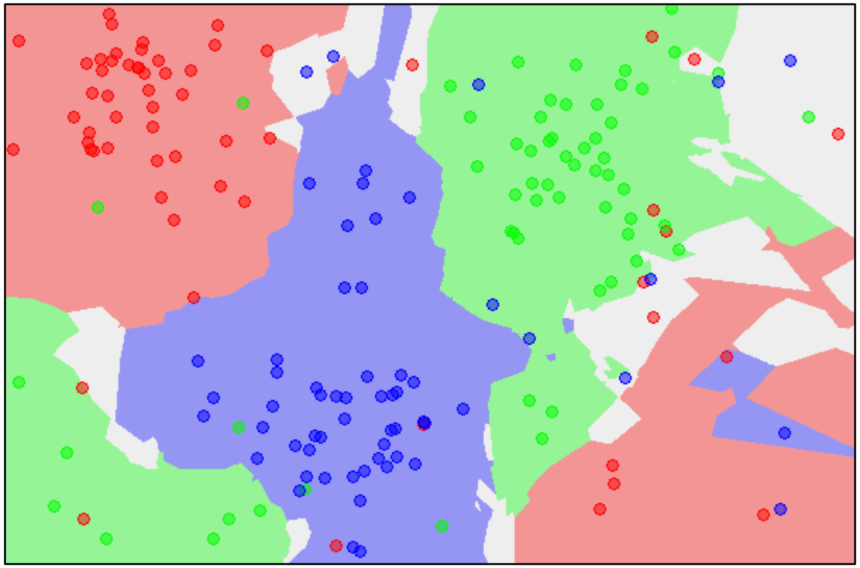
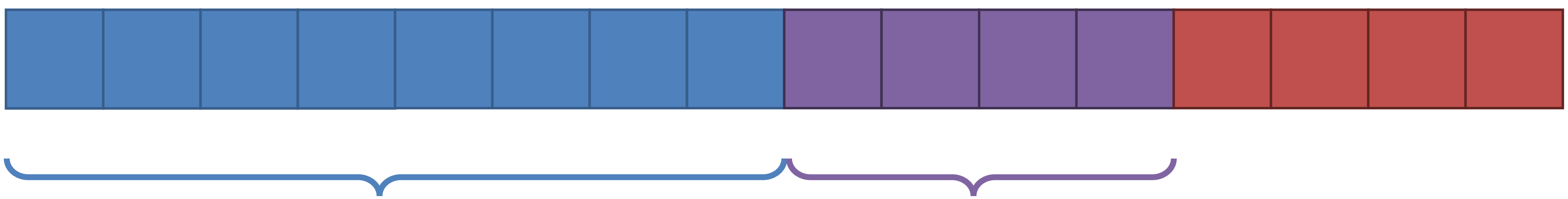


Diagram Credit: Wikipedia

K-Nearest Neighbors

What distance? What value for K?

Training Validation Test



Use these data points for lookup

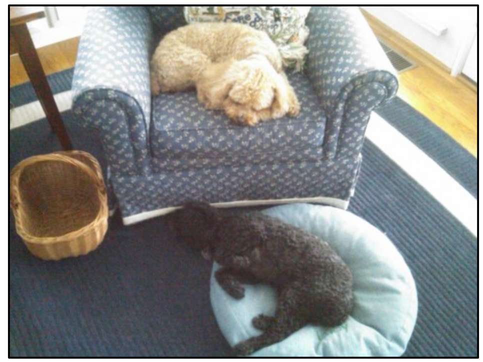
Evaluate on these points for different k, distances

K-Nearest Neighbors

- No learning going on but usually effective
- Same algorithm for every task
- As number of datapoints $\rightarrow \infty$, error rate is guaranteed to be at most 2x worse than optimal you could do on data

Linear Models

Example Setup: 3 classes


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

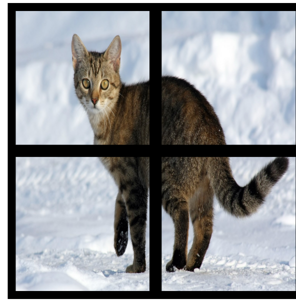
$\boldsymbol{w}_0^T\boldsymbol{x}$ big if cat

