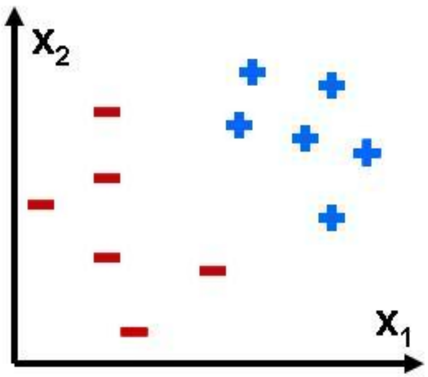
Announcements

- pset1 out today, due Thursday 9/21 (2 weeks)

Classification

$y \in \{0,1\}$ 0: "Negative Class" (e.g., benign tumor)
1: "Positive Class" (e.g., malignant tumor)

Tumor: Malignant / Benign?
Email: Spam / Not Spam?
Video: Viral / Not Viral?

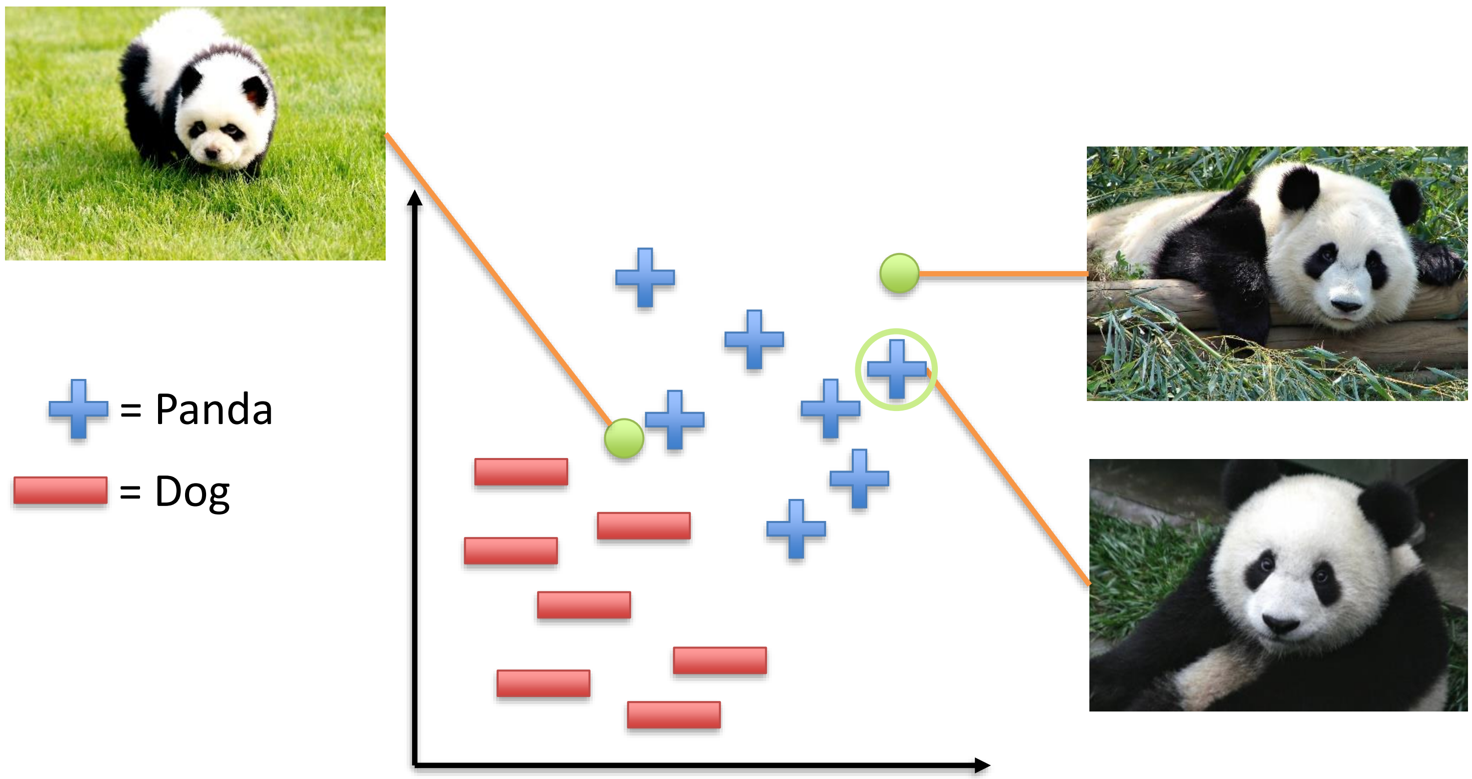


Today

- Classification Intro
  - Nearest Neighbors
  - Learning to classify
    - Error Rates
- Maximum Likelihood
- Bayesian Methods

Many slides adapted from Kate Saenko and Relja Arandjelović

Idea for a simple classifier:
Use similarity (e.g., L2 distance) to labeled examples
i.e., Nearest Neighbor Classifier



