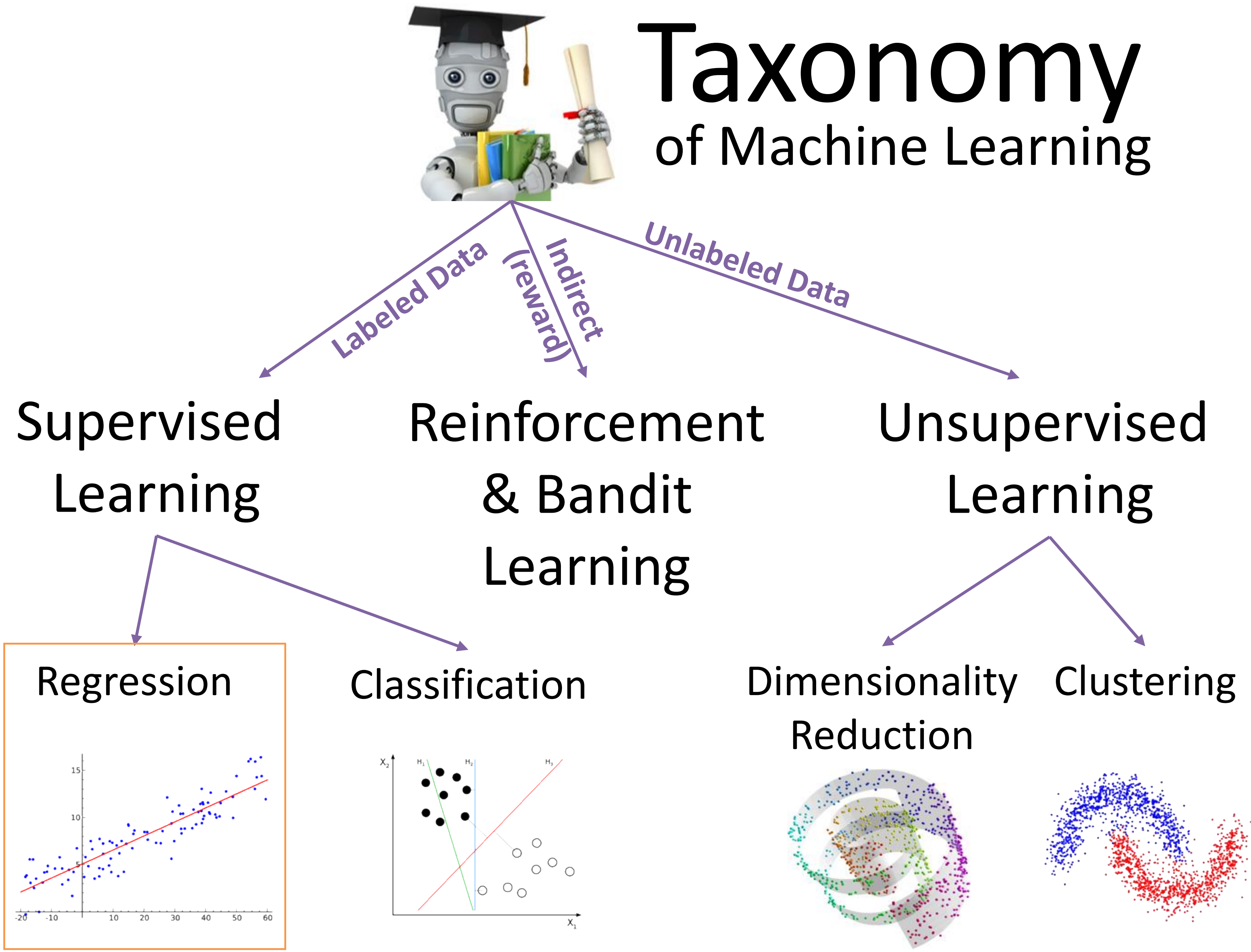
CS150: Database & Datamining

Lecture 22: Analytics & Machine Learning IV

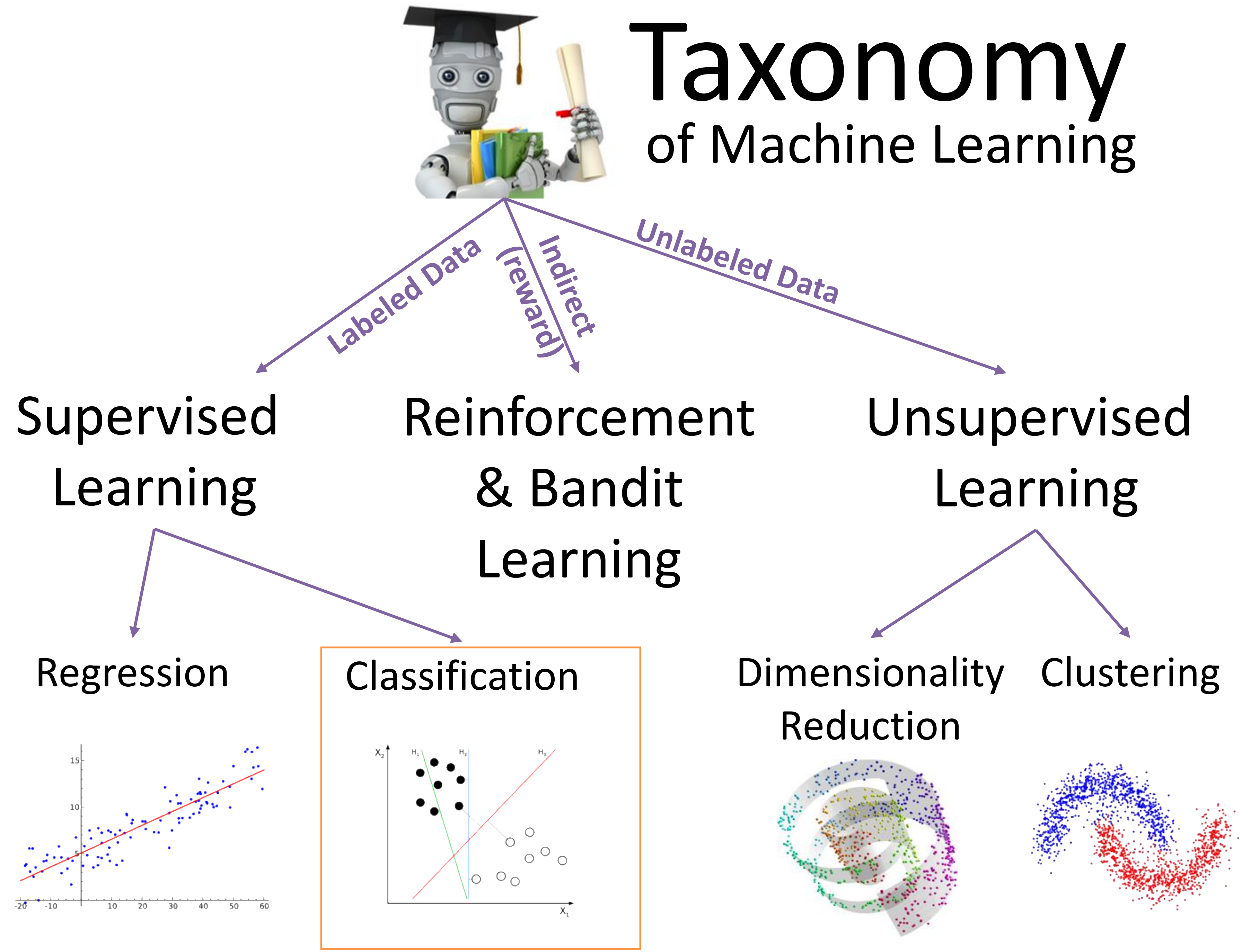
Xuming He

Spring 2019

Acknowledgement: Slides are adopted from the Berkeley course CS186 by Joey Gonzalez and Joe Hellerstein, Stanford CS145 by Peter Bailis.



