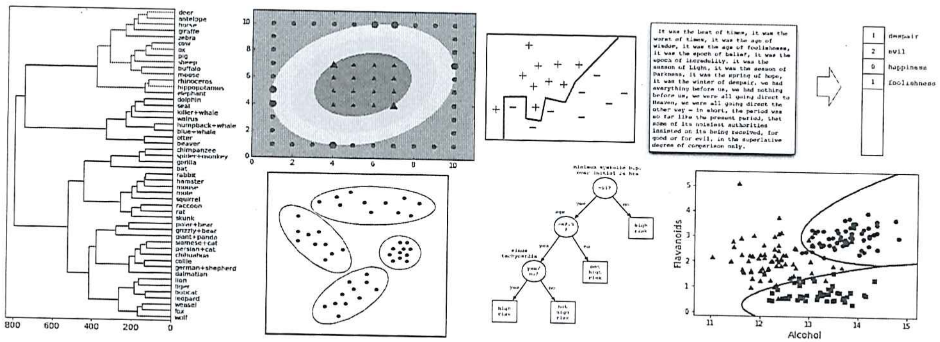
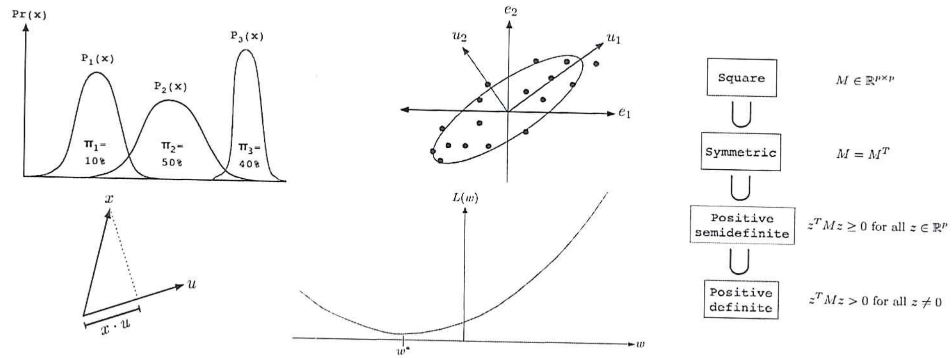
WEEK 1-1 12/June/2018.

# DSE 220X: Fundamentals of Machine Learning




| | | |
|---|---|---|
| Nearest neighbor | Kernel methods | Hierarchical clustering |
| Generative models | Decision trees | Principal component analysis |
| Least-squares regression | Boosting and bagging | Singular value decomposition |
| Ridge regression, Lasso | Random forests | Autoencoders |
| Logistic regression | $k$-means | Deep learning |
| Support vector machines | Mixtures of Gaussians | |




| | | |
|---|---|---|
| Probability | Linear algebra | Optimization |
| Probability spaces | Matrices and vectors | Gradient descent |
| Bayes' rule | Projections | Stochastic gradient descent |
| Random variables | Positive definiteness | Convex optimization |
| Mean and variance | Eigendecompositions | Duality |
| Measuring dependence | Spectral decomposition | |

## Skills you will acquire

❶ Familiarity with most widely-used ML methods
- How they work
- The kinds of data they are suited to
- Their strengths and weaknesses

❷ Adapting existing methods to a particular application

❸ The foundational knowledge to keep pace with a fast-moving field jupyter kernel-perceptron Last Checkpoint: a minute ago (autosaved)

File Edit View Insert Cell Kernel Navigate Widgets Help Trusted

In [5]:
```
# RBF kernel function
def rbfk(x,z,sigma):
    tmp = np.linalg.norm(x-z)/sigma
    return np.exp(-tmp*tmp)
# Evaluate a classifier (alpha,x,y) on a new point x0
def evaluate(alpha,x,y,x0):
    value = 0.0
    for i in range(len(alpha)):
        if alpha[i] > 0:
            value = value + alpha[i] * y[i] * rbfk(x[i,:],x0,1.0)
    return np.sign(value)
# Now train an RBF kernel perceptron
working = True
alpha = np.zeros(n)
np.random.seed(None)
while working:
    working = False
    I = np.random.permutation(n)
    for i in range(n):
        j = I[i]
        if (evaluate(alpha,x,y,x[j,:]) != y[j]):
            working = True
            alpha[j] = alpha[j] + 1
```