Initial Observations:
- Requires a large dataset to work well
- Sensitive to outliers

slido

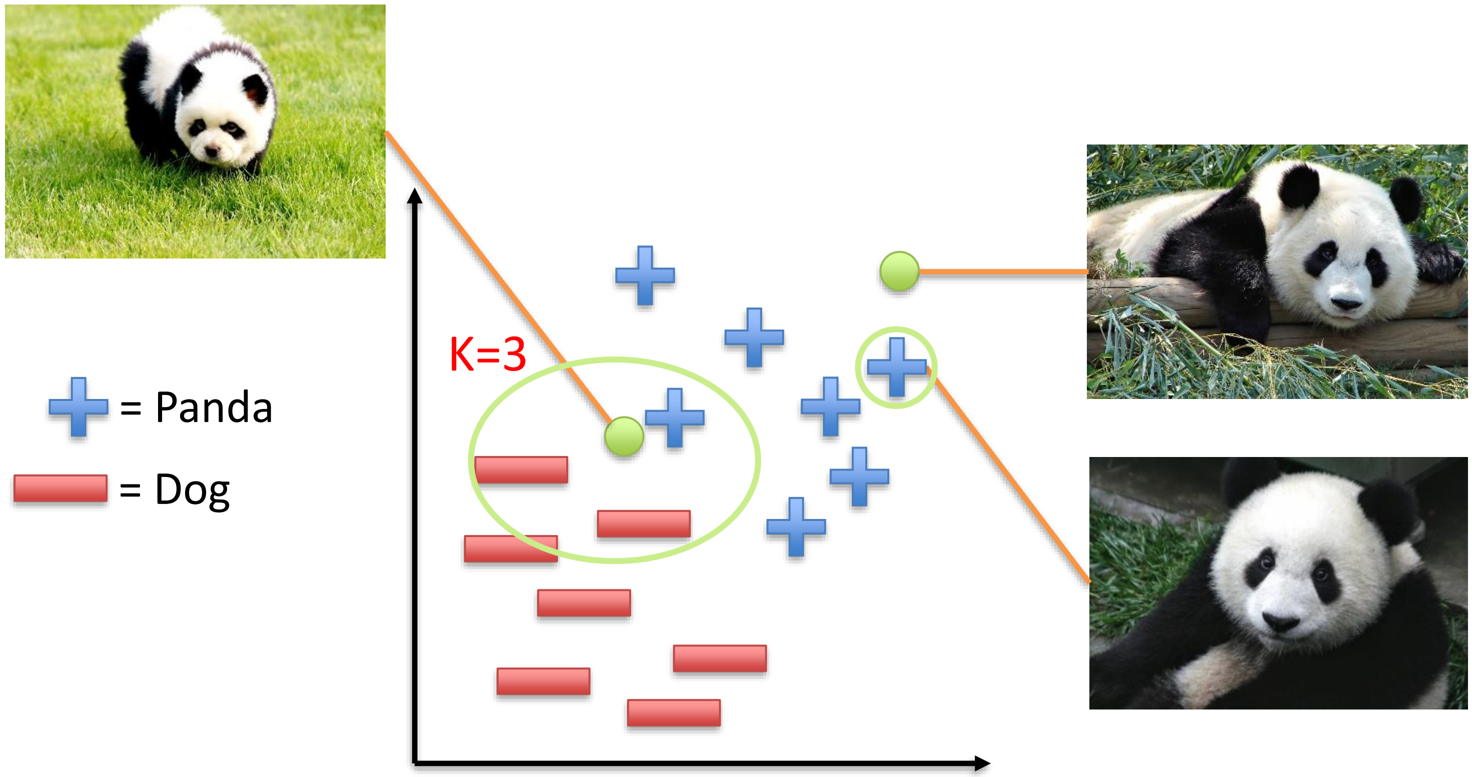
How can we improve our NN classifier (e.g., make it less sensitive to outliers)?

ⓘ Start presenting to display the poll results on this slide.

Idea for a simple classifier:

Use similarity (e.g., L2 distance) to labeled examples

i.e., K-Nearest Neighbor Classifier



Takeaways:

- Selecting K requires tuning, and the optimal value will vary
- Distance functions can also significantly affect performance

How to speed NN up?

Dimensionality reduction?

- Reasonable first step, but typically insufficient

Use GPU?

- Adds lots of complexity
- Insufficient memory
- Overhead for memory copying between CPU & GPU

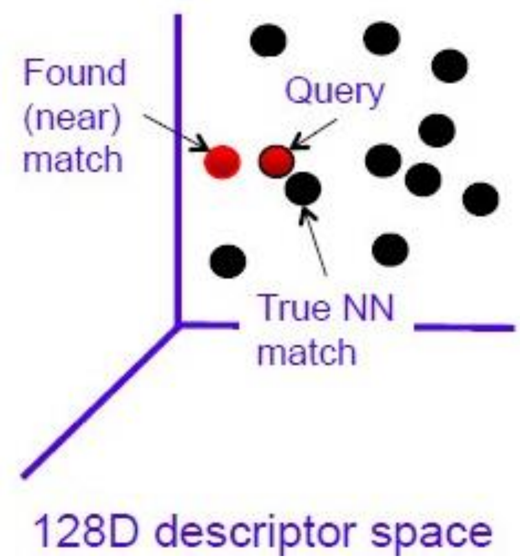
Buy more machines?

- Costs money to buy, maintain, ..
- Adds lots of complexity
- For real-time systems: communication overhead
- Still often insufficient, e.g., if all pairwise distances are needed (e.g. building a neighbourhood graph, clustering, ..)