Taxonomy
of Machine Learning
Labeled Data
Indirect (reward)
Unlabeled Data
Supervised Learning
Reinforcement & Bandit Learning
Unsupervised Learning
Regression
Classification
Dimensionality Reduction
Clustering


Taxonomy
of Machine Learning
Labeled Data
Indirect
(reward)
Unlabeled Data
Supervised Learning
Reinforcement & Bandit Learning
Unsupervised Learning
Regression
Classification
Dimensionality Reduction
Clustering

Spam **Classification**

- **Goal:** given the **text in an** email **predict** whether it is **spam**
- **Training Data:**

Content	Is Spam
Viagra & Cialas half-off today…	SPAM
Class is Cancelled today	NOT SPAM
Deals on new Autos	SPAM
Receipt from Ritual Coffee …	NOT SPAM

- **First best solution?**
  - What is wrong with this?

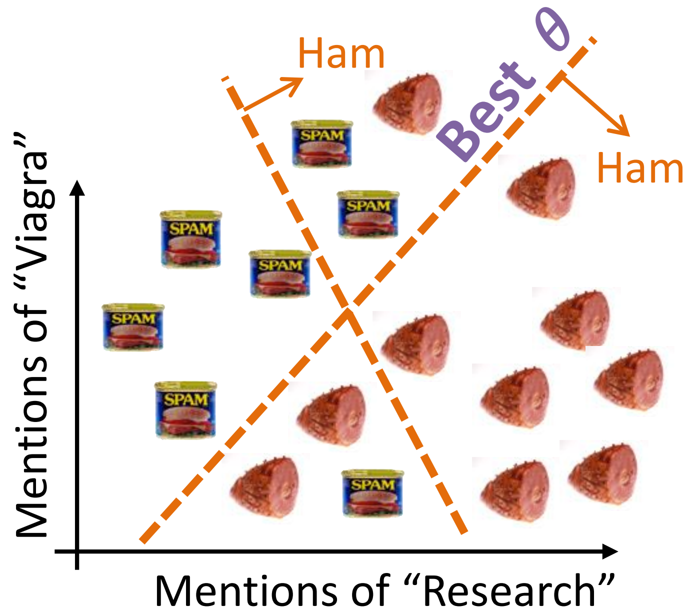
```
def predSpam(doc):
  if “Viagra” in doc:
    return True
  elif “Cialas” in doc:
    return True
  elif “Class” in doc:
    return False
  elif “Deals” in doc:
    return True
  else:
    return False
```

- **Why is Spam Classification Hard?**
  - Easy for humans to **recognize**
  - **Difficult** to formally describe (as an algorithm)
  - **Personal:** different people have different tastes in Spam
  - Good candidate for Machine Learning, the second best solution

Spam **Classification**

- **Goal:** given the **text in an** email **predict** whether it is **spam**
- **Training Data:**

Content	Is Spam
Viagra & Cialas half-off today…	SPAM
Class is Cancelled today	NOT SPAM
Deals on new Autos	SPAM
Receipt from Ritual Coffee …	NOT SPAM



- **Machine Learning:**
  - Learn a function that generalizes the relation:

$$\mathbf{F}(\mathit{Content};\ \theta) \rightarrow \mathit{isSpam}$$

  - **F**: is the model type
  - θ: are the model parameters

- Machine learning alg. search for the "best" θ

Common Classification Models

- **Most models predict the probability**
  - Why would probability be helpful?
- **Nearest Neighbor:** *works embarrassingly well*
  - return the label of the nearest training point to the query point
- **Logistic Regression:** *widely used and simple*
  - Similar to least squares regression but for classification
- **Naïve Bayes:** *occasionally used*
  - Classic model based on Bayes Rule
- **Support Vector Machines:** *kernel methods*
  - Capable of automatically growing model size with data
- **Deep Learning**: *more on this soon …*

Logistic Regression

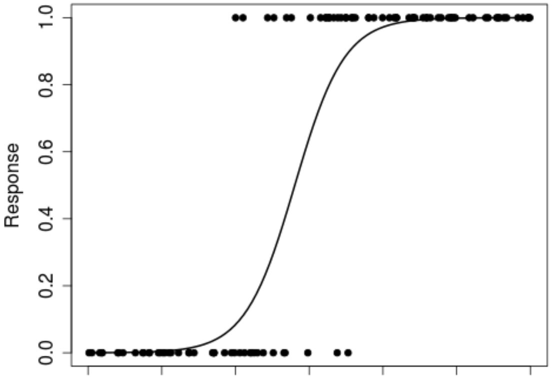
- Basic Model:

$$\mathbf{P}\left(y|x,\theta\right) = \sigma\left(y(\theta^T x)\right)$$
$$= \frac{1}{1+\exp\left(-y(\theta^T x)\right)}$$

Note that y is either +1 or -1

- Logistic Function:

$$\sigma(z) = \frac{1}{1+\exp(-z)}$$



- Symmetric

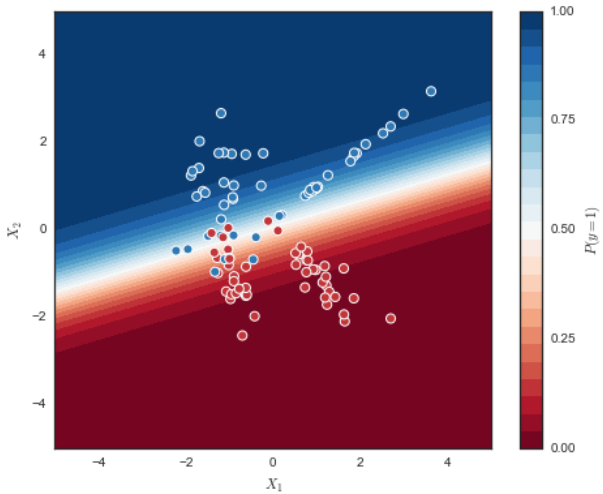
$$1-\sigma(a) = \frac{\exp(-a)}{1+\exp(-a)} = \frac{1}{\exp(a)+1} = \sigma(-a)$$

- Gradient

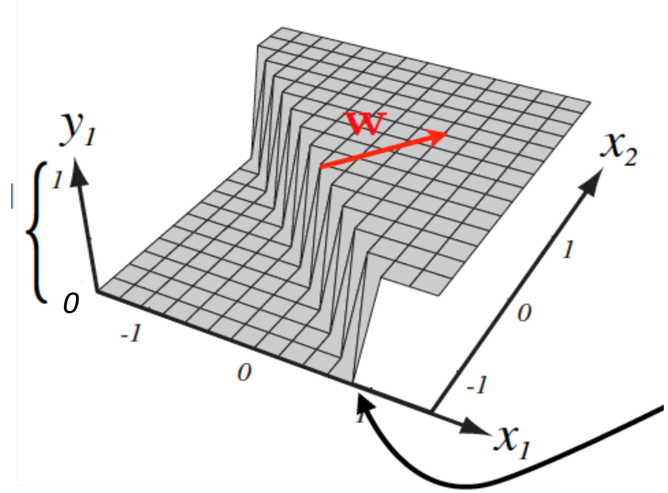
$$\sigma'(a) = \frac{\exp(-a)}{(1+\exp(-a))^2} = \sigma(a)(1-\sigma(a))$$

Logistic Regression

- 2D example:



X		Y
x_1	x_2	
3.9	3.1	+1
0.2	0.1	-1
4.2	0.9	-1
1.1	2.9	+1

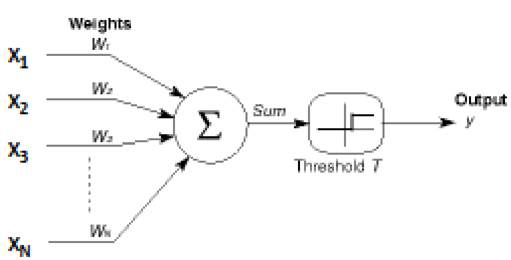


Class decision boundary

What a linear classifier does?

