Boltzmann Machine

Overview and Prospects

Hooman Zolfaghari
Ramtin Moslemi
Borna (lastname)
Houman (lastname)

Machine Learning & Physics

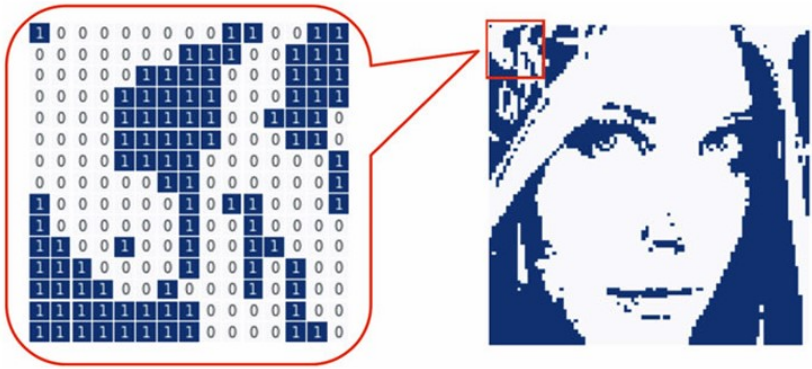
How are they related

- Bridge between physics and machine learning: **information**
- Amount of information = Extent of surprise
- Amount of information of event $A = -\log \mathbb{P}(\text{event } A)$.

Little “information” $\Leftrightarrow$ difficult to predict $\Leftrightarrow$ large information entrop

A lot of “information” $\Leftrightarrow$ easy to predict $\Leftrightarrow$ small information entropy

Maxwell’s demon & Drawing Decisions



Machine Learning Overview

> "*Intelligence is not just about pattern recognition and function approximation. It's about modeling the world*". — Josh Tenenbaum, NeurIPS 2021.

- We have an observable space $\mathcal{Z}$, a probability measure $\mathbb{P}$, a sample $S = \{Z_i\}_{i=1}^n$.
- Formulate the problem with a Hypothesis set $\mathcal{H}$ and a "Loss function".
- (Ultimate goal) Statistical Risk: $\arg\min_{h \in \mathcal{H}} R(h) = \mathbb{E}(\ell(h))$
- Empirical Risk: $\hat{R}_S(h) = \frac{1}{n} \sum_{Z \in S} \ell(h(Z))$

Artificial Neural Network

- A feed-forward neural network defines a function $f : \mathbb{R}^d \to \mathbb{R}^k$ recursively via layers .

$$f(X) = f^{(L)} \circ f^{(L-1)} \circ \cdots \circ f^{(1)}(X) \quad f^{(i)}(X^{(i)}) = \sigma(WX^{(i)} + b)$$

Universal Approximation Theorem:

For any continuous function $g : \mathbb{R}^d \to \mathbb{R}^k$ on a compact subset $K \subset \mathbb{R}^d$ and for any $\varepsilon > 0$, there exists a neural network $f \in \mathcal{H}$ such that:

$$\sup_{x \in K} \left| g(x) - f(x) \right| < \varepsilon.$$

Training ANN

- **Gradient Descent:**

$$\theta \to f, \quad \theta_{t+1} = \theta_t - \eta \nabla_\theta \hat{R}_n(f)$$

- In infinite-width limit, the distribution over outputs of a randomly initialized neural network converges to a Gaussian Process
- In gradient descent, the evolution of the network's predictions can be approximated by a linear model characterized by the NTK:

$$\Theta(x, x') = \langle \nabla_\theta f(x; \theta), \nabla_\theta f(x'; \theta) \rangle.$$

- In infinite-width limit, the NTK can guarantee convergence under some assumptions.