Finding *approximate* nearest neighbor vectors

- Approximations are not guaranteed to find the nearest neighbor

- Can be much faster, but comes at a cost of missing some nearest matches



Approximate Nearest Neighbors (ANN)

Is finding only approximate nearest neighbors acceptable?

Often there is no choice!

- Use ANN or not = have Google or not

Often it is good enough

- What is this?

?

Big Ben

Approximate Nearest Neighbors (ANN)

Is finding only approximate nearest neighbors acceptable?

Often there is no choice!

- Use ANN or not = have Google or not

Often it is good enough

- What is this?

?

Big Ben

Approximate Nearest Neighbor search: Overview

Approximate the vectors: fast distances, memory savings

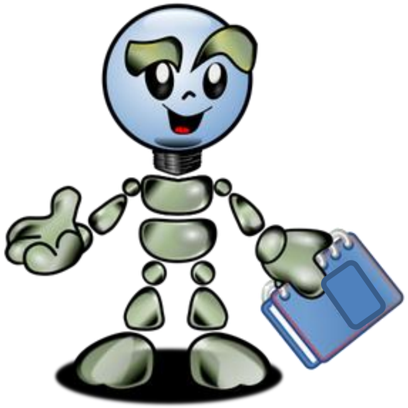
- Hashing
  - LSH, see ITQ, Spectral Hashing, ..
- Vector Quantization
  - **Product Quantization**, see OPQ, Cartesian K-means, ..

Approximate the search

- Non-exhaustive through space partitioning
- Hashing
- **Vector Quantization**
- (Randomized) K-d trees
- Mind the dimensionality

In real life – use a **hybrid** approach

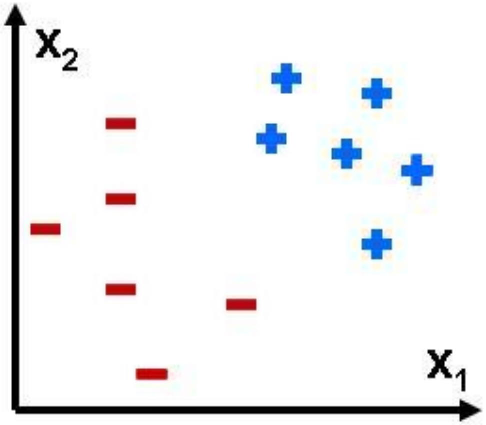
- E.g. Google Goggles/Photos 2011: Vector Quantization + OPQ



Learning to Classify

Intro

Probabilistic Classification



$$D = (x_i, y_i)\text{ : data}$$
$$x \in \mathbb{R}^p$$
$$y \in \{c\}, c = 1, \dots, C$$

- Can model output value directly, but having a probability is often more useful
- Bayes classifier: minimizes the probability of misclassification

$$y = \underset{c}{\operatorname{argmax}}\, P(Y = c|X = x)$$

- Want to model conditional distribution, $P(\mathrm{Y} = \mathrm{y}|X = x)$, then assign label based on it

Example: Temperature Prediction

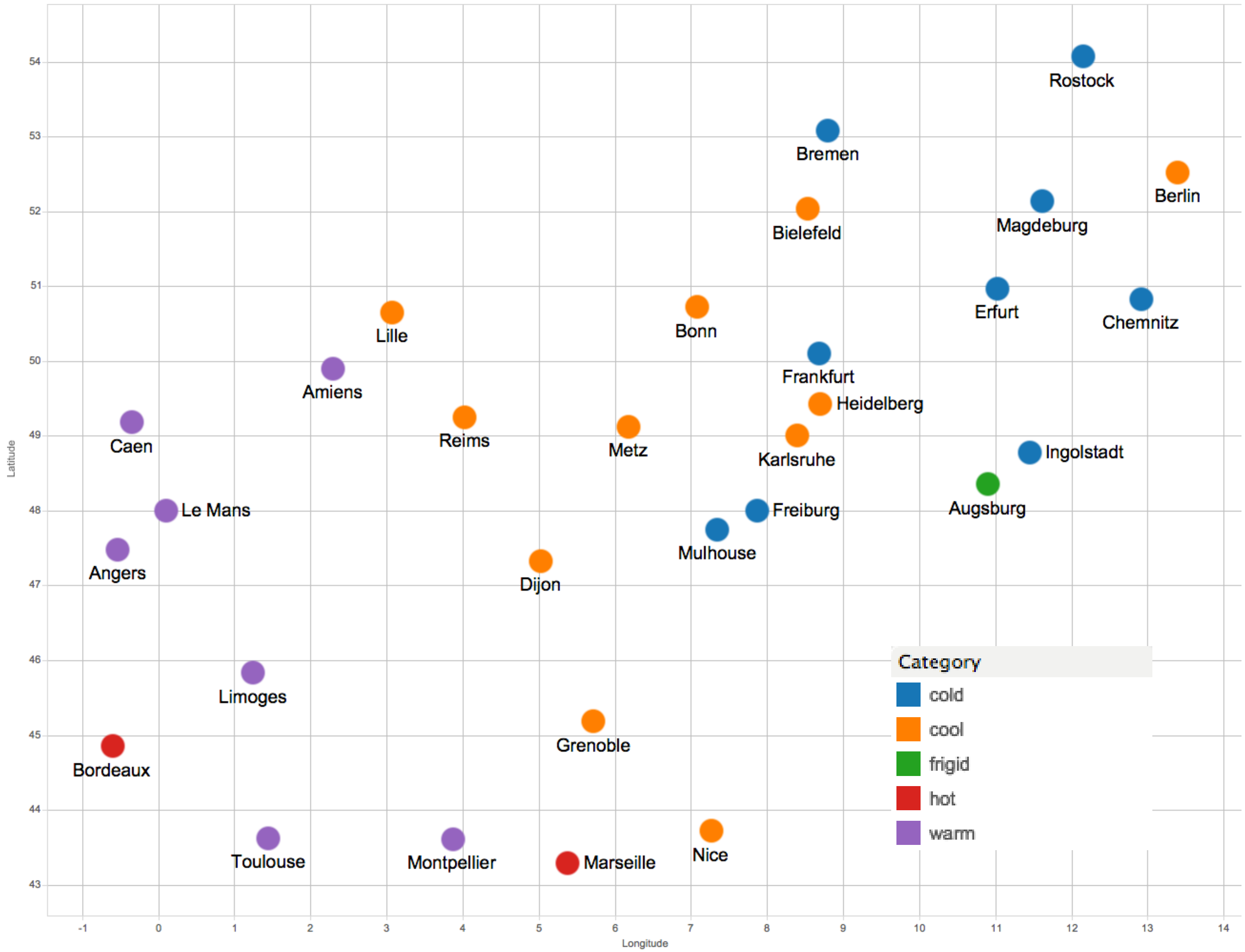
- City temperatures – France and Germany
- Features: longitude, latitude
- Labels: frigid, cold, cool, warm, hot