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

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

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

Linear Models



Cat weight vector	0.2	-0.5	0.1	2.0	1.1
Dog weight vector	1.5	1.3	2.1	0.0	3.2
Hippo weight vector	0.0	0.3	0.2	-0.3	-1.2

W

56
231
24
2
1

x_i

-96.8	Cat score
437.9	Dog score
61.95	Hippo score

Wx_i

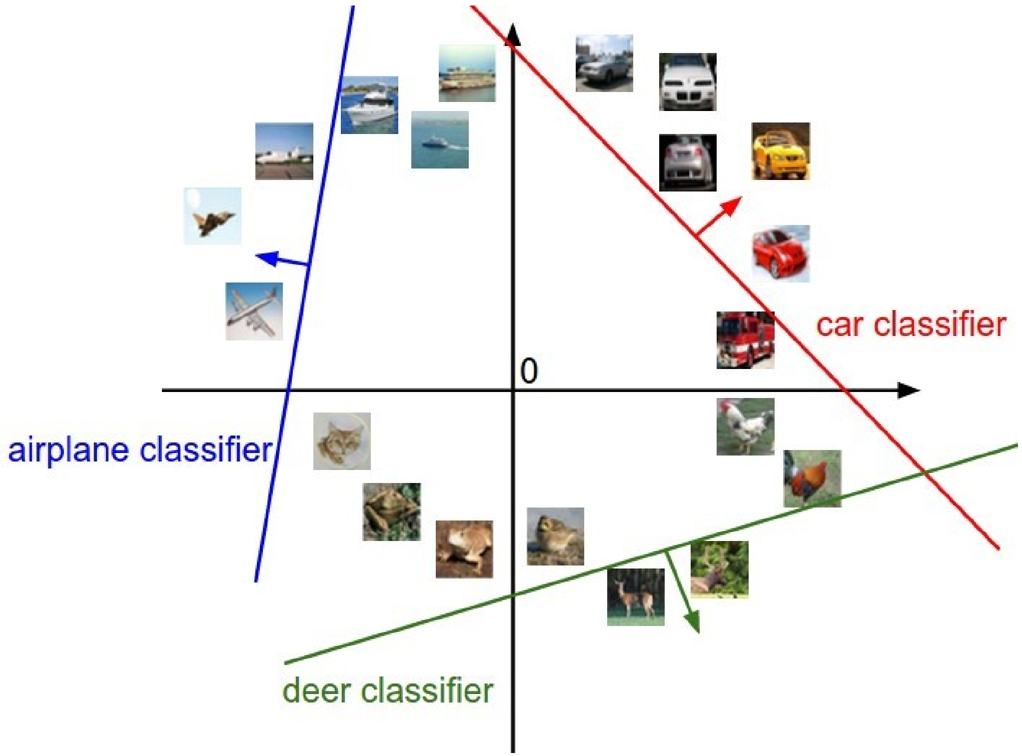
Weight matrix a collection of scoring functions, one per class

Prediction is vector where jth component is "score" for jth class.

Diagram by: Karpathy, Fei-Fei

Geometric Intuition*

What does a linear classifier look like* in 2D?