- If its input is within a specific angle of its weight
  - If the input pattern matches the weight pattern closely enough

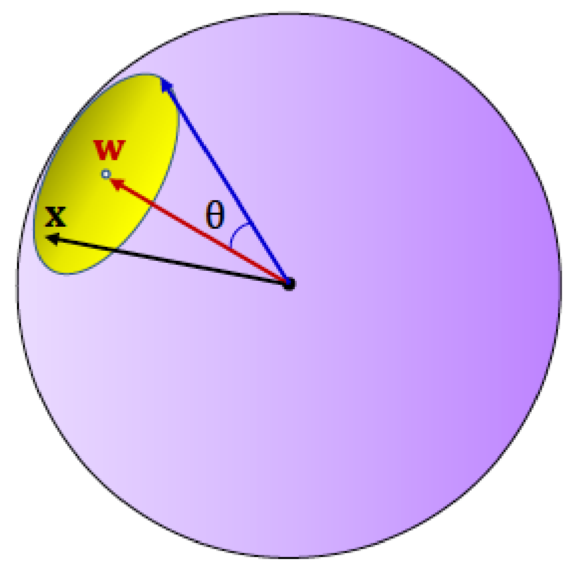
$$P(y=1) > 0.5 \Leftrightarrow \frac{1}{1+\exp(-\mathbf{w}^T\mathbf{x})} > 0.5 \Leftrightarrow \mathbf{w}^T\mathbf{x} > 0$$



$$\mathbf{x}^{\mathrm{T}}\mathbf{w} > \mathrm{T}$$

$$\Rightarrow \cos\theta > \frac{\mathrm{T}}{|\mathbf{x}||\mathbf{w}|}$$

$$\Rightarrow \theta < \cos^{-1}\left(\frac{\mathrm{T}}{|\mathbf{x}||\mathbf{w}|}\right)$$



Learning the Logistic Regression Model

- How do we fit the Logistic Regression model?
  - method of maximum likelihood
- Select the best θ by maximizing prob. of data
  - Solve the following **convex** optimization problem

$$\hat{\theta} = \arg\min_{\theta \in \mathbb{R}^p} \quad \frac{1}{n}\sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i(\theta^T x_i)\right)\right) + \lambda R(\theta)$$

  - Regularized using same techniques as regression

- Optimized using numerical methods
  - **SGD**: Stochastic Gradient Descent

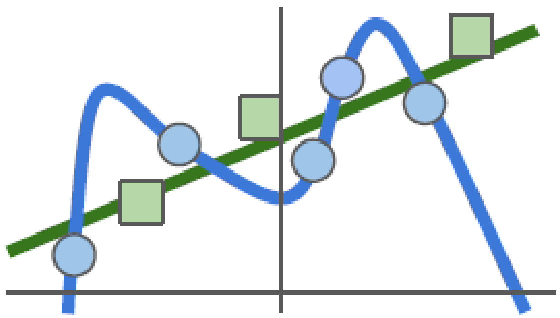
Learning with regularization

- Constraints on hypothesis space

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

Data loss: Model predictions should match training data

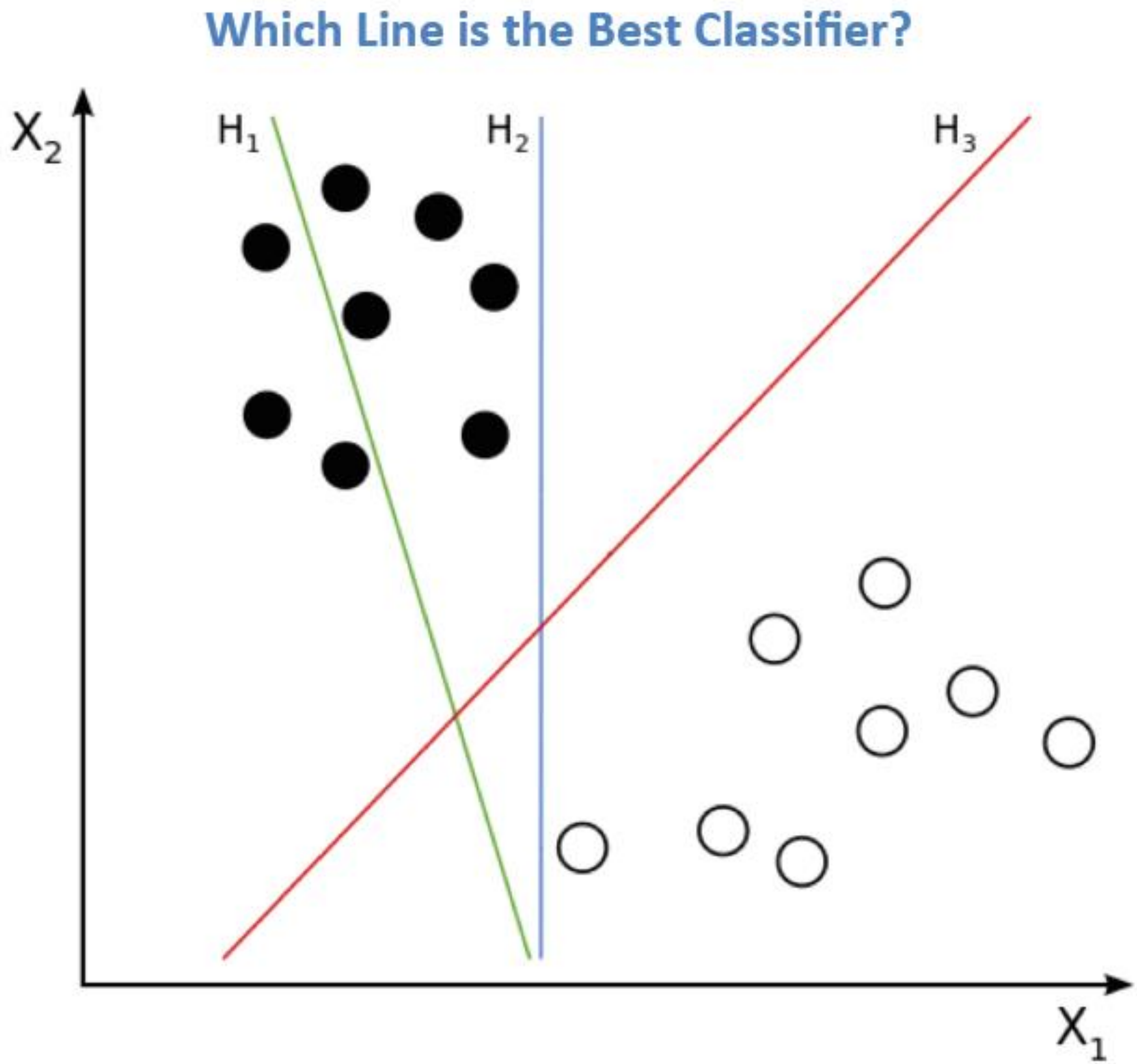
Regularization: Model should be "simple", so it works on test data