Nice (7.27, 43.72) cool
Toulouse (1.45, 43.62) warm
Frankfurt (8.68, 50.1) cold
……

Predict temperature category from longitude and latitude

Credit: Jennifer Widom

Example: Temperature Prediction



Credit: Jennifer Widom

Training set

Training set:

longitude, latitude **(x)**	**label (y)**
7.27, 43.72 (Nice)	cool
1.45, 43.62 (Toulouse)	warm
8.68, 50.1 (Frankfurt)	cold
…	…

Notation:

N = Number of training examples

x_{i} = “input” variable / features

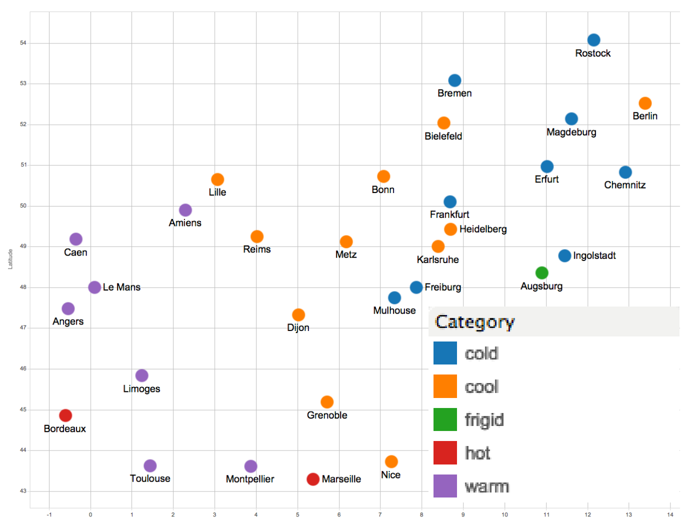
y_i = “output” variable / “target” variable

Supervised Learning

Predict: Is the city cold?

What should the learner be??

Want: input x → ? → output y

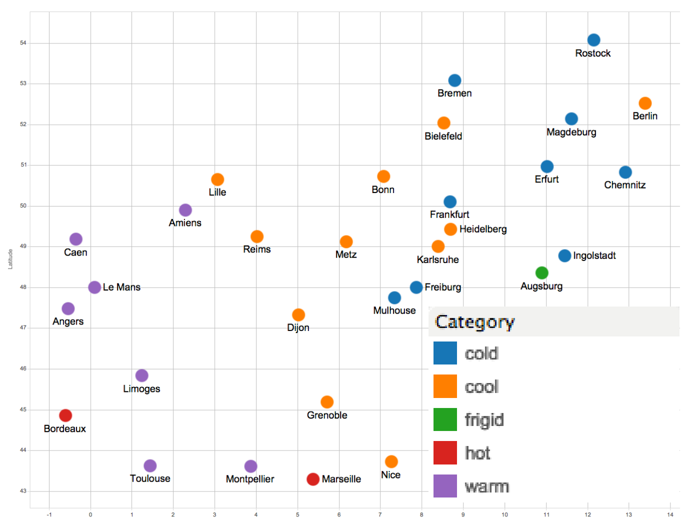


Hypothesis *h*

h : function parametrized by θ, e.g.,

$$h(x) = \text{sign}(\underbrace{\boxed{a}}_{\theta_{0,1}}x + \underbrace{\boxed{b}}_{\theta_2})$$

Want: input x $\longrightarrow$ $\boxed{h_{a,b}{}^{*}}$ $\longrightarrow$ output y

