Machine Learning

Nando de Freitas

January 11, 2005

Lecture 1 - *Introduction*

OBJECTIVE: Introduce machine learning, relate it to other fields, identify its applications, and outline the various types of learning.

◇ MACHINE LEARNING

Machine Learning is the processes of deriving abstractions of the real world from a set of observations. The resulting abstractions (models) are useful for

1. Making decisions under uncertainty.
2. Predicting future events.
3. Classifying massive quantities of data quickly.
4. Finding patterns (clusters, hierarchies, abnormalities, associations) in the data.
5. Developing autonomous agents (robots, game agents and other programs).

◇ MACHINE LEARNING AND OTHER FIELDS

Machine learning is closely related to many disciplines of human endeavor. For example:

Information Theory :

- Compression: Models are compressed versions of the real world.
- Complexity: Suppose we want to transmit a message over a communication channel

$$Sender \xrightarrow{data} Channel \xrightarrow{data} Receiver$$

To gain more efficiency, we can compress the data and send both the compressed data and the model to decompress the data.

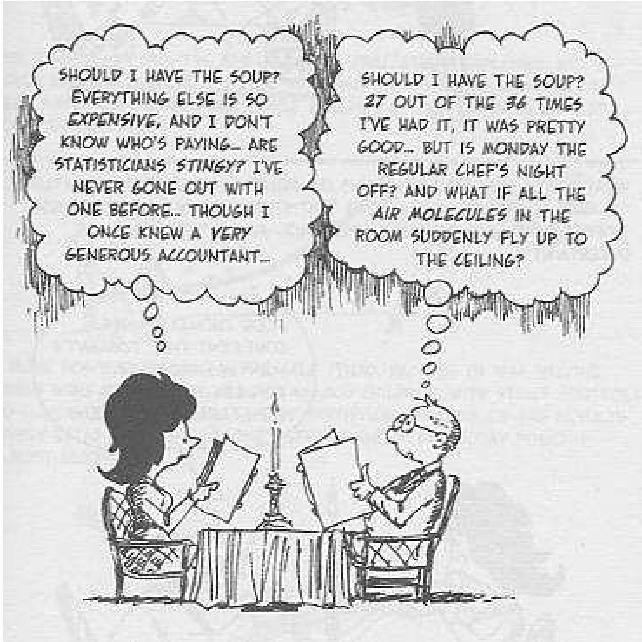
$$Sender \xrightarrow{data} Encoder \underset{model}{\overset{comp.\ data}{\longrightarrow}} Channel \underset{model}{\overset{comp.\ data}{\longrightarrow}} Decoder \xrightarrow{data} Receiver$$

There is a fundamental tradeoff between the amount of compression and the cost of transmitting the model. More complex models allow for more compression, but are expensive to transmit. Learners that balance these two costs tend to perform better in the

future. That is, they *generalise* well.

Probability Theory :

- Modelling noise.
- Dealing with uncertainty: occlusion, missing data, synonymy and polisemy, unknown inputs.



Statistics :

- *Data Analysis and Visualisation*: gathering, display and summary of data.
- *Inference*: drawing statistical conclusions from specific data.

Computer Science :

- Theory.
- Database technology.
- Software engineering.
- Hardware.

Optimisation : Searching for optimal *parameters* and models in constrained and unconstrained settings is ubiquitous in machine learning.

Philosophy : The study of the nature of knowledge (epistemology) is central to machine learning. Understanding the learning process and the resulting abstractions is a

question of fundamental importance to human beings. At the onset of Western philosophy, Plato and Aristotle distinguished between "essential" and "accidental" properties of things. The Zen patriarch, Bodhidharma also tried to get to the essence of things by asking "what is that?" in a serious sense, of course.

Other Branches of Science :

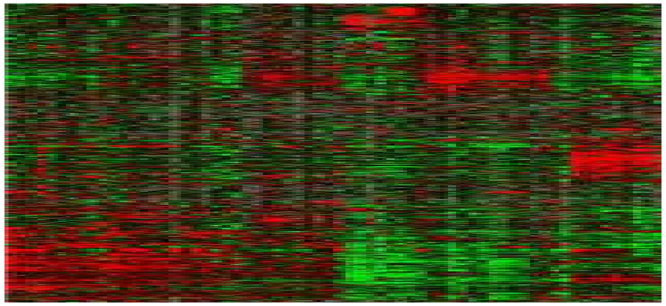
- Game theory.
- Econometrics.
- Cognitive science.
- Engineering.
- Psychology.
- Biology.