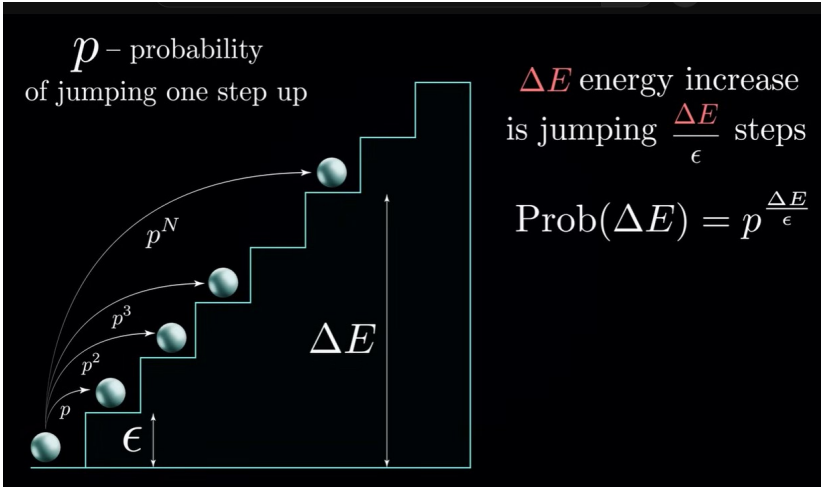
Energy Based Models

Boltzman Machine

Introduction

- Idea was taken from statistical physics and formulated by cognitive scientists.
- The Hopfield network can reproduce specific patterns it has memorized
- Goal: "Understanding" the data instead of only memorizing
- Boltzman machine can generate new patterns it has never seen before.
- Multiple potential outputs (non-deterministic)
- BM = Hopfiel network + Stochasticity + Hidden Units

Introduction



Introduction

- We can formulate this by denoting $T = \frac{-\ln(p)}{\epsilon}$, as:

$$\mathbb{P}(\Delta E) = e^{\frac{-\Delta E}{kT}} \tag{1}$$

- This gives us relative probability of changing energy.
- Knowing all probabilities have volume one, gives us absolute probability of each state.
- Giving us the partition function as the total volume.

$$\mathbb{P}(E) = \frac{1}{Z} e^{-E/T} \tag{2}$$

Inference

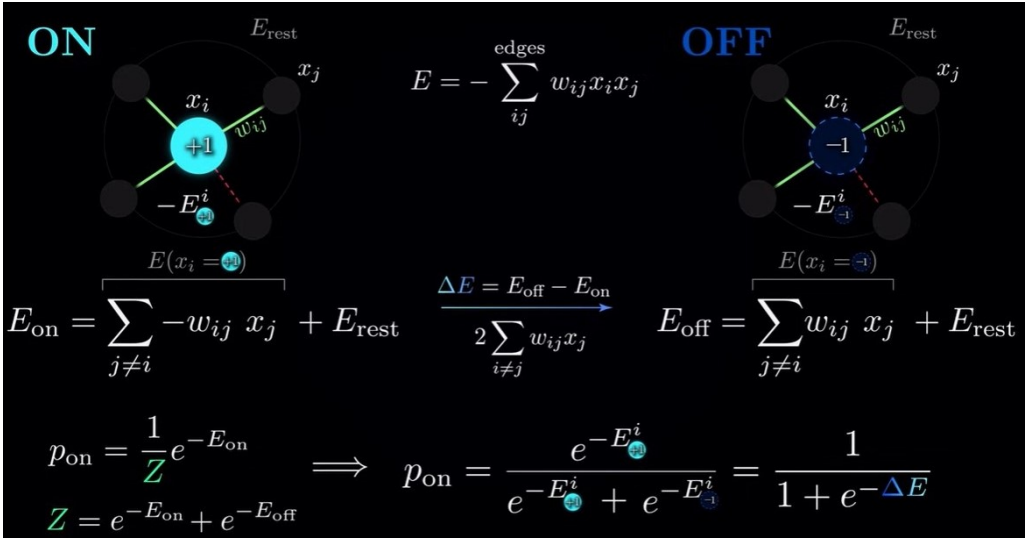
- Hopfield Network's **deterministic** update rule:

$$h_i = \sum_{j \neq i} w_{ij} x_j \rightarrow \text{input to neuron } i, \quad x_i = \begin{cases} +1 & h_i > 0 \\ -1 & h_i < 0 \end{cases} \rightarrow \text{update rule}$$