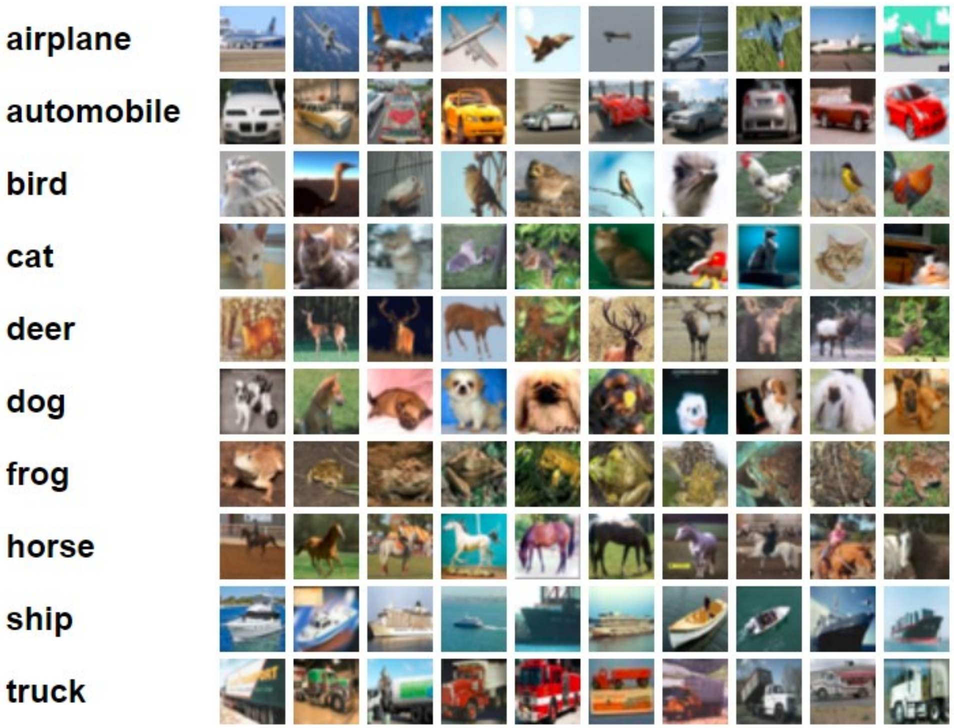


*2D is good for vague intuitions, but ML typically deals with at least dozens if not *thousands* of dimensions. Your intuitions about space and geometry from living in 3D are **completely wrong** in high dimensions. Never trust people who show you 2D diagrams and write "Intuition" in the slide title. See: *On the Surprising Behavior of Distance Metrics in High Dimensional Space.* Charu, Hinneburg, Keim. ICDT 2001

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

Visual Intuition

CIFAR 10:
32x32x3 Images, 10 Classes



- Turn each image into feature by unrolling all pixels
- Fit 10 linear models

Slide credit: Karpathy & Fei-Fei

Guess The Classifier

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?

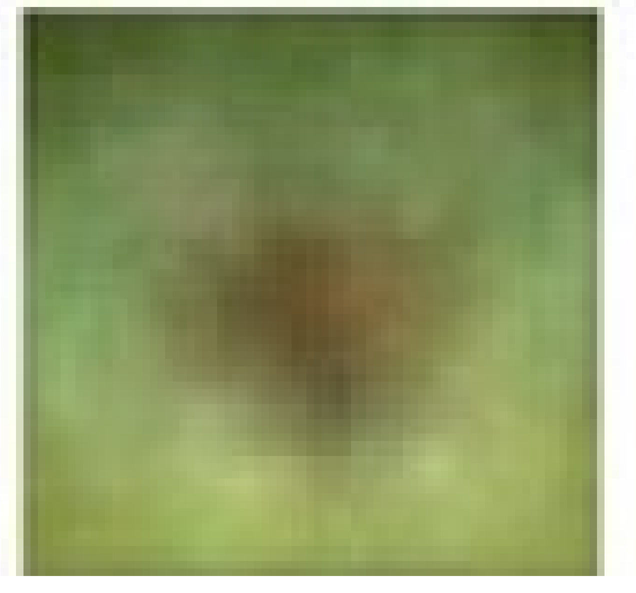


Diagram credit: Karpathy & Fei-Fei

Guess The Classifier

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.

Ship or Dog?