◇ APPLICATION AREAS

Machine learning and data mining play an important role in the following fields:

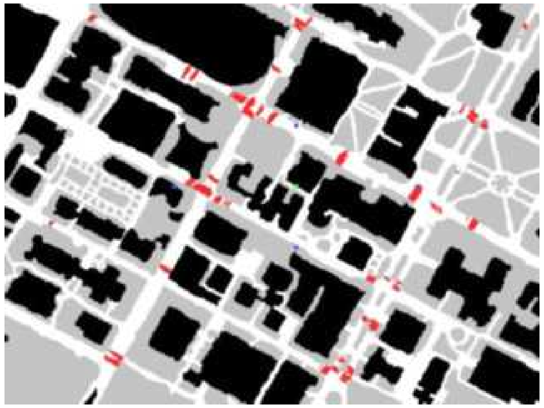
Software : Teaching the computer instead of programming it.

Bioinformatics : Sequence alignment, DNA micro-arrays, drug design, novelty detection.

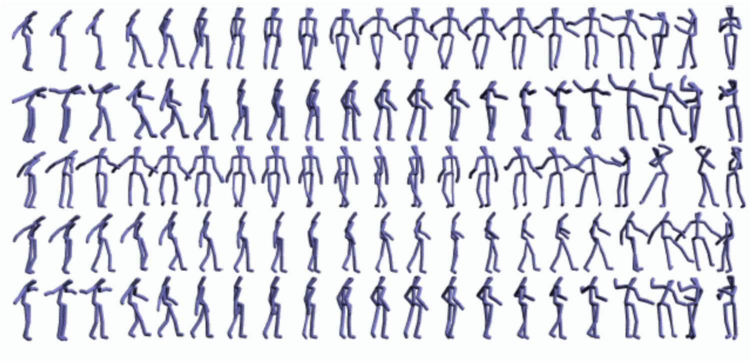


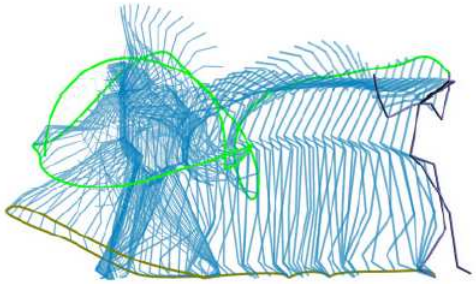
Computer Vision : Handwritten digit recognition (Le Cun), tracking, segmentation, object recognition.

Robotics : State estimation, control, localisation and map building.



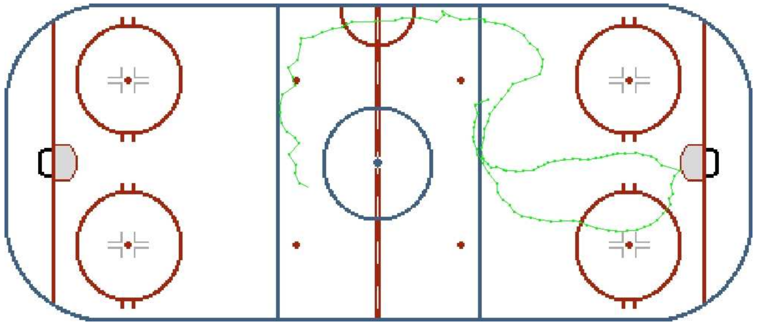
Computer Graphics : Automatic motion generation, realistic simulation. E.g., style machines by Brand and Hertzmann:





Electronic Commerce : Data mining, collaborative filtering, recommender systems, spam.

Computer Games : Intelligent agents and realistic games.



Financial Analysis : Options and derivatives, forex, portfolio allocation.

Medical Sciences : Epidemiology, diagnosis, prognosis, drug design.

Speech : Recognition, speaker identification.

Multimedia : Sound, video, text and image databases; multimedia translation, browsing, information retrieval (search engines).

◇ TYPES OF LEARNING

Supervised Learning

We are given input-output *training* data $\{x_{1:N}, y_{1:N}\}$, where $x_{1:N} \triangleq (x_1, x_2, \ldots, x_N)$. That is, we have a teacher that tell us the outcome y for each input x. Learning involves adapting the model so that its predictions $\widehat{y}$ are close to y. To achieve this we need to introduce a $loss function$ that tells us how close $\widehat{y}$ is to y. Where does the loss function come from?

$$x \longrightarrow Model \longrightarrow \widehat{y}$$

After learning the model, we can apply it to novel inputs and study its response. If the predictions are accurate we have reason to believe the model is correct. We can exploit this during training by splitting the dataset into a training set and a test set. We learn the model with the training set and validate it with the test set. This is an example of a model selection technique knowns as *cross-validation*.