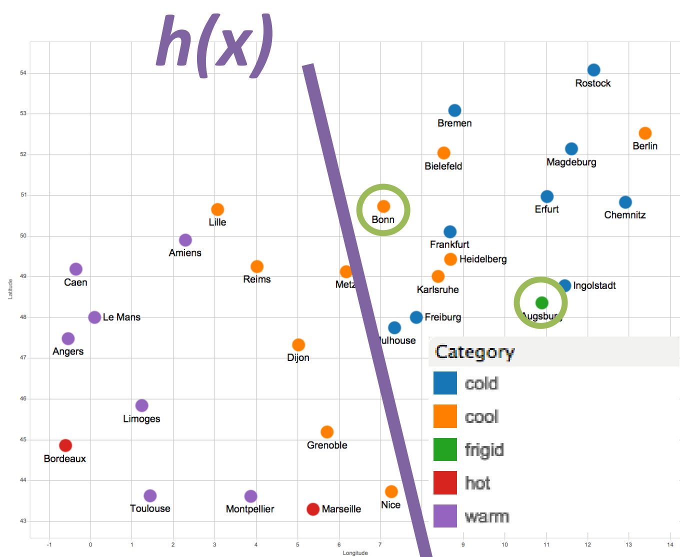


How to learn a,b?

But what if h(x_i) ≠ y_i ?

Given: Training Set $\{x_i, y_i\}$

Want: input x $\longrightarrow$ $\boxed{h_{a,b}{}^*}$ $\longrightarrow$ output y



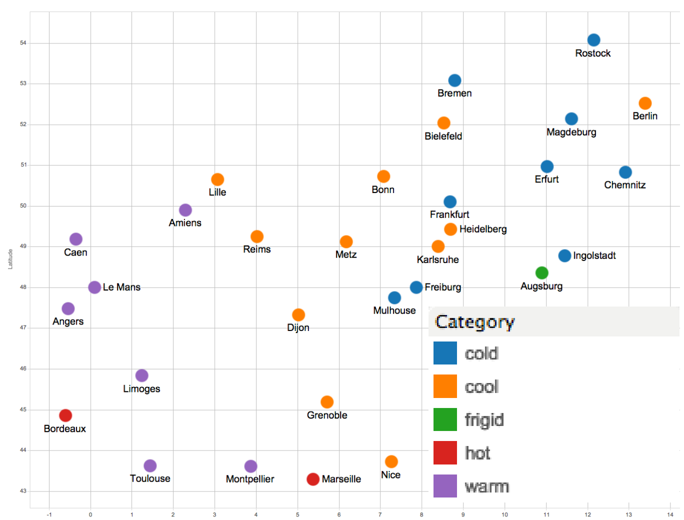
Cost function

Given: Training Set $\{x_i, y_i\}$

Cost/Error function $\text{Cost}(\text{h}(x_i), y_i)$

learning == minimizing cost

Want: input x $\longrightarrow$ $\boxed{h_{\text{a,b}}{}^{*}}$ $\longrightarrow$ output y



Supervised learning in one slide

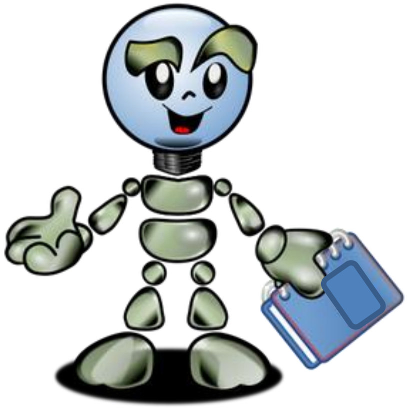
Given: Training Set $\{x_i, y_i\}$

Cost function $\mathrm{Cost}(\mathrm{h}(x_i), y_i)$

learning == minimizing cost

Learn a,b*: $\min_{a,b} \mathrm{Cost}(\, \mathrm{h}_{a,b}(x_i), y_i \,)$

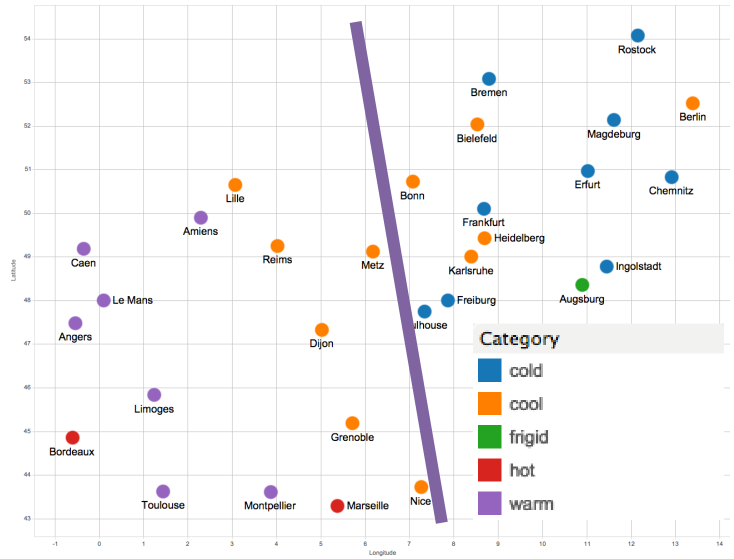
Result: input $x_i \longrightarrow \boxed{h_{a,b}^*} \longrightarrow$ output y_i



Learning to Classify

Error Rates

How do we know if h is good?