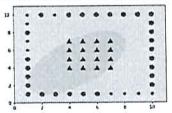
In [3]:
```
delta = 0.05
xx, yy = np.meshgrid(np.arange(xmin, xmax, delta), np.arange(ymin, ymax, delta))
grid = np.c_[xx.ravel(), yy.ravel()]
gn, gp = grid.shape
Z = np.zeros(gn)
for i in range(gn):
    Z[i] = evaluate(alpha,x,y,grid[i,:])

# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.pcolormesh(xx, yy, Z, cmap=plt.cm.PRGn, vmin=-3, vmax=3)

# Identify the support vectors
sv = (alpha > 0)
notsv = np.logical_not(sv)

# Plot also the training points
plt.plot(x[(y==1)*notsv,0], x[(y==1)*notsv,1], 'ro')
plt.plot(x[(y==1)*sv,0], x[(y==1)*sv,1], 'ro', markersize=10)
plt.plot(x[(y==-1)*notsv,0], x[(y==-1)*notsv,1], 'k^')
plt.plot(x[(y==-1)*sv,0], x[(y==-1)*sv,1], 'k^', markersize=10)

plt.xlim(xmin, xmax)
plt.ylim(ymin, ymax)
plt.show()
```



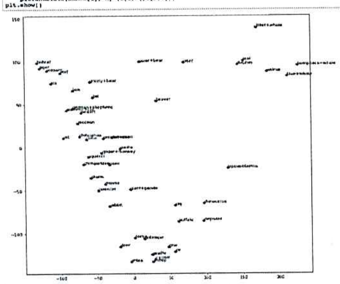
jupyter cluster-aws Last Checkpoint: 32 minutes ago (autosaved)

File Edit View Insert Cell Kernel Navigate Widgets Help

```
['ox', 'buffalo', 'pig', 'cow']
['skunk', 'rat', 'weasel', 'raccoon']
['killer+whale', 'blue+whale', 'humpback+whale', 'seal', 'otter', 'walrus', 'dolphin']
['spider+monkey', 'gorilla', 'chimpanzee']
['dalmatian', 'persian+cat', 'german+shepherd', 'collie']
['siamese+cat', 'chihuahua']
```

In [4]:
```
# Now visualize using PCA
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
z = pca.fit_transform(x)
```

In [6]:
```
import matplotlib.pyplot as plt
from pylab import rcParams
rcParams['figure.figsize'] = 12, 12
plt.plot(z[:,0], z[:,1], 'ro')
for i in range(50):
    plt.annotate(names[i], xy=(z[i,0],z[i,1]))
plt.show()
```



WEEK 1-2. 12/June/2018.

# Nearest neighbor classification

## Topics we'll cover

1. What is a classification problem?
2. The training set and test set
3. Representing data as vectors
4. Distance in Euclidean space
5. The 1-NN classifier
6. Training error versus test error
7. The error of a random classifier

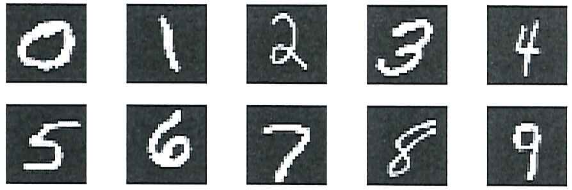
## The problem we'll solve today

Given an image of a handwritten digit, say which digit it is.

$\Longrightarrow$ 3

The number of loops / straight line & their positions to decide what digit it is.

Some more examples:



Problems. Handwriting are super noisy / a lot of variability in the way people write 4, 7, 9 etc.

## The machine learning approach

Assemble a data set:

i machine needs a huge amount of data (MNIST data set handwritten image + correct digit).

An entirely different approach. - ML. Rather than trying to figure out the underlying patterns ourselves, we'll just let the machine do it for us.

The MNIST data set of handwritten digits:

- ii **Training set** of 60,000 images and their labels. to learn a classifier.
- iii **Test set** of 10,000 images and their labels. ~~to test~~ errors.

**And let the machine figure out the underlying patterns.**

a function that takes an image and then outputs what digit is.

## Nearest neighbor classification

Training images $x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(60000)}$
Labels $y^{(1)}, y^{(2)}, y^{(3)}, \ldots, y^{(60000)}$ are numbers in the range $0 - 9$

Input. New Image x.