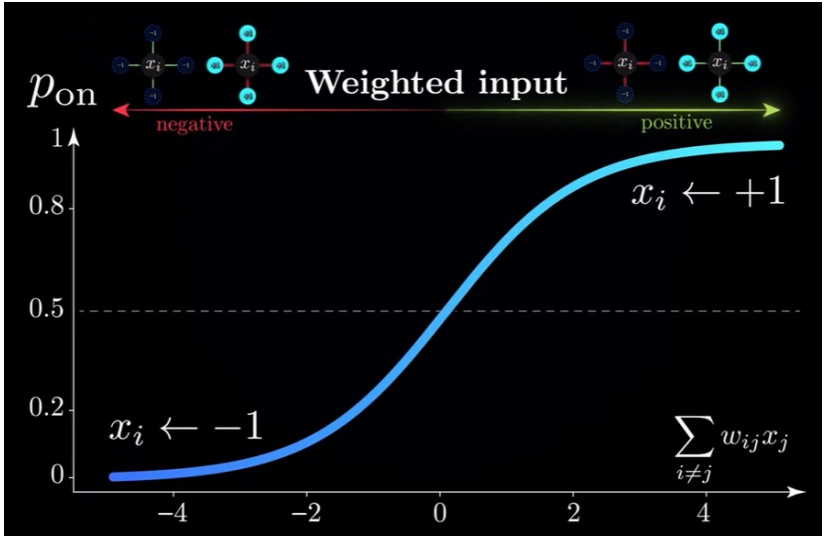
- Always Greedy moving to the lowest energy state possible
- Boltzmann Machine update rule:

$$x_i = \begin{cases} +1 & \text{with probability } \mathbb{P}(E_{\text{on}}) \\ -1 & \text{with probability } 1 - \mathbb{P}(E_{\text{off}}) \end{cases}$$

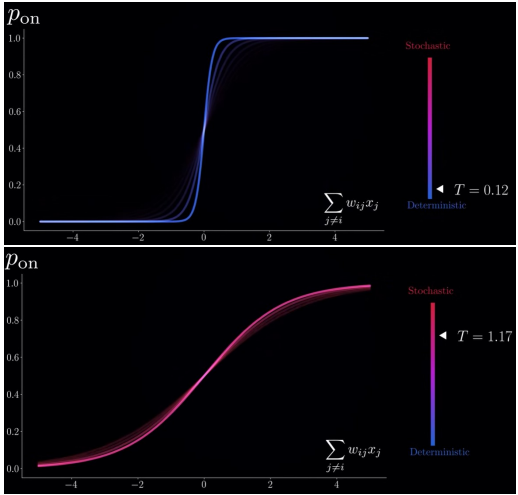
Inference


ON
E_{rest}
x_j
x_i
w_{ij}
+1
$-E^i_{+1}$
$E = -\sum_{ij}^{\text{edges}} w_{ij} x_i x_j$
OFF
E_{rest}
x_j
x_i
w_{ij}
−1
$-E^i_{-1}$
$E_{\text{on}} = \sum_{j \neq i} -w_{ij}\ x_j\ + E_{\text{rest}}$
$E(x_i = +1)$
$\Delta E = E_{\text{off}} - E_{\text{on}}$
$= 2\sum_{i \neq j} w_{ij} x_j$
$E_{\text{off}} = \sum_{j \neq i} w_{ij}\ x_j\ + E_{\text{rest}}$
$E(x_i = -1)$
$p_{\text{on}} = \frac{1}{Z} e^{-E_{\text{on}}}$
$Z = e^{-E_{\text{on}}} + e^{-E_{\text{off}}}$
$\Longrightarrow$
$p_{\text{on}} = \frac{e^{-E^i_{+1}}}{e^{-E^i_{+1}} + e^{-E^i_{-1}}} = \frac{1}{1 + e^{-\Delta E}}$

Inference


p_{on}
Weighted input
negative
positive
1
0.8
0.5
0.2
0
$x_i \leftarrow +1$
$x_i \leftarrow -1$
$\sum_{i \neq j} w_{ij} x_j$
−4
−2
0
2
4