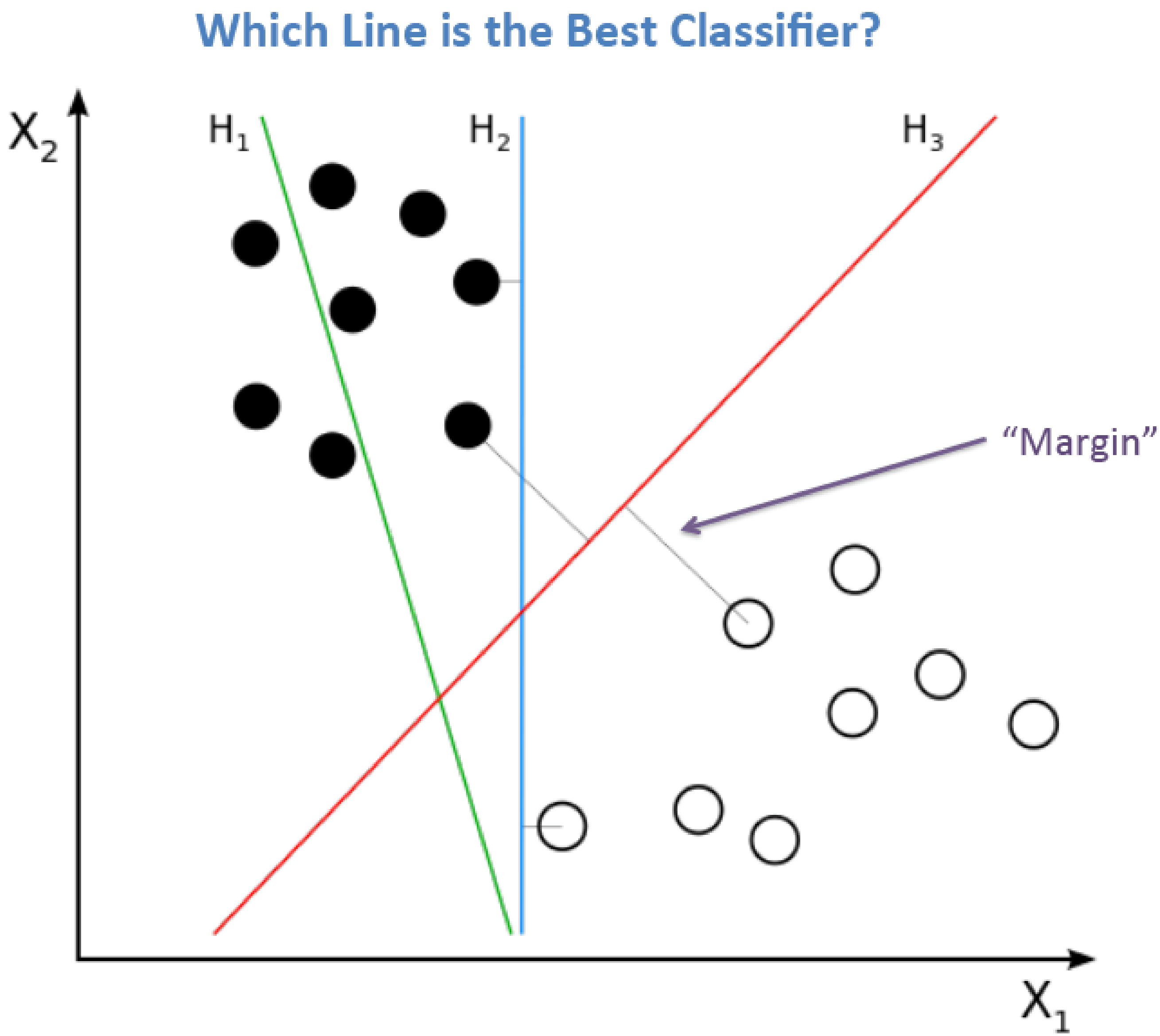
In common use:

L2 regularization	$R(W) = \sum_k \sum_l W_{k,l}^2$
L1 regularization	$R(W) = \sum_k \sum_l \lvert W_{k,l} \rvert$
Elastic net (L1 + L2)	$R(W) = \sum_k \sum_l \beta W_{k,l}^2 + \lvert W_{k,l} \rvert$

Support Vector Machines

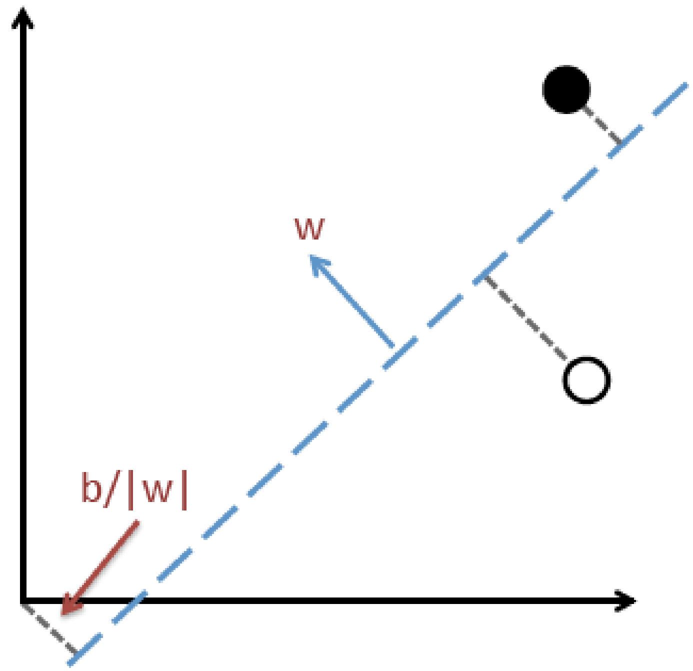


Support Vector Machines



Hyperplane Distance

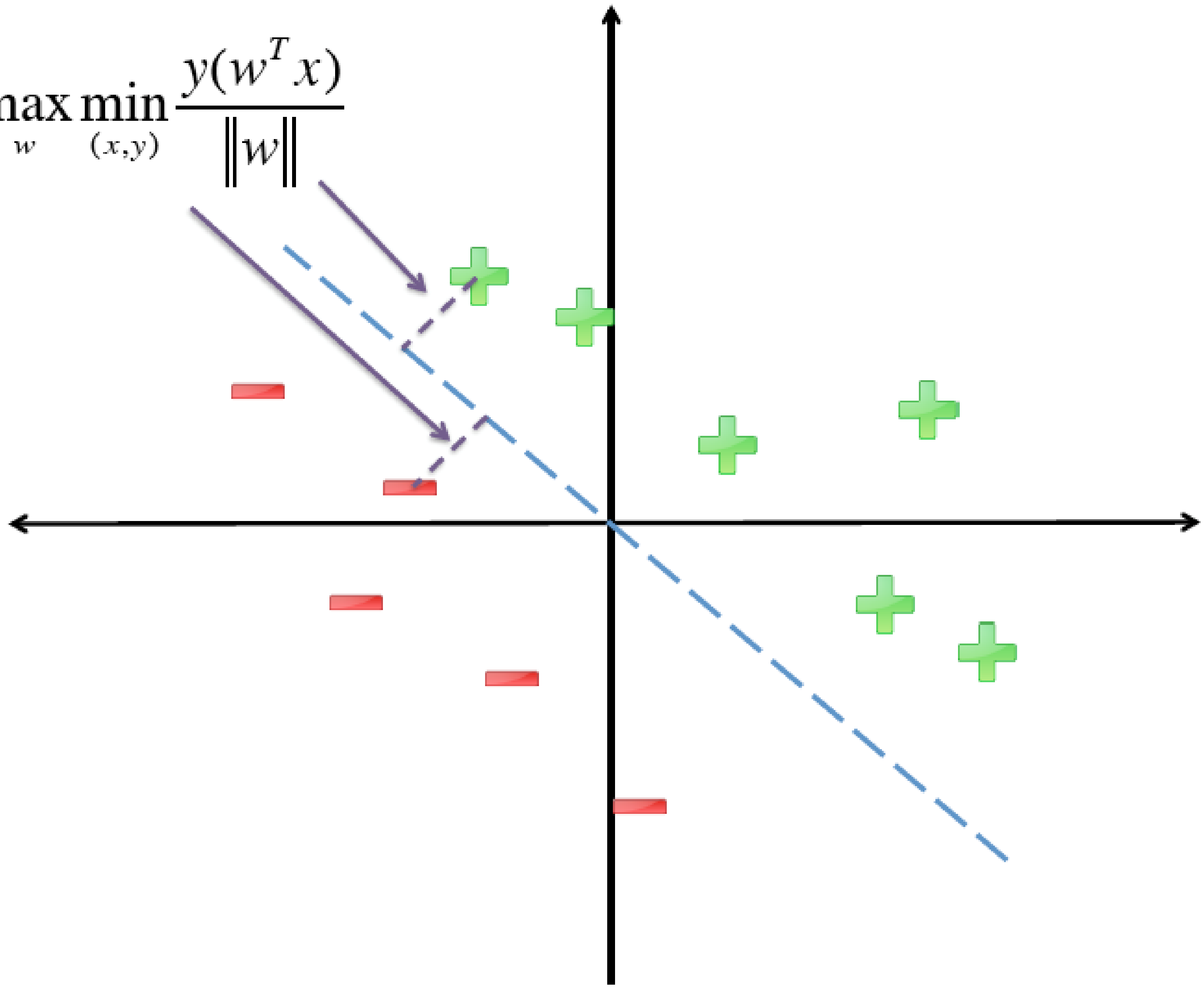
- Line is a 1D, Plane is 2D
- Hyperplane is many D
  - Includes Line and Plane
- Defined by (w,b)
- Distance: $\dfrac{|w^T x - b|}{\|w\|}$
- Signed Distance: $\dfrac{w^T x - b}{\|w\|}$



Linear Model = un-normalized signed distance!