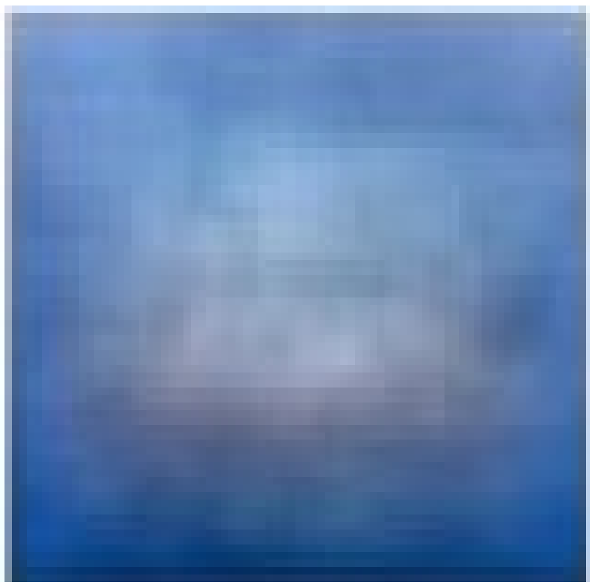


Diagram credit: Karpathy & Fei-Fei

Interpreting a Linear Classifier

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.

Diagram credit: Karpathy & Fei-Fei

Objective 1: Multiclass SVM

Inference (x): $\arg\max_{k} \ (\boldsymbol{W}\boldsymbol{x})_k$

(Take the class whose weight vector gives the highest score)

Objective 1: Multiclass SVM

Inference (x,y): $\arg\max_{k} \ (\boldsymbol{W}\boldsymbol{x})_k$

(Take the class whose weight vector gives the highest score)

Training ($\mathbf{x}_i$,y_i):

$$\arg\min_{\boldsymbol{W}} \lambda \|\boldsymbol{W}\|_2^2 + \sum_{i}^{n} \sum_{j \neq y_i} \underbrace{\max(0, (\boldsymbol{W}\boldsymbol{x}_i)_j - (\boldsymbol{W}\boldsymbol{x}_i)_{y_i} + m)}$$

Regularization

Over all data points

For every class j that's NOT the correct one (y_i)

Pay no penalty if prediction for class yi is bigger than j by m ("margin"). Otherwise, pay proportional to the score of the wrong class.

Objective: Multiclass SVM

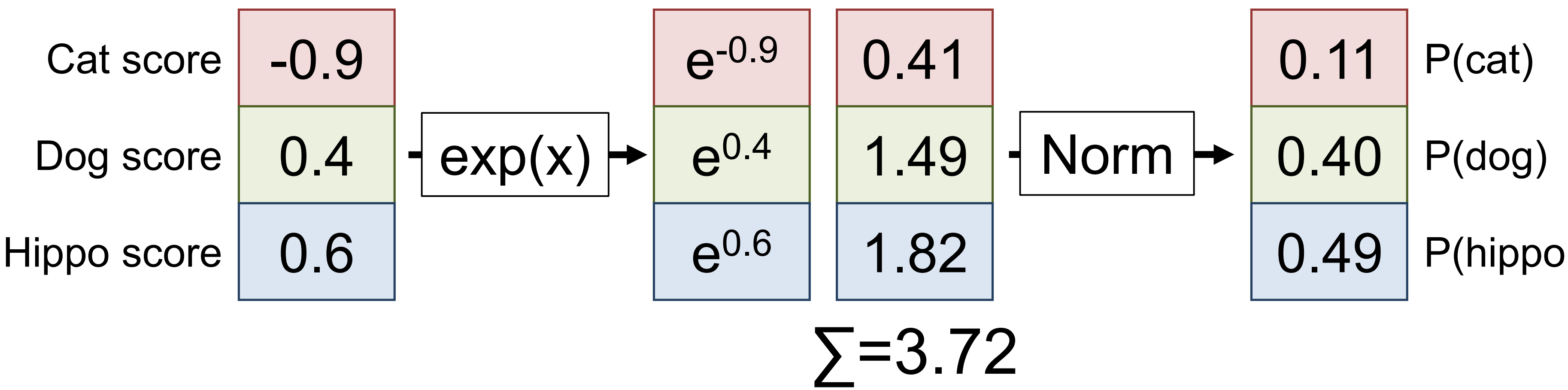
How on earth do we optimize:

$$\arg\min_{\boldsymbol{W}} \boldsymbol{\lambda}||\boldsymbol{W}||_{\mathbf{2}}^{\mathbf{2}} + \sum_{i}^{n} \sum_{j \neq y_i} \max(0, (\boldsymbol{W}\boldsymbol{x}_i)_j - (\boldsymbol{W}\boldsymbol{x}_{\boldsymbol{i}})_{y_i} + m)$$

Hold that thought!