How to **classify** a new image $x$?

- Find its nearest neighbor amongst the $x^{(i)}$
- Output: Return $y^{(i)}$

Questions: Distance (looking for the one that's closest to x, so we have some notion of distance between images).

## The data space

So How to measure the distance between images?

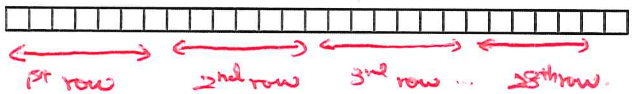
i. represent image as vector.

MNIST images:

- Size $28 \times 28$ (total: 784 pixels)
- Each pixel is grayscale: 0-255

Black white.

ii. Stretch each image into a vector with 784 coordinates: Dimensional Vector.

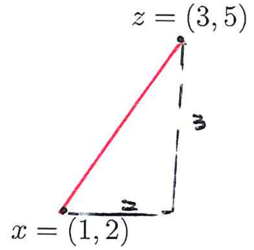



iii. 
- Data space $\mathcal{X} = \mathbb{R}^{784}$ Euclidean space.
- Label space $\mathcal{Y} = \{0, 1, \ldots, 9\}$

Questions: How to compute the distance between vectors?

## The distance function (the most common / default).

Remember Euclidean distance in two dimensions?




distance (x, z) = length of the line.

$= \sqrt{2^2+3^2}$

$= \sqrt{13}$.

Euclidean Distance in 2 Dimensions.

Compute the distance between 2 vectors x, z; simply find out how much differ on each individual coordinates.

## Euclidean distance in higher dimension

Euclidean distance between 784-dimensional vectors $x, z$ is

$$\|x - z\| = \sqrt{\sum_{i=1}^{784} (x_i - z_i)^2}$$

Here $x_i$ is the $i$th coordinate of $x$.

## Nearest neighbor classification

Training images $x^{(1)}, \ldots, x^{(60000)}$, labels $y^{(1)}, \ldots, y^{(60000)}$

To classify a new image $x$:

- Find its nearest neighbor amongst the $x^{(i)}$ **using Euclidean distance in $\mathbb{R}^{784}$**
- Return $y^{(i)}$

How accurate is this classifier?

## Accuracy of nearest neighbor on MNIST

Training set of 60,000 points.

- What is the error rate on training points? **Zero.**
  In general, training error is an overly optimistic predictor of future performance.
- A better gauge: separate test set of 10,000 points.
  Test error = fraction of test points incorrectly classified.
- What test error would we expect for a *random classifier*? (One that picks a label 0 − 9 at random?) **90%.**
- Test error of nearest neighbor: **3.09%.** - Out of 10,000 points,
  it gets 309 of them wrong $\frac{309}{10{,}000} = 3.09\%$

## Examples of errors

Test set of 10,000 points:

- 309 are misclassified
- Error rate 3.09%

Examples of errors:

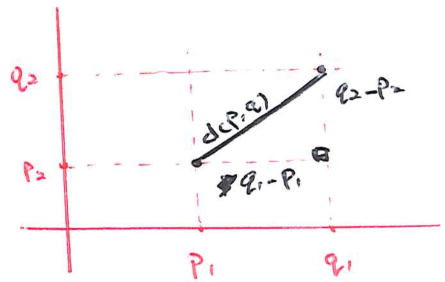
Query

NN

More. 2-dimensional Euclidean Distance.

if $p = (p_1, p_2)$ and $q = (q_1, q_2)$ the the distance is

$$d(p,q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}$$




$$d(p,q)^2 = (q_1 - p_1)^2 + (q_2 - p_2)^2$$

Pythagorean theorm.

WEEK 1-3. 12/June/2018.

## Improving the performance of nearest neighbor

## Recall: nearest neighbor on MNIST

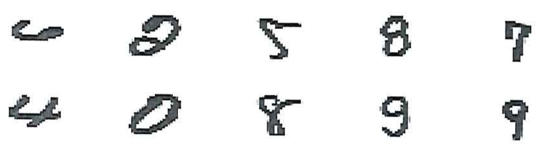
- Images of handwritten digits, represented as vectors in $\mathbb{R}^{784}$.
- Labels $0-9$
- Training set: 60,000 points; test set: 10,000 points

Test error of nearest neighbor using Euclidean distance: 3.09%

Examples of errors:

Query

NN



Ideas for improvement: (1) $k$-NN (2) better distance function. To improve the performance.