What are the advantages and disadvantages of this technique?

In the literature, inputs are also known as predictors, explanatory variables or covariates, while outputs are often referred to as responses or variates.

Unsupervised Learning

Here, there is not teacher. The learner must identify structures and patterns in the data. Many times, there is no single correct answer. Examples of this include image segmentation and data clustering.

Semi-supervised Learning

It's a mix of supervised and unsupervised learning.

Reinforcement Learning

Here, the learner is given a reward for an action performed in a particular environment. Human cognitive tasks as well as "simple" motor tasks like balancing while walking seem to make use of this learning paradigm. RL, therefore, is likely to play an important role in graphics and computer games in the future.

Active Learning

$$World \xrightarrow{data} Passive\ Learner \longrightarrow Model$$

$$World \underset{query}{\overset{data}{\rightleftarrows}} Active\ Learner \longrightarrow Model$$

Active learners query the environment. Queries include questions and requests to carry out experiments. As an analogy, I like to think of good students as active learners! But, how do we select queries optimally? That is, what questions should we ask? What is the price of asking a question?

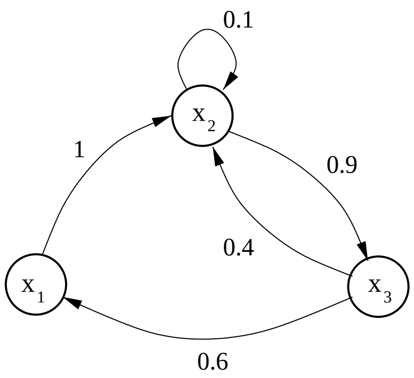
Active learning plays an important role when establishing *causal* relationships.

Lecture 2 - *Google's PageRank: Why math helps*

OBJECTIVE: Motivate linear algebra and probability as important and necessary tools for understanding large datasets. We also describe the algorithm at the core of the Google search engine.

◇ PAGERANK

Consider the following mini-web of 3 pages (the data):



The nodes are the webpages and the arrows are links. The numbers are the normalised number of links. We can re-write this directed graph as a **transition matrix**:

⋆

T is a **stochastic matrix**: its columns add up to 1, so that

$$T_{i,j} = P(x_j|x_i)$$

$$\sum_j T_{i,j} = 1$$

In information retrieval, we want to know the "relevance" of each webpage. That is, we want to compute the probability of each webpage: $p(x_i)$ for $i = 1, 2, 3$.

Let's start with a random guess $\pi = (0.5, 0.2, 0.3)^T$ and "crawl the web" (multiply by T several times). After, say

$N = 100$, iterations we get:

$$\pi^T T^N = (0.2, 0.4, 0.4)$$

We soon notice that no matter what initial π we choose, we always converge to $p = (0.2, 0.4, 0.4)^T$. So

⋆

$$p^T T =$$

⋆

The distribution p is a measure of the relevance of each page. Google uses this. But will this work always? When does it fail?

$\star$

The **Perron-Frobenius Theorem** tell us that for any starting point, the chain will converge to the invariant distribution p, as long as T is a stochastic transition matrix that obeys the following properties:

1. **Irreducibility**: For any state of the Markov chain, there is a positive probability of visiting all other states.

That is, the matrix T cannot be reduced to separate smaller matrices, which is also the same as stating that the transition graph is connected.

2. **Aperiodicity**: The chain should not get trapped in cycles.

Google's strategy is to add am matrix of uniform noise E to T:

$$L = T + \epsilon E$$

where ϵ is a small number. L is then normalised. This ensures irreducibility.

How quickly does this algorithm converge? What determines the rate of convergence? Again matrix algebra and spectral theory provide the answers:

⋆

Lecture 3 - *The Singular Value Decomposition (SVD)*

OBJECTIVE: The SVD is a matrix factorization that has many applications: e.g., information retrieval, least-squares problems, image processing.

◇ EIGENVALUE DECOMPOSITION

Let $\mathbf{A} \in \mathbb{R}^{m \times m}$. If we put the eigenvalues of $\mathbf{A}$ into a diagonal matrix $\mathbf{\Lambda}$ and gather the eigenvectors into a matrix $\mathbf{X}$, then the eigenvalue decomposition of $\mathbf{A}$ is given by

$$\mathbf{A} = \mathbf{X}\mathbf{\Lambda}\mathbf{X}^{-1}.$$

But what if $\mathbf{A}$ is not a square matrix? Then the SVD comes to the rescue.