Margin

$$\gamma = \max_{w} \min_{(x,y)} \frac{y(w^T x)}{\|w\|}$$

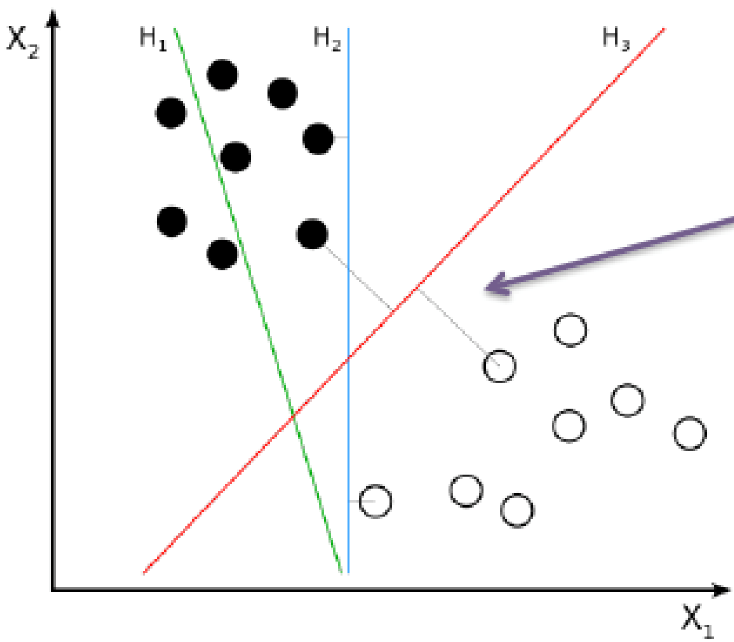


How to maximize margin

Choose w that maximizes:

$$\underset{w,b}{\operatorname{argmax}}\left[\min_{(x,y)}\frac{y\left(w^T x - b\right)}{\|w\|}\right]$$

Margin

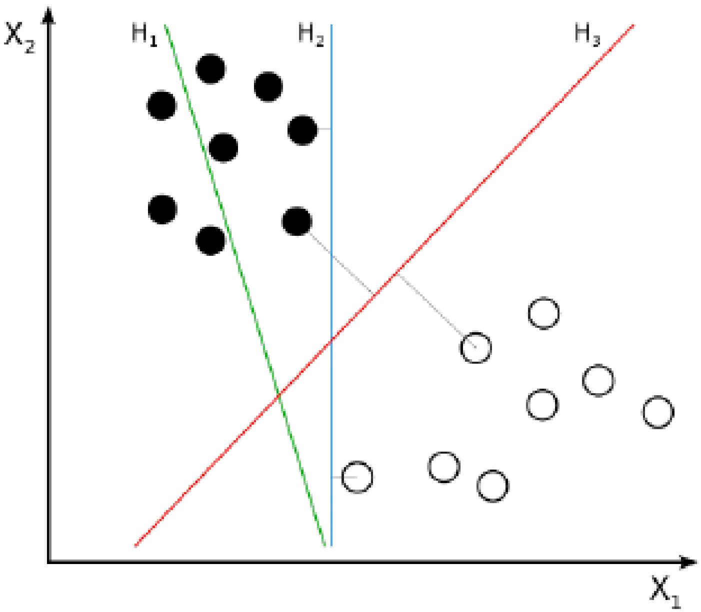


How to maximize margin

$$\underset{w,b}{\operatorname{argmax}}\left[\min_{(x,y)}\frac{y\left(w^T x-b\right)}{\|w\|}\right]$$

$$\equiv \underset{w,b:\ \|w\|=1}{\operatorname{argmax}}\left[\min_{(x,y)} y\left(w^T x-b\right)\right]$$

Hold Denominator Fixed



Suppose we instead enforce:

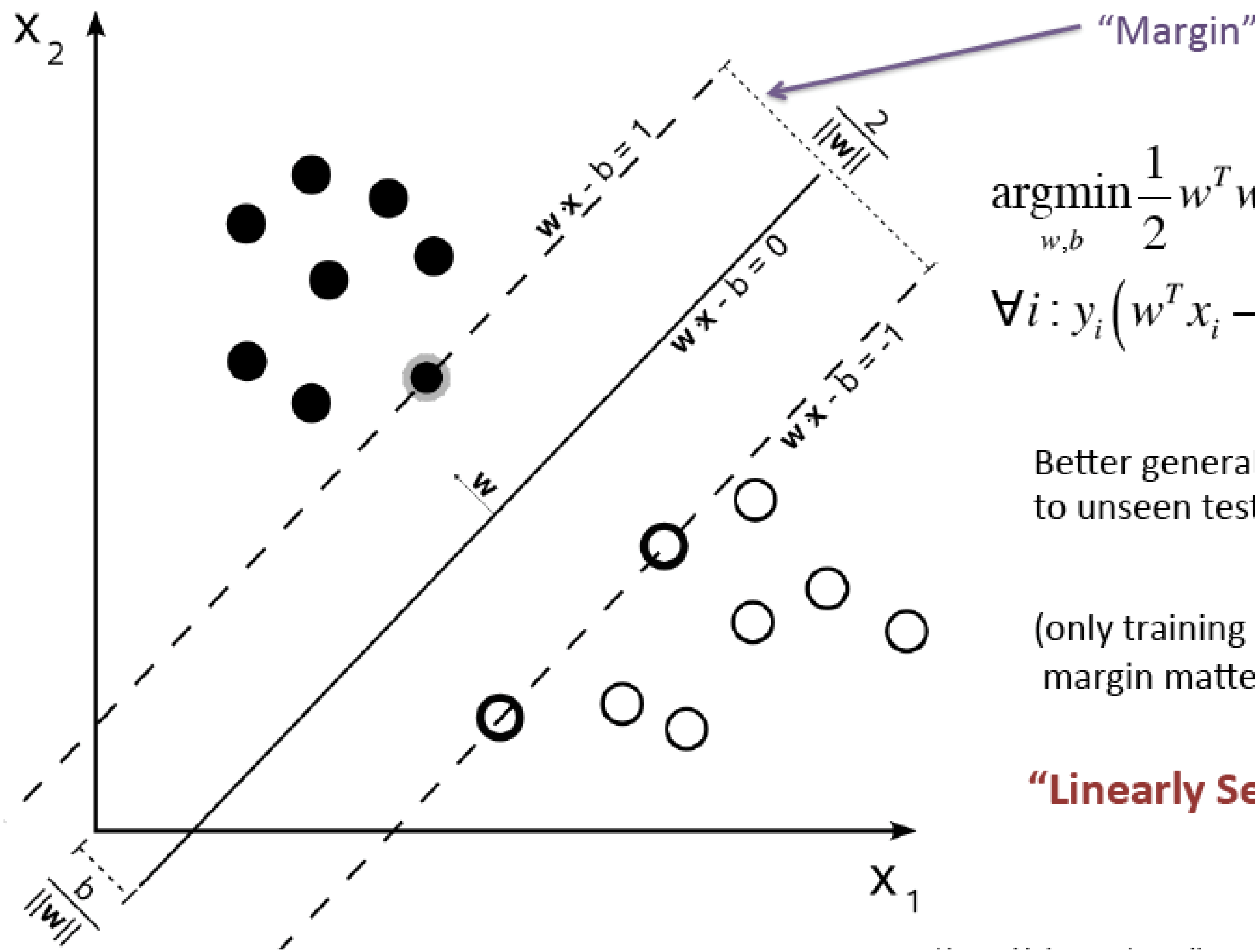
$$\min_{(x,y)} y\left(w^T x-b\right)=1$$