## $K$-nearest neighbor classification

To classify a new point:

i • Find the $k$ nearest neighbors in the training set.

ii • Return the most common label amongst them.

MNIST:

| $k$ | 1 | 3 | 5 | 7 | 9 | 11 |
|---|---|---|---|---|---|---|
| Test error (%) | 3.09 | 2.94 | 3.13 | 3.10 | 3.43 | 3.34 |

Error Rate. stop goes down.

In real life, there's no test set. How to decide which $k$ is best?

finding '3' closest images in the train set and return the majority label.

## Cross-validation - to select the value of k.

How to estimate the error of $k$-NN for a particular $k$?

**10-fold cross-validation** k-fold Cross Validation.

Input i • Divide the training set into 10 equal pieces. Randomly divide the data into k equal sized 'folds'.
Training set (call it $S$): 60,000 points
Call the pieces $S_1, S_2, \ldots, S_{10}$: 6,000 points each.

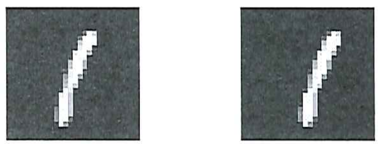
Input ii • For each piece $S_i$: Drop a single fold and fit the classifier to the remaining k-1 folds.
- Classify each point in $S_i$ using $k$-NN with training set $S - S_i$
- Let $\epsilon_i$ = fraction of $S_i$ that is incorrectly classified

Output iii • Take the average of these 10 numbers: Report the classification error as the average of the misclassification rates.

$$\text{estimated error with } k\text{-NN} = \frac{\epsilon_1 + \cdots + \epsilon_{10}}{10}$$

## Another improvement: better distance functions

The Euclidean ($\ell_2$) distance between these two images is very high!



Much better idea: distance measures that are invariant under:

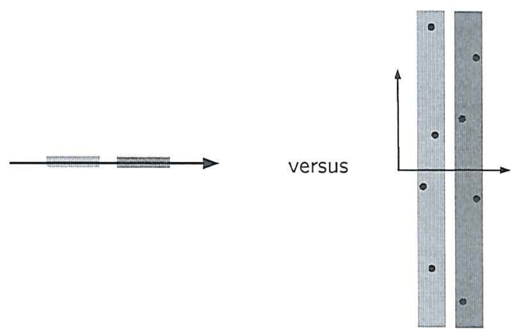
- Small translations and rotations. e.g. *tangent distance.*
- A broader family of natural deformations. e.g. *shape context.*

Test error rates:

| $\ell_2$ | tangent distance | shape context |
|---|---|---|
| 3.09 | 1.10 | 0.63 |

## Related problem: feature selection Prior - using KNN/NN.

Feature selection/reweighting is part of picking a distance function.
And, one noisy feature can wreak havoc with nearest neighbor!




## Algorithmic issue: speeding up NN search

Naive search takes time $O(n)$ for training set of size $n$: slow!

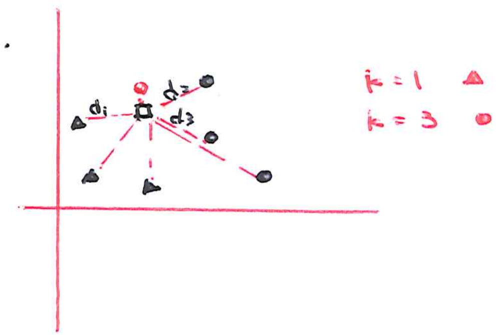
Luckily there are data structures for speeding up nearest neighbor search, like:

1. Locality sensitive hashing
2. Ball trees
3. $K$-d trees

These are part of standard Python libraries for NN, and help a lot.

Extra. KNN classification algorithm.

- given.
  - training examples $\{x_i, y_i\}$
  - New point $x$ that we want to classify
- Algorithm.
  - Compute distance $D(x, x_i)$ to every training example $x_i$
  - select $k$ closest instances $x_{i1}, \ldots x_{ik}$ and their labels $y_{i1}, \ldots, y_{ik}$
  - Output the class $y^*$ which is most frequent in $y_{i1} \ldots y_{ik}$.



WEEK 1-4 12/June/2018.