Preliminaries

Converting Scores to “Probability Distribution”



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

Objective 2: Softmax

Inference (x): $\arg\max_{k} \; (\boldsymbol{W}\boldsymbol{x})_k$

(Take the class whose weight vector gives the highest score)

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

Why can we skip the exp/sum exp thing to make a decision?

Objective 2: Softmax

Inference (x): $\arg\max_{k} \ (\boldsymbol{Wx})_k$

(Take the class whose weight vector gives the highest score)

Training ($\mathbf{x}_i$,y_i):

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

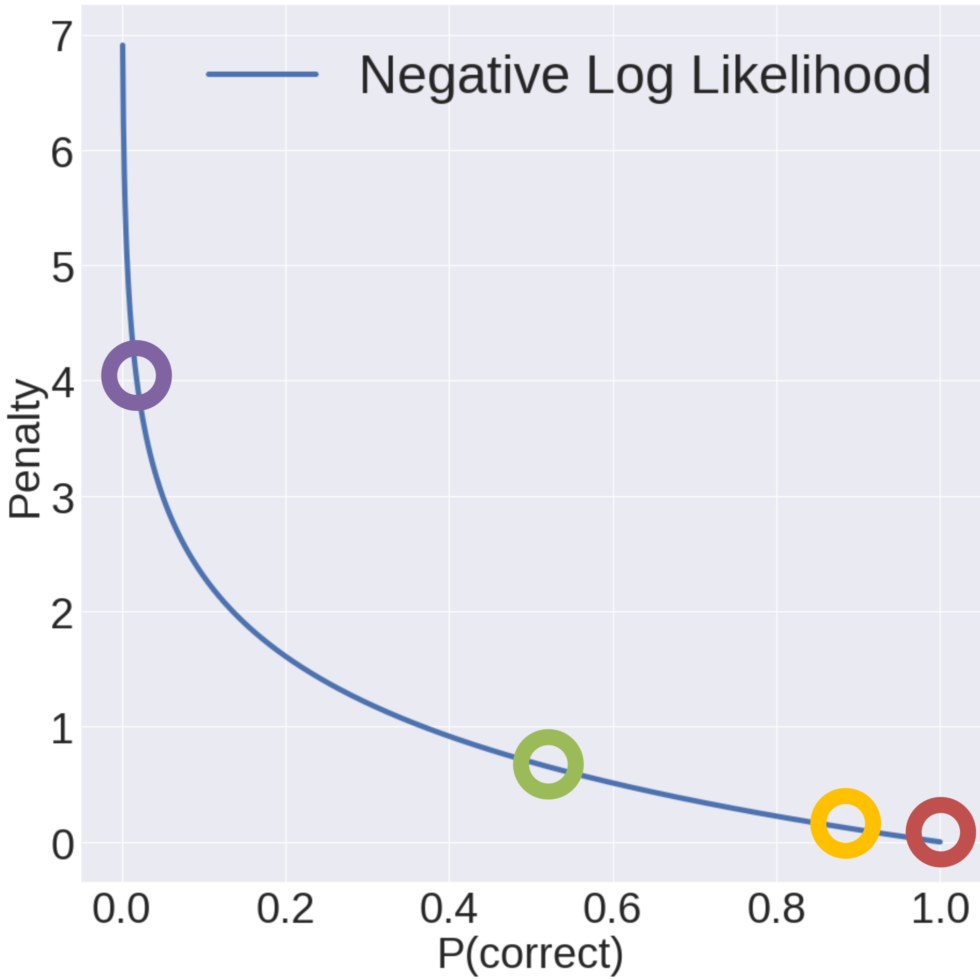
P(correct class)

Regularization

Over all data points

Pay penalty for negative log-likelihood of correct class

Objective 2: Softmax



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!

Next Class

- How do we optimize more complex stuff?
- A bit more ML