Inference


p_{on}
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4
$\sum_{j \neq i} w_{ij} x_j$
Stochastic
Deterministic
$T = 0.12$
p_{on}
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4
$\sum_{j \neq i} w_{ij} x_j$
Stochastic
Deterministic
$T = 1.17$

Learning

- Instead of memorizing patterns we want to learn probability distribution of data.
- We want the training data to have high probability
- Notice that increasing a states probability affects other states probability through partition function

$$\log \mathbb{P}(\text{data states}) = \sum_{i=1}^{N} \log(\frac{1}{Z} e^{-E(x^{(i)})/T})$$

Learning

$$\underbrace{\log \mathbb{P}(\text{data states})}_{\text{Maximaize}} = -\frac{1}{T}[\sum_{i=1}^{N} \underbrace{E(x^{(i)})}_{\text{Minimize}}] - N \underbrace{\log Z}_{\text{Minimize}}$$

- Taking derivative with each w_{ij}:
  - First term:(similar to Hopfield) $\frac{\partial E(x^{(i)})}{\partial w_{ij}} = -(x_i x_j)$
  - Second term: $\frac{\partial \log Z}{\partial w_{ij}} = \sum_{s \in \text{all states}} \mathbb{P}(s) x_i^s x_j^s$
- Giving us the **Contrastive Hebbian Rule**:

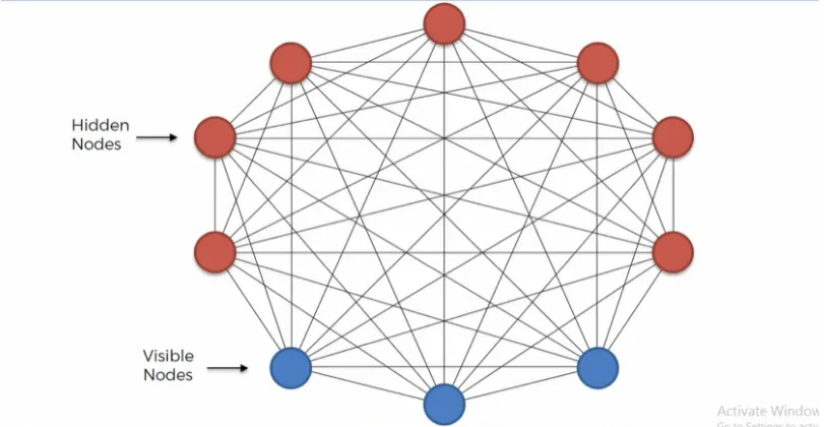
$$\Delta w_{ij} = \frac{1}{N} \sum_{n=1}^{N} x_i^{(n)} x_j^{(n)} - \mathbb{E}[x_i^s x_j^s]$$

- second term computed by iterative sampling until equilibrium.

Hidden Units

- Visible Units: States directly encoding the data
- Hidden Units: Internal representation of latent variables
- We use the same update rule in inference
- In Learning, input the data and sample the hidden state

Hidden Units



Energy Based Model

- Energy function:

$$E(x;\theta) = -(\mathbf{x^T W x} + \mathbf{b^T x})$$

- Parameters same as in *Hopfield* networks and *Ising* models
- Problem: Hard to train (due to the partition function)
- Solution:
  - Introducing latent variables
  - Restricting connections among observables.

Restricting BMs

- Consider: binary observable variables $\mathbf{x} \in \{\mathbf{0}, \mathbf{1}\}^{\mathbf{D}}$ and binary latent (hidden) variables $\mathbf{z} \in \{\mathbf{0}, \mathbf{1}\}^{\mathbf{M}}$
- The relationships among variables specified through this *energy function*:

$$E(\mathbf{x}, \mathbf{z}; \theta) = -\mathbf{x}^{\mathbf{T}}\mathbf{W}\mathbf{z} - \mathbf{b}^{\mathbf{T}}\mathbf{x} - \mathbf{c}^{\mathbf{T}}\mathbf{x} \tag{3}$$