Hold Numerator Fixed

Then:

$$=\underset{w,b}{\operatorname{argmin}}\|w\| \equiv \underset{w,b}{\operatorname{argmin}}\|w\|^2$$

Max-margin classifier (SVM)



$$\underset{w,b}{\operatorname{argmin}} \frac{1}{2} w^T w \equiv \frac{1}{2} \|w\|^2$$

$$\forall i : y_i \left(w^T x_i - b \right) \geq 1$$

Better generalization
to unseen test examples

(only training data on
margin matter)

"Linearly Separable"

Multiclass Linear classifier

$$\underbrace{\boxed{f(x,W)}}_{10\times1} = \underbrace{\boxed{W}}_{10\times3072}\overbrace{\boxed{x}}^{3072\times1} + \underbrace{\boxed{b}}_{10\times1}$$

Image

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

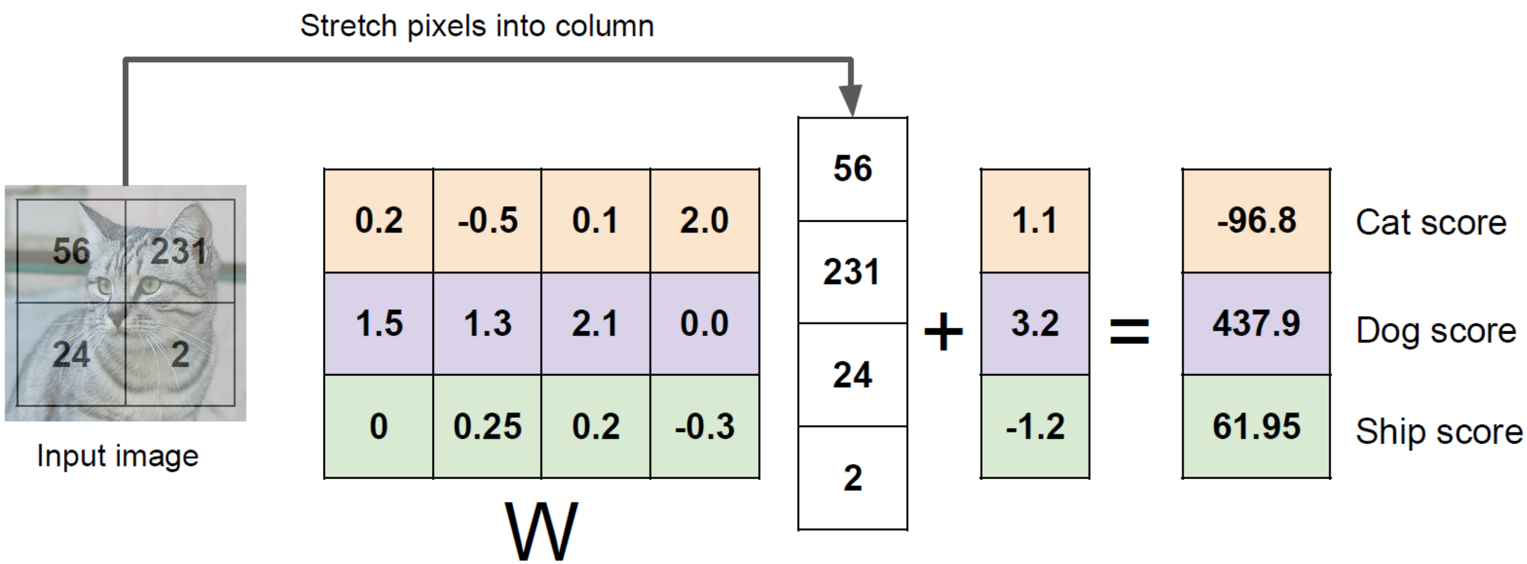
$\longrightarrow f(\mathbf{x},\mathbf{W}) \longrightarrow$ **10** numbers giving class scores

$\uparrow$

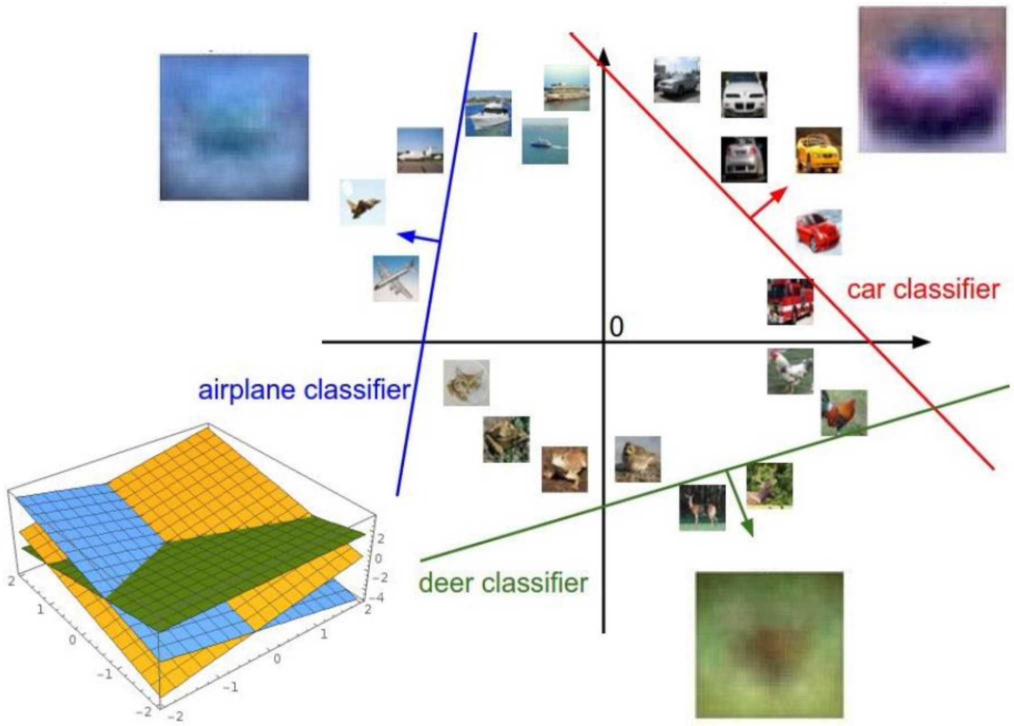
W
parameters
or weights

Multiclass Linear classifier

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



Interpreting a Linear classifier



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

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

Instance-based learning

- Alternative to parametric models are non-parametric models
- These are typically simple methods for approximating discrete-valued or real-valued target functions (they work for classification or regression problems)
- Learning amounts to simply storing training data
- Test instances classified using similar training instances
- Embodies often sensible underlying assumptions:
    - Output varies smoothly with input
    - Data occupies sub-space of high-dimensional input space

Nearest Neighbors

- Training example in Euclidean space: $\mathbf{x} \in \Re^d$
- Idea: The value of the target function for a new query is estimated from the known value(s) of the nearest training example(s)
- Distance typically defined to be Euclidean:

$$||\mathbf{x}^{(a)} - \mathbf{x}^{(b)}||_2 = \sqrt{\sum_{j=1}^{d} (x_j^{(a)} - x_j^{(b)})^2}$$

Algorithm:

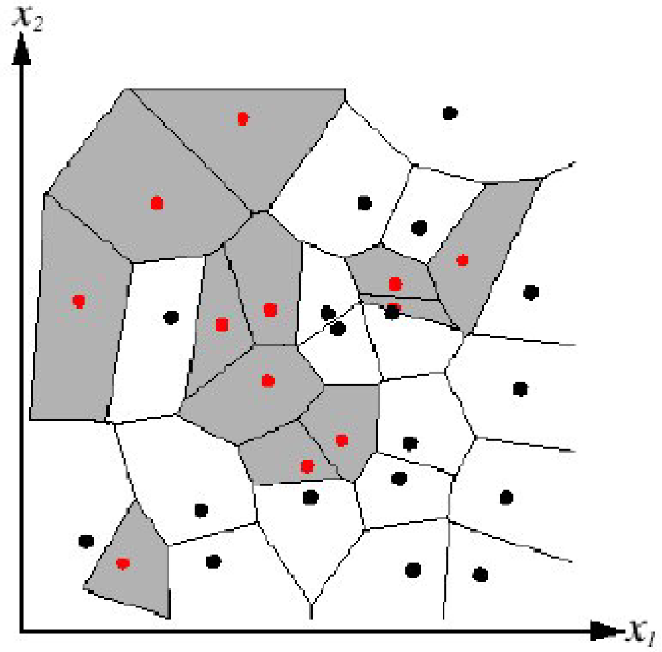
1. Find example $(\mathbf{x}^*, t^*)$ (from the stored training set) closest to the test instance $\mathbf{x}$. That is:

$$\mathbf{x}^* = \underset{\mathbf{x}^{(i)} \in \text{train. set}}{\operatorname{argmin}} \ \text{distance}(\mathbf{x}^{(i)}, \mathbf{x})$$

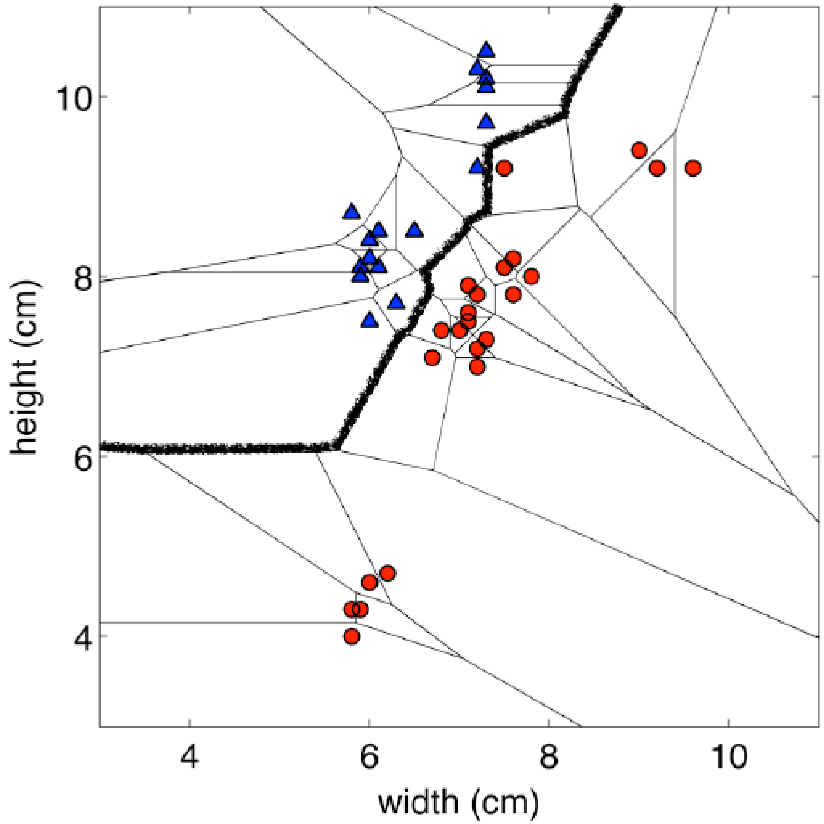
2. Output $y = t^*$

Nearest Neighbors

- Nearest neighbor algorithm does not explicitly compute decision boundaries, but these can be inferred
- Decision boundaries: Voronoi diagram visualization
  - show how input space divided into classes
  - each line segment is equidistant between two points of opposite classes