## Useful distance functions for machine learning

### Topics we'll cover

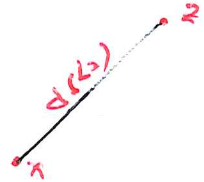
2 families of distance function.

1. $L_p$ norms
2. Metric spaces

### Measuring distance in $\mathbb{R}^m$

Usual choice: **Euclidean distance:** L2

$$\|x - z\|_2 = \sqrt{\sum_{i=1}^{m} (x_i - z_i)^2}.$$




For $p \geq 1$, here is $\ell_p$ **distance**:

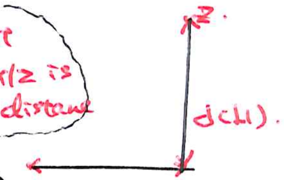
$$\|x - z\|_p = \left( \sum_{i=1}^{m} |x_i - z_i|^p \right)^{1/p}$$

- $p = 2$: Euclidean distance
- $\ell_1$ distance: $\|x - z\|_1 = \sum_{i=1}^{m} |x_i - z_i|$
- $\ell_\infty$ distance: $\|x - z\|_\infty = \max_i |x_i - z_i|$

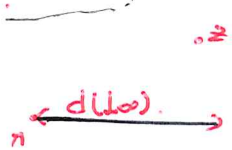
Greatest between x/z is which distance along

L infinite distance: look at 5th coordinate

Look at the difference along each coordinates, and simply add up those differences.




### Example 1




Consider the all-ones vector $(1, 1, \ldots, 1)$ in $\mathbb{R}^d$.
What are its $\ell_2$, $\ell_1$, and $\ell_\infty$ length?

- D-dimensional space.
- $x = (1, 1, \cdots, 1)$
- the length of vector is its distance from the origin, from the all 0 points

$x = (1, 1, 1 \cdots 1)$

$L_2$. $\|x\|_2 = \sqrt{1^2 + 1^2 + \cdots 1^2}$
$= \sqrt{d}$

$L_\infty$ $\|x\|_\infty = 1$.

$L_1$. $\|x\|_1 = |x_1| + \cdots |x_d|$
$= d$.

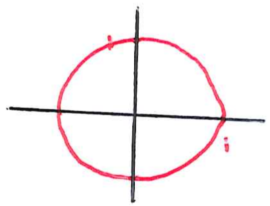
## Example 2

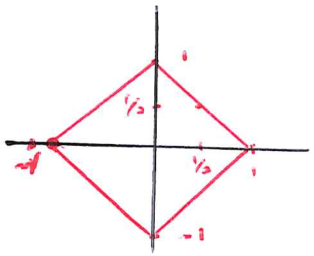
In $\mathbb{R}^2$, draw all points with:

1. $\ell_2$ length 1
2. $\ell_1$ length 1
3. $\ell_\infty$ length 1

- 2-Dimensional plan
- want to draw all points whose $l_2 / l_1 / l_\infty$ length is 1.

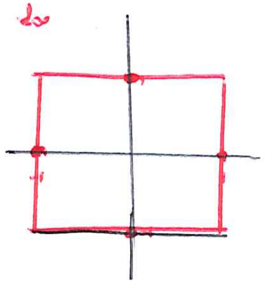
$l_2$. $\{(x_1, x_2) : \sqrt{x_1^2 + x_2^2} = 1\}$





$l_1$ : $\{(x_1, x_2) : |x_1| + |x_2| = 1\}$

$l_\infty$



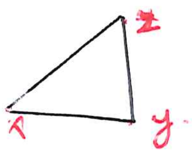
## Metric spaces

Other distance function needed. e.g. distance between objects (string/graph)

Let $\mathcal{X}$ be the space in which data lie.

A distance function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a **metric** if it satisfies these properties:

- $d(x, y) \geq 0$ (nonnegativity)
- $d(x, y) = 0$ if and only if $x = y$
- $d(x, y) = d(y, x)$ (symmetry)
- $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality)




## Example 1

$\mathcal{X} = \mathbb{R}^m$ and $d(x, y) = \|x - y\|_p$

$l_1$ in Metric Space.

$d(x, y) = \sum_{i=1}^{m} |x_i - y_i|$

Check:

- $d(x, y) \geq 0$ (nonnegativity) ✓
- $d(x, y) = 0$ if and only if $x = y$ ✓
- $d(x, y) = d(y, x)$ (symmetry) ✓
- $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality) ✓

$l_1$ distance satisfies all the properties of a metric, and all the $\ell_p$ distances do.