- For this EF, the RBM is defined by the *Gibbs* distribution:

$$p(\mathbf{x}, \mathbf{z}; \theta) = \frac{\mathbf{1}}{\mathbf{Z}_\theta} \exp(-\mathbf{E}(\mathbf{x}, \mathbf{z}; \theta)) \tag{4}$$

Restricting BMs

- the *partition function*:

$$Z_\theta = \sum_{\mathbf{x}} \sum_{\mathbf{z}} \exp\big(-E(\mathbf{x}, \mathbf{z}; \theta)\big) \tag{5}$$

- The marginal probability over observables (the likelihood of observation):

$$p(\mathbf{x}|\theta) = \frac{1}{Z_\theta} \exp\big(-F(\mathbf{x}; \theta)\big) \tag{6}$$

- where $F(\cdot)$ is the *free energy*:

$$F(\mathbf{x}; \theta) = -\mathbf{b}^\top \mathbf{x} - \sum_j \log\big(1 + \exp(c_j + (\mathbf{W}_{\cdot j})^\top \mathbf{x})\big). \tag{7}$$

RBM

- The presented model is called a restricted Boltzmann machine (RBM).
- Useful property: the conditional distribution over the hidden variables factorizes given the observable variables and vice versa:

$$p(z_m = 1|\mathbf{x}, \theta) = \text{sigm}\big(c_m + (\mathbf{W}_{\cdot m})^\top \mathbf{x}\big), \tag{8}$$

$$p(x_d = 1|\mathbf{z}, \theta) = \text{sigm}\big(b_d + \mathbf{W}_{d\cdot}\mathbf{z}\big). \tag{9}$$

Learning RBMs

- For given data $\mathcal{D} = \{\mathbf{x}_n\}_{n=1}^N$
- Train an RBM using the maximum likelihood

$$\ell(\theta) = \frac{1}{N} \sum_{\mathbf{x}_n \in \mathcal{X}} \log p(\mathbf{x}_n \mid \theta) \tag{10}$$

- The gradient with respect to θ:

$$\nabla_\theta \ell(\theta) = -\frac{1}{N} \sum_{n=1}^{N} \left(\nabla_\theta F(\mathbf{x}_n; \theta) - \sum_{\hat{\mathbf{x}}} p(\hat{\mathbf{x}}|\theta) \nabla_\theta F(\hat{\mathbf{x}}; \theta) \right) \tag{11}$$

- Cannot be computed analytically because the second term requires summing over all configurations of observables.

Learning RBMs

- One way to sidestep this: Stochastic approximation
- Replacing the expectation under $p(\mathbf{x}|\theta)$ by a sum over S samples $\{\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_S\}$ drawn according to $p(\mathbf{x}|\theta)$:

$$\nabla_\theta \ell(\theta) \approx -\frac{1}{N}\sum_{n=1}^{N} \nabla_\theta F(\mathbf{x}_n;\theta) - \frac{1}{S}\sum_{s=1}^{S} \nabla_\theta F(\hat{\mathbf{x}}_s;\theta). \tag{12}$$

Learning RBMs

- A different approach: *contrastive divergence*
- Approximates the expectation under $p(\mathbf{x}|\theta)$ by a sum over samples $\tilde{\mathbf{x}}_n$ drawn from a distribution obtained by applying K steps of the block Gibbs sampling procedure:

$$\nabla_\theta \ell(\theta) \approx -\frac{1}{N} \sum_{n=1}^{N} \big(\nabla_\theta F(\mathbf{x}_n; \theta) - \nabla_\theta F(\tilde{\mathbf{x}}_n; \theta) \big). \tag{13}$$

Learning RBMs

- The original CD used K steps of the Gibbs chain, starting and is restarted after every parameter update.
- An alternative approach, Persistent Contrastive Divergence (PCD) does not restart the chain after each update typically resulting in a slower convergence rate but eventually better performance

Higher-Order Relationships

- The energy function allows the modeling of higher-order dependencies among variables.
- For instance: Third-order multiplicative interactions by introducing two kinds of hidden variables
  - **Subspace Units:** Reflect feature variations, robust to invariances:
  - **Gate Units:** Activate subspace units, pool subspace features.