Linear hypothesis:

$$h_{a,b}(x) = \mathrm{sign}(ax + b)$$

a, b: Parameters

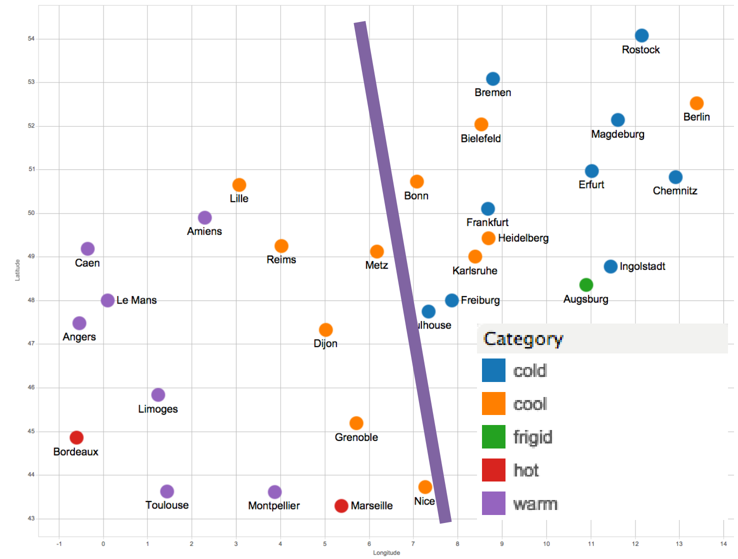
Error Function:

Portion of incorrect predictions

$$Error(h_{a,b}, D\{x,y\}) = \frac{1}{N}\sum_{i=1}^{N} h_{a,b}(x_i) \neq y_i$$

Goal: minimize $Error(h_{a,b}, D\{x,y\})$

What is a good baseline to compare to?



Current hypothesis:

$$h_{a,b}(x) = \text{sign}(ax + b)$$

a, b: Parameters

Random Baseline h_{rand} (a know nothing strategy):

Given a sample x_i, sample a label y_i uniformly at random from classes C

✅ $$Error(h_{a,b}, D\{x, y\}) > Error(h_{rand}, D\{x, y\})$$

slido

How might we improve our sampling strategy given our data?

ⓘ Start presenting to display the poll results on this slide.

Diving deeper into model analysis

Confusion Matrix Example (Table 1.1 in Forsyth)

Predict

True	0	1	2	3	4	Class error
0	151	7	2	3	1	7.9%
1	32	5	9	9	0	91%
2	10	9	7	9	1	81%
3	6	13	9	5	2	86%
4	2	3	2	6	0	100%

Diving deeper into model analysis

	GT Label "1"	GT Label "0"
Predicted "1"	True Positive (TP)	False Positive (FP)
Predicted "0"	False Negative (FN)	True Negative (TN)

Diving deeper into model analysis

Confusion Matrix Example (Table 1.1 in Forsyth)

$$Recall = \frac{TP}{TP + FN}$$

Predict

True		0	1	2	3	4	Class error
	0	151	7	2	3	1	7.9%
	1	32	5	9	9	0	91%
	2	10	9	7	9	1	81%
	3	6	13	9	5	2	86%
	4	2	3	2	6	0	100%

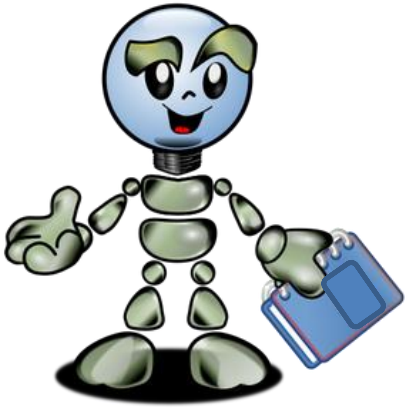
Diving deeper into model analysis

Confusion Matrix Example (Table 1.1 in Forsyth)