## Example 2

$\mathcal{X} = \{\text{strings over some alphabet}\}$ and $d = \text{edit distance}$

- Input space doesn't consist of vectors.

Check:

- $d(x, y) \geq 0$ (nonnegativity) ✓
- $d(x, y) = 0$ if and only if $x = y$ ✓
- $d(x, y) = d(y, x)$ (symmetry) ✓
- $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality) ✓

- $\mathcal{X} = \{A, C, G, T\}^*$ strings over alphabet. Input.
  $^*$ means the strings with arbitrary length.
- $x$ = A C C G T. 2 strings we have.
  $y$ = C C G T. distance between the 2 strings.

$d(x, y)$ = # of insertions, deletions and substitutions from $x$ to $y$ to be a metric.

## A non-metric distance function

Let $p, q$ be probability distributions on some set $\mathcal{X}$.

The **Kullback-Leibler divergence** or **relative entropy** between $p, q$ is:

$$d(p,q) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)}.$$

$\Downarrow$

$p = (1/2, 1/4, 1/8, 1/8)$

$q = (1/6, 1/3, 1/3, 1/6).$

$$d(p,q) = \frac{1}{2}\log\frac{1/2}{1/6} + \frac{1}{4}\log\frac{1/4}{1/3} + 1/8 \log\frac{1/8}{1/3} + \frac{1}{8}\log\frac{1/8}{1/6}$$

week1-5. 12|Jun|2018

# A host of prediction problems

## Topics we'll cover

1. Machine learning versus algorithms
2. A taxonomy of prediction problems
3. Roadmap for the course

## Machine learning versus Algorithms

A central goal of both fields:

*develop procedures that exhibit a desired input-output behavior.*

- **Algorithms**: input-output mapping can be precisely defined.
  Input: Graph $G$, two nodes $u, v$ in the graph.
  Output: Shortest path from $u$ to $v$ in $G$
- **Machine learning**: mapping cannot easily be made precise.
  Input: Picture of an animal.
  Output: Name of the animal.

Instead, provide examples of (input,output) pairs.
Ask the machine to *learn* a suitable mapping itself.

## Prediction problems: inputs and outputs

Basic terminology:

- The input space, $\mathcal{X}$.
  E.g. $32 \times 32$ RGB images of animals.
- The output space, $\mathcal{Y}$.
  E.g. Names of 100 animals.

$x$: 

$y$: "bear"

After seeing a bunch of examples $(x, y)$, pick a mapping

Input Input

$$f : \mathcal{X} \to \mathcal{Y}$$

that accurately recovers the input-output pattern of the examples.

Categorize prediction problems by the type of **output space**:
(1) discrete, (2) continuous, or (3) probability values

## Discrete output space: classification

**Binary classification**

E.g., Spam detection
$\mathcal{X} = \{\text{email messages}\}$
$\mathcal{Y} = \{\text{spam, not spam}\}$

**Multiclass**

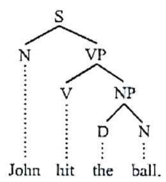
E.g., News article classification
$\mathcal{X} = \{\text{news articles}\}$
$\mathcal{Y} = \{\text{politics, business, sports}, \ldots\}$

**Structured outputs**

E.g., Parsing
$\mathcal{X} = \{\text{sentences}\}$
$\mathcal{Y} = \{\text{parse trees}\}$

$x =$ "John hit the ball"

$y =$




## Continuous output space: regression

- Pollution level prediction
  Predict tomorrow's air quality index in my neighborhood
  $\mathcal{Y} = [0, \infty)$ $\quad$ ($< 100$: okay, $> 200$: dangerous)
- Insurance company calculations
  What is the expected life expectancy of this person?
  $\mathcal{Y} = [0, 120]$

What are suitable predictor variables ($\mathcal{X}$) in each case?

## Probability estimation

$\mathcal{Y} = [0, 1]$ represents **probabilities**

Example: Credit card transactions

- $x$ = details of a transaction
- $y$ = probability this transaction is fraudulent

Why not just treat this as a binary classification problem?

## Roadmap for the course

1. Solving prediction problems
   Classification, regression, probability estimation
2. Representation learning
   Clustering, projection, dictionary learning, autoencoders
3. Deep learning