Higher-Order Relationships

Random Variables for SubspaceRBM

- Observables: $\mathbf{x} \in \{0,1\}^D$.
- Gate Units: $\mathbf{h} \in \{0,1\}^M$.
- Subspace Units: $\mathbf{S} \in \{0,1\}^{M \times K}$.
- **Connections:** x_i, h_j, and s_{jk}.

Higher-Order Relationships

Energy Function for SubspaceRBM

$$E(\mathbf{x}, \mathbf{h}, \mathbf{S}; \theta) = -\sum_{i=1}^{D}\sum_{j=1}^{M}\sum_{k=1}^{K} W_{ijk} x_i h_j s_{jk}$$
$$-\sum_{i=1}^{D} b_i x_i - \sum_{j=1}^{M} c_j h_j - \sum_{j=1}^{M} h_j \sum_{k=1}^{K} D_{jk} s_{jk}, \tag{14}$$

$$\theta = \{W, \mathbf{b}, \mathbf{c}, \mathbf{D}\}, \quad W \in \mathbb{R}^{D \times M \times K}, \quad \mathbf{b} \in \mathbb{R}^{D}, \quad \mathbf{c} \in \mathbb{R}^{M}, \quad \mathbf{D} \in \mathbb{R}^{M \times K}.$$

Higher-Order Relationships

Conditional Distributions in SubspaceRBM

$$p(x_i = 1|\mathbf{h}, \mathbf{S}) = \text{sigm}\left(\sum_j \sum_k W_{ijk} h_j s_{jk} + b_i\right) \tag{15}$$

$$p(s_{jk} = 1|\mathbf{x}, h_j) = \text{sigm}\left(\sum_i W_{ijk} x_i h_j + h_j D_{jk}\right) \tag{16}$$

$$p(h_j = 1|\mathbf{x}) = \text{sigm}\left(-K \log 2 + c_j + \sum_{k=1}^{K} \text{softplus}\left(\sum_i W_{ijk} x_i + D_{jk}\right)\right) \tag{17}$$

Research Prospect

Template Things

Single Column

For 4-by-3 aspect ratio slides, specify `standard` as an option to the document class. Write your presentation like a normal LaTeX file with a `\maketitle` command and `\chapter` and `\section` headings. The `\maketitle` contents are defined by the following macros:

```
\pretitle    \author
\title       \subtitle
```

The `\chapter` heading creates a slide with just the chapter name, and the `\section` heading sets the title of a new slide. However, if no text follows the section, no slide will be created. Text which does not fit on one slide will flow onto the next slide automatically.

Double Column

Use the `\twocolumn` and `\onecolumn` commands right after the section heading to control the number of columns. Text will flow from the left column to the right.

- Point one
- Point two
- Point three
- Point four
- Point five
- Point six
- Point seven
- Point eight
- Point nine
- Point ten
- Point eleven
- Point twelve

You can use `\pagebreak` to force text onto the next column.

Table of Stuff

You can create any variety of subdivisions on your slide by using the `tabular` environment.

Primary	Secondary	Tertiary
First	Second	Third
One	Two	Three

Alpha	Beta	Gamma
Green	Blue	Red
Cyan	Yellow	Magenta

The `\cellcolor` command sets the background color of a table cell.

Centering

Use the `Center` environment
to center horizontally *and* vertically.