$$Precision = \frac{TP}{TP + FP}$$

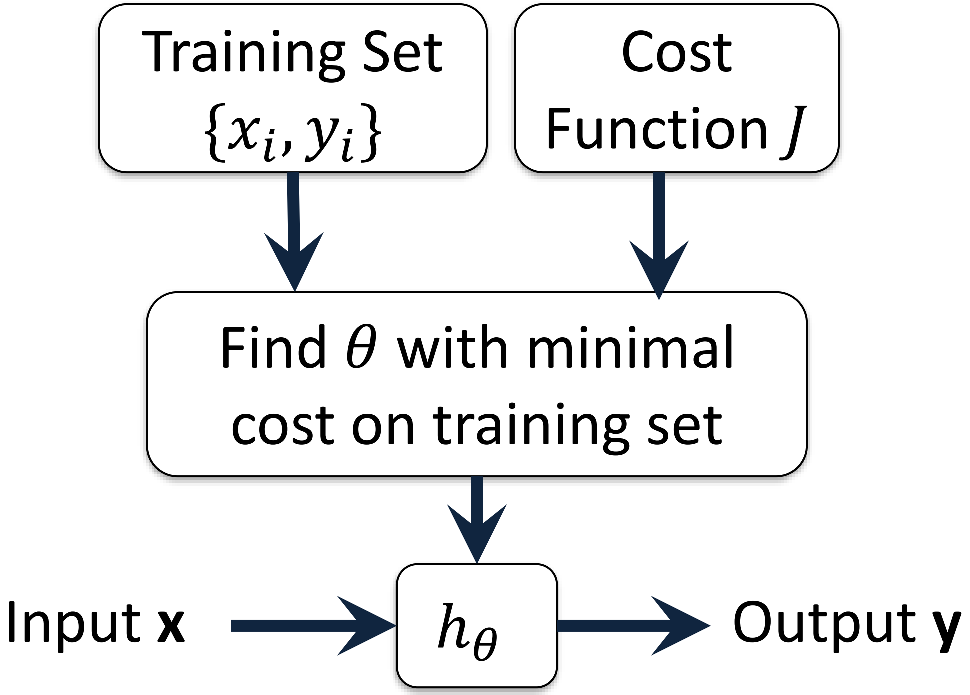
Predict

True		0	1	2	3	4	Class error
	0	151	7	2	3	1	7.9%
	1	32	5	9	9	0	91%
	2	10	9	7	9	1	81%
	3	6	13	9	5	2	86%
	4	2	3	2	6	0	100%

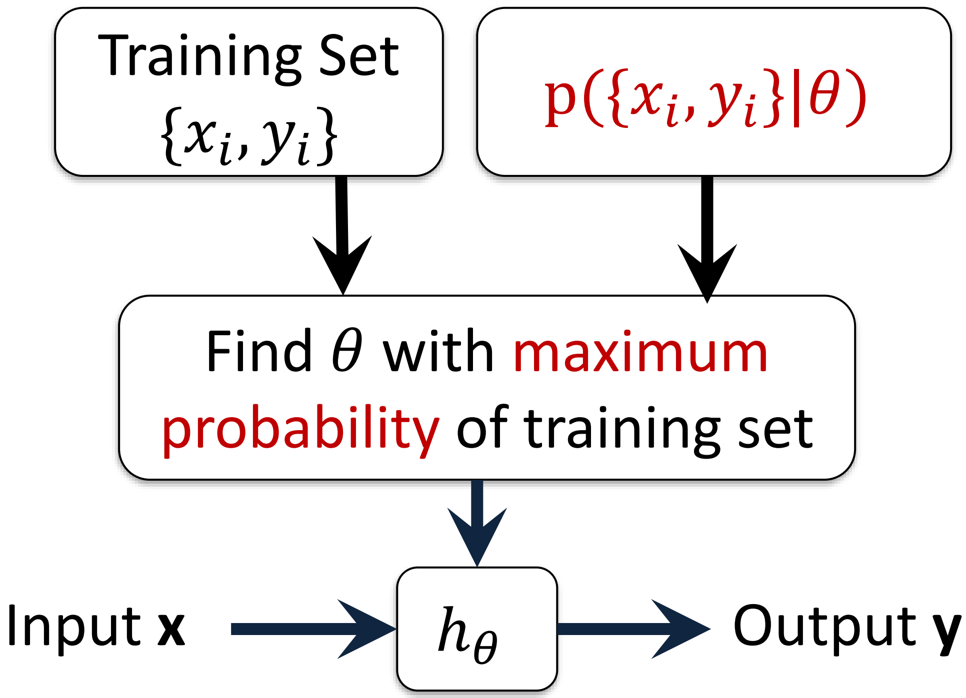


Maximum Likelihood Principle

Recall: Cost Function



Alternative View:
"Maximum Likelihood"



Maximum Likelihood: Example

- Intuitive example: Estimate a coin toss

  I have seen 3 flips of heads, 2 flips of tails, what is the chance of head (or tail) of my next flip?

- Model:

  Each flip is a Bernoulli random variable X

  X can take only two values: 1 (head), 0 (tail)

  $$p(X = 1) = \theta, \quad p(X = 0) = 1 - \theta$$

- θ is a parameter to be identified from data

Maximum Likelihood: Example

- 5 (independent) trials

$X_1 = 1$ $X_2 = 0$ $X_3 = 1$ $X_4 = 1$ $X_5 = 0$

- Likelihood of all 5 observations:

$$p(X_1, \ldots, X_5 | \theta) = \theta^3 (1 - \theta)^2$$

- Intuition

ML chooses θ such that likelihood is maximized

Maximum Likelihood: Example

- 5 (independent) trials

$X_1 = 1$ $X_2 = 0$ $X_3 = 1$ $X_4 = 1$ $X_5 = 0$

- Likelihood of all 5 observations:

$$p(X_1, \ldots, X_5 | \theta) = \theta^3 (1 - \theta)^2$$

- Solution (left as exercise)

$$\theta_{ML} = \frac{3}{(3 + 2)}$$

i.e. fraction of heads in total number of trials

Maximum likelihood way of estimating model parameters θ

In general, assume data is generated by some distribution

$$U \sim p(U|\theta)$$

Observations (i.i.d.)