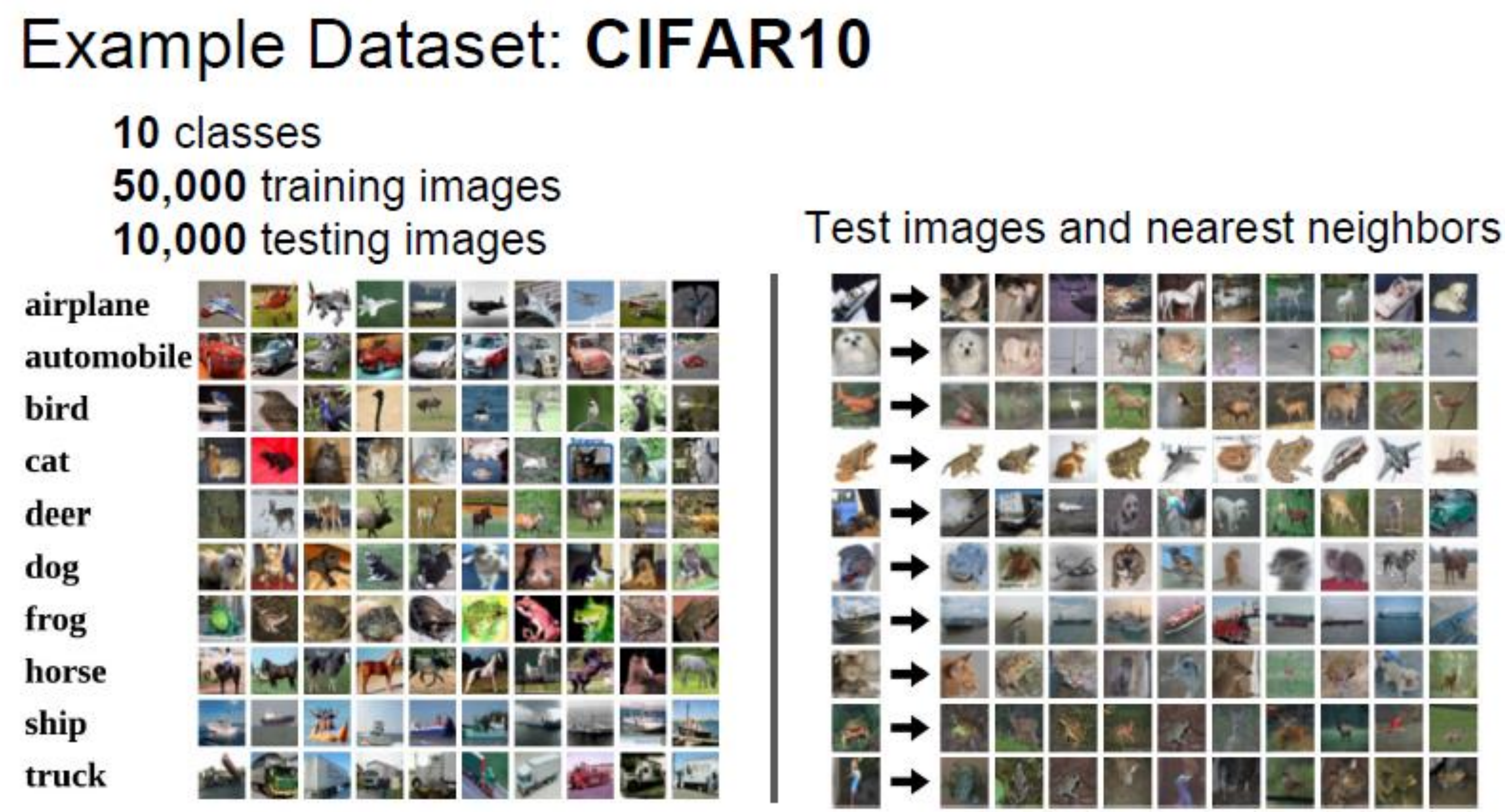


Nearest Neighbors



Example: 2D decision boundary

Vision example



Vision example

Distance Metric to compare images

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

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

Vision example

```
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

Nearest Neighbor classifier

Q: With N examples, how fast are training and prediction?

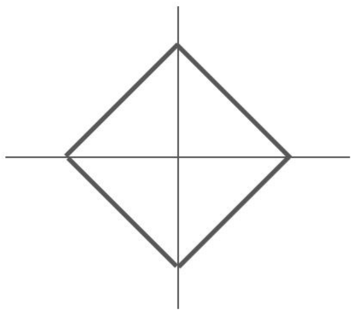
A: Train O(1), predict O(N)

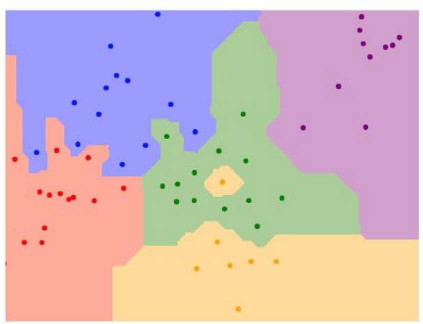
This is bad: we want classifiers that are **fast** at prediction; **slow** for training is ok

KNN: Distance metric

L1 (Manhattan) distance

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

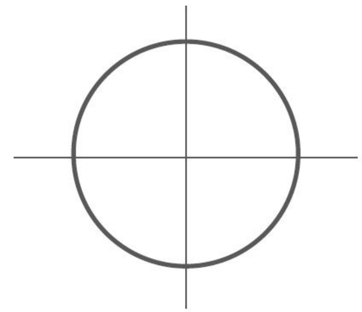


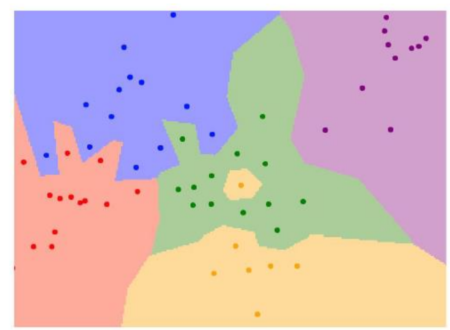


K = 1

L2 (Euclidean) distance

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





K = 1

Practical issues

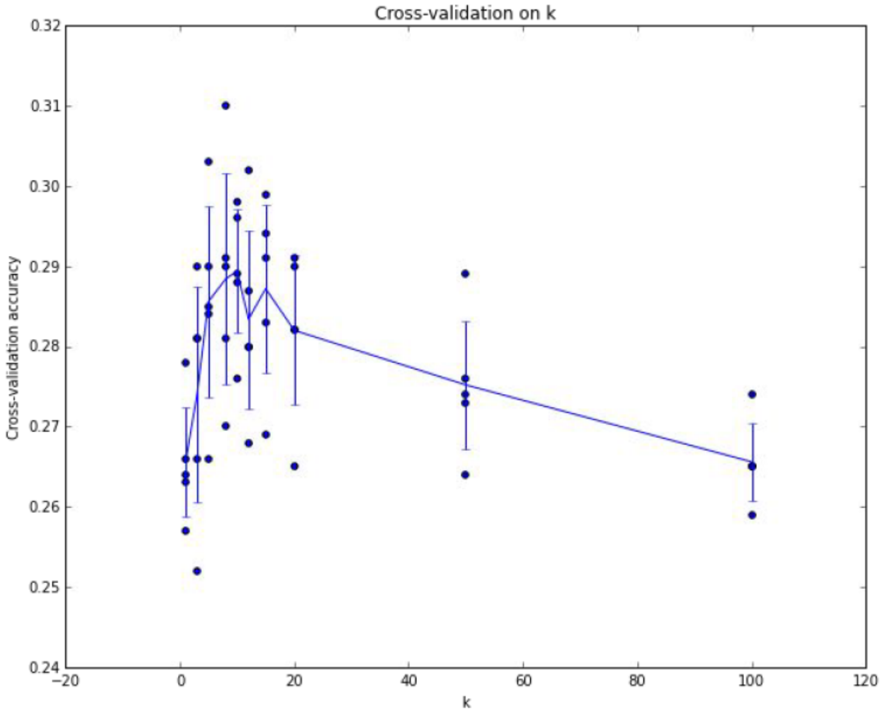
Hyperparameters

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

These are **hyperparameters**: choices about the algorithm that we set rather than learn

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

Model selection



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

Each point: single outcome.

The line goes through the mean, bars indicated standard deviation

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

KNN in CV

- Decent performance when lots of data

- Yann LeCunn – MNIST Digit Recognition
  - Handwritten digits
  - 28x28 pixel images: $d = 784$
  - 60,000 training samples
  - 10,000 test samples
- Nearest neighbour is competitive

	Test Error Rate (%)
Linear classifier (1-layer NN)	12.0
K-nearest-neighbors, Euclidean	5.0
K-nearest-neighbors, Euclidean, deskewed	2.4
K-NN, Tangent Distance, 16x16	1.1
K-NN, shape context matching	0.67
1000 RBF + linear classifier	3.6
SVM deg 4 polynomial	1.1
2-layer NN, 300 hidden units	4.7
2-layer NN, 300 HU, [deskewing]	1.6
LeNet-5, [distortions]	0.8
Boosted LeNet-4, [distortions]	0.7

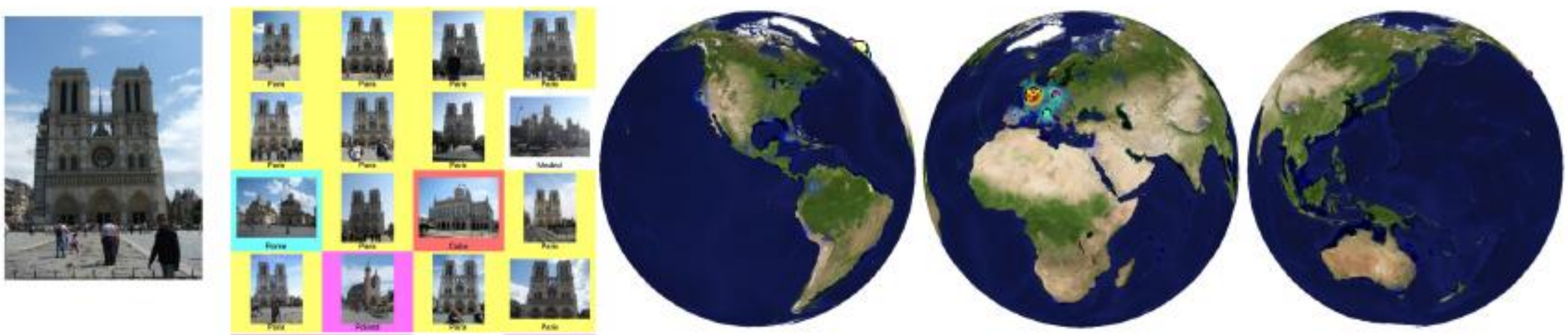
KNN in CV

- Problem: Where (e.g., which country or GPS location) was this picture taken?

[Paper: James Hays, Alexei A. Efros. im2gps: estimating geographic information from a single image. CVPR'08. Project page: http://graphics.cs.cmu.edu/projects/im2gps/]

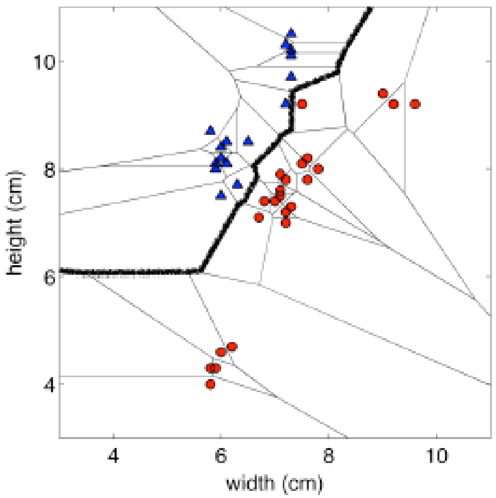
KNN in CV

- Problem: Where (e.g., which country or GPS location) was this picture taken?
  - Get 6M images from Flickr with GPs info (dense sampling across world)
  - Represent each image with meaningful features
  - Do kNN!



[Paper: James Hays, Alexei A. Efros. im2gps: estimating geographic information from a single image. CVPR'08. Project page: http://graphics.cs.cmu.edu/projects/im2gps/]

KNN Summary



- Naturally forms complex decision boundaries; adapts to data density
- If we have lots of samples, kNN typically works well
- Problems:
  - Sensitive to class noise
  - Sensitive to scales of attributes
  - Distances are less meaningful in high dimensions
  - Scales linearly with number of examples

Taxonomy of Machine Learning