◇ FORMAL DEFINITION OF THE SVD

Given $\mathbf{A} \in \mathbb{R}^{m\times n}$, the SVD of $\mathbf{A}$ is a factorization of the form

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

where $\mathbf{u}$ are the left **singular vectors**, σ are the **singular values** and $\mathbf{v}$ are the right singular vectors.

$\mathbf{\Sigma} \in \mathbb{R}^{n\times n}$ is diagonal with positive entries (singular values in the diagonal).
$\mathbf{U} \in \mathbb{R}^{m\times n}$ with orthonormal columns.
$\mathbf{V} \in \mathbb{R}^{n\times n}$ with orthonormal columns.
($\Rightarrow$ $\mathbf{V}$ is orthogonal so $\mathbf{V}^{-1} = \mathbf{V}^T$)

The equations relating the right singular values $\{\mathbf{v}_j\}$ and the left singular vectors $\{\mathbf{u}_j\}$ are

$$\mathbf{A}\mathbf{v}_j = \sigma_j \mathbf{u}_j \qquad j = 1, 2, \ldots, n$$

i.e.,

$$\mathbf{A}\begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \ldots & \mathbf{v}_n \end{bmatrix} = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \ldots & \mathbf{u}_n \end{bmatrix} \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{bmatrix}$$

or $\mathbf{AV} = \mathbf{U\Sigma}$.

⋆

1. There is no assumption that $m \geq n$ or that $\mathbf{A}$ has full rank.

2. All diagonal elements of $\mathbf{\Sigma}$ are non-negative and in non-increasing order:

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p \geq 0$$

where $p = \min(m, n)$

Theorem 1 *Every matrix* $\mathbf{A} \in \mathbb{R}^{m \times n}$ *has singular value decomposition* $\mathbf{A} = \mathbf{U\Sigma V}^T$
Furthermore, the singular values $\{\sigma_j\}$ *are uniquely determined.*
If $\mathbf{A}$ *is square and* $\sigma_i \neq \sigma_j$ *for all* $i \neq j$*, the left singular vectors* $\{\mathbf{u}_j\}$ *and the right singular vectors* $\{\mathbf{v}_j\}$ *are uniquely determined to within a factor of* ± 1*.*

◇ EIGENVALUE DECOMPOSITION

Theorem 2 *The nonzero singular values of* $\mathbf{A}$ *are the (positive) square roots of the nonzero eigenvalues of* $\mathbf{A}^T\mathbf{A}$ *or* $\mathbf{A}\mathbf{A}^T$ *(these matrices have the same nonzero eigenvalues).*

⋆ Proof:

◇ LOW-RANK APPROXIMATIONS

Theorem 3 $\|\mathbf{A}\|_2 = \sigma_1$, *where* $\|\mathbf{A}\|_2 = \max_{\mathbf{x}\neq 0} \frac{\|\mathbf{Ax}\|}{\|\mathbf{x}\|} = \max_{\|\mathbf{x}\|\neq 1} \|\mathbf{Ax}\|$.

⋆ Proof:

Another way to understand the SVD is to consider how a matrix may be represented by a sum of rank-one matrices.

Theorem 4

$$\mathbf{A} = \sum_{j=1}^{r} \sigma_j \mathbf{u}_j \mathbf{v}_j^T$$

where r *is the rank of* $\mathbf{A}$.

⋆ Proof:

What is so useful about this expansion is that the ν^{th} *partial sum captures as much of the "energy" of* $\mathbf{A}$ *as possible by a matrix of at most rank-*ν. In this case, "energy" is defined by the 2-norm.

Theorem 5 *For any* ν *with* $0 \leq \nu \leq r$ *define*

$$\mathbf{A}_\nu = \sum_{j=1}^{\nu} \sigma_j \mathbf{u}_j \mathbf{v}_j^T$$

If $\nu = p = \min(m, n)$*, define* $\sigma_{\nu+1} = 0$*.*

Then,

$$\|\mathbf{A} - \mathbf{A}_\nu\|_2 = \sigma_{\nu+1}$$

Lecture 4 - *Fun with the SVD*

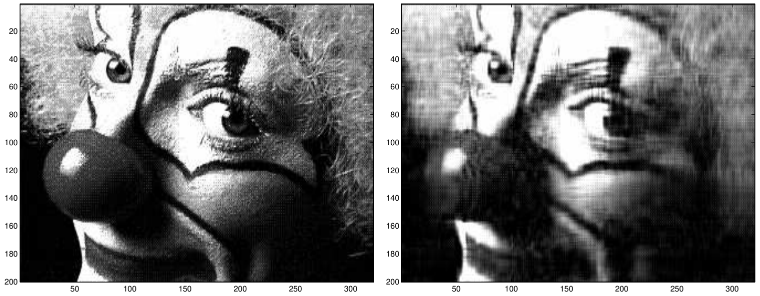
OBJECTIVE: Applications of the SVD to image compression, dimensionality reduction, visualization, information retrieval and latent semantic analysis.