$$\mathrm{D} = \{u^{(1)}, u^{(2)}, \dots, u^{(m)}\}$$

Maximum likelihood estimate

Likelihood

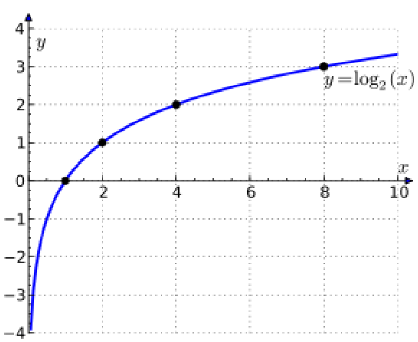
$$\mathcal{L}(\mathrm{D}) = \boxed{\prod_{i=1}^{m} p(u^{(i)}|\theta)}$$

Log likelihood

$$\theta_{ML} = \underset{\theta}{\operatorname{argmax}}\, \mathcal{L}(\mathrm{D})$$

$$= \underset{\theta}{\operatorname{argmax}} \boxed{\sum_{i=1}^{m} \log p(u^{(i)}|\theta)}$$

Note: *p* replaces *h*, and max replaces min



log(f(x)) is monotonic/increasing, same argmax as *f(x)*

i.i.d. observations

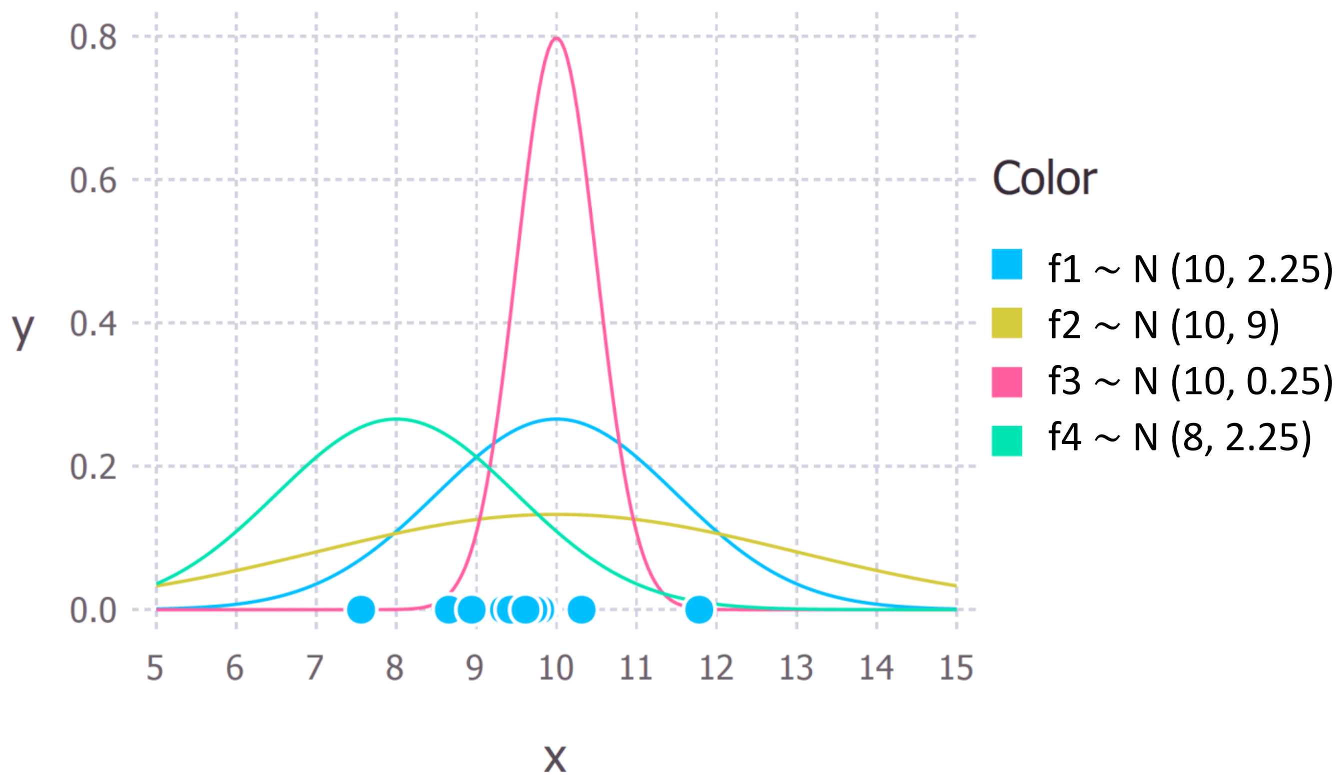
- independently identically distributed random variables
- If u^i are i.i.d. r.v.s, then

$$p(u^1, u^2, \dots, u^m) = p(u^1)p(u^2) \dots p(u^m)$$

- A reasonable assumption about many datasets, but not always

ML: Another example

- Observe a dataset of points $D = \{x_i\}_{i=1:10}$
- Assume x is generated by Normal distribution, $x \sim N(x|\mu, \sigma)$
- Find parameters $\theta_{ML} = [\mu, \sigma]$ that maximize $\prod_{i=1}^{10} N(x_i|\mu, \sigma)$



slido

What model best fits the data?

ⓘ Start presenting to display the poll results on this slide.

Next Class

Finish with Bayesian methods +

Support Vector Machines I:

Maximum margin methods, support vector machines, hinge loss, regularization

Reading: Forsyth Ch 2.1-2.1.2