◇ IMAGE COMPRESSION EXAMPLE

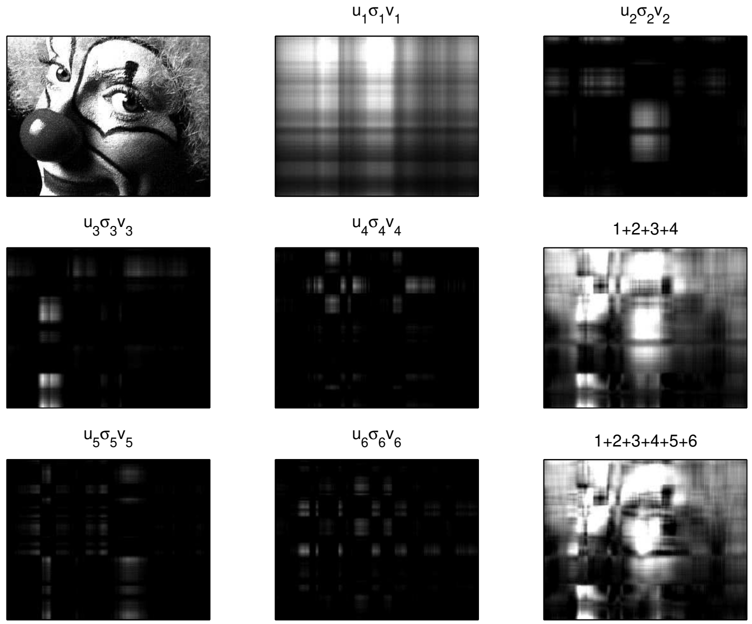
```
load clown.mat;
figure(1)
colormap('gray')
image(A);

[U,S,V] = svd(A);
figure(2)
k = 20;
colormap('gray')
image(U(:,1:k)*S(1:k,1:k)*V(:,1:k)');
```

The code loads a clown image into a 200×320 array $\mathbf{A}$; displays the image in one figure; performs a singular value decomposition on $\mathbf{A}$; and displays the image obtained from a rank-20 SVD approximation of $\mathbf{A}$ in another figure. Results are displayed below:



The original storage requirements for $\mathbf{A}$ are $200 \cdot 320 = 64,000$, whereas the compressed representation requires $(200 + 300 + 1) \cdot 20 \approx 10,000$ storage locations.



Smaller eigenvectors capture high frequency variations (small brush-strokes).

◇ TEXT RETRIEVAL - LSI

The SVD can be used to cluster documents and carry out information retrieval by using concepts as opposed to word-matching. This enables us to surmount the problems of synonymy (car,auto) and polysemy (money bank, river bank). The data is available in a term-frequency matrix

⋆

If we truncate the approximation to the k-largest singular values, we have

$$\mathbf{A} = \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^T$$

So

$$\mathbf{V}_k^T = \mathbf{\Sigma}_k^{-1} \mathbf{U}_k^T \mathbf{A}$$

⋆

In English, $\mathbf{A}$ is projected to a lower-dimensional space spanned by the k singular vectors $\mathbf{U}_k$ (eigenvectors of $\mathbf{A}\mathbf{A}^T$). To carry out **retrieval**, a **query** $\mathbf{q} \in \mathbb{R}^n$ is first projected to the low-dimensional space:

$$\widehat{\mathbf{q}}_k = \mathbf{\Sigma}_k^{-1}\mathbf{U}_k^T\mathbf{q}$$

And then we measure the angle between $\widehat{\mathbf{q}}_k$ and the $\mathbf{v}_k$.

⋆

◇ PRINCIPAL COMPONENT ANALYSIS (PCA)

The columns of $\mathbf{U\Sigma}$ are called the **principal components** of $\mathbf{A}$. We can project high-dimensional data to these components in order to be able to visualize it. This idea is also useful for cleaning data as discussed in the previous text retrieval example.

⋆

For example, we can take several 16×16 images of the digit 2 and project them to 2D. The images can be written as vectors with 256 entries. We then from the matrix $\mathbf{A} \in \mathbb{R}^{n \times 256}$, carry out the SVD and truncate it to $k = 2$. Then the components $\mathbf{U}_k \mathbf{\Sigma}_k$ are 2 vectors with n data entries. We can plot these 2D points on the screen to visualize the data.

★