Explicit Code

Python

Use the `python` environment for Python code.

```
def write_list(fid, x, level):
    ind = '  '*level
    xs = '0' if abs(x[0]) < 1e-3 else "%.3f"
    txt = '\n%svalues=\"%s' % (ind, xs)
    for n in range(1, len(x)):
        xs = '0' if abs(x[n]) < 1e-3 else "%.3f"
        if len(txt) + 3 + len(xs) >= 80:
            fid.write(txt + ';\n')
            txt = ind + '  ' + xs
        else:
            txt += '; ' + xs
    fid.write(txt + '\"')
```

Python

You can use the `\HL` command to highlight a line of code.

```
1  def write_list(fid, x, level):
2      ind = '  '*level
3      xs = '0' if abs(x[0]) < 1e-3 else "%.3f"
4      txt = '\n%svalues=\"%s' % (ind, xs)
5      for n in range(1, len(x)):
6          xs = '0' if abs(x[n]) < 1e-3 else "%.3f"
7          if len(txt) + 3 + len(xs) >= 80:
8              fid.write(txt + ';\n')
9              txt = ind + '  ' + xs
10         else:
11             txt += '; ' + xs
12     fid.write(txt + '\"')
```

MATLAB

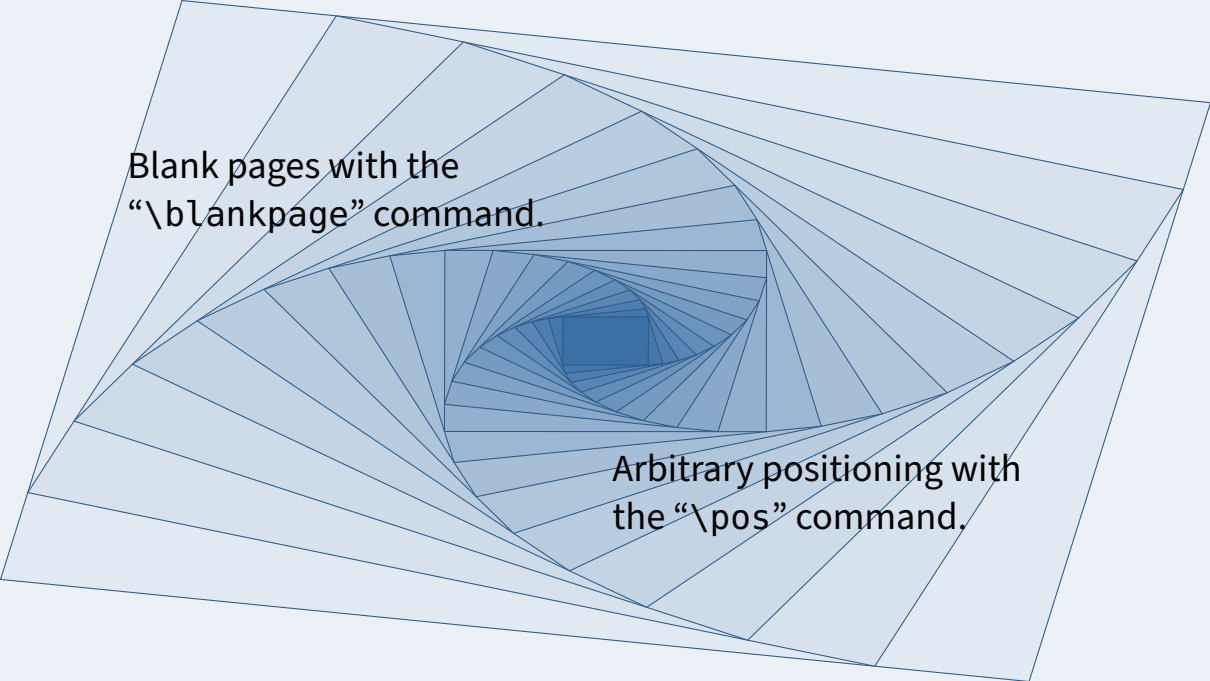
Use the `matlab` environment for MATLAB code.

```matlab
function savepdf(name, width, height)
    % name is the file name including ".pdf".
    % Both width and height are in (cm).
    set(gcf, 'units', 'centimeters', ...
        'position', [0, 0, width, height])
    set(gca, 'FontSize', 9);
    set(gca, 'FontName', 'Times New Roman');
    exportgraphics(gcf, name, ...
        'ContentType', 'vector');
end
```

R Language

Use the `rlang` environment for R code.

```r
factorial <- function(n) {
    if (n == 0 || n == 1) {
        return(1)
    } else {
        return(n * factorial(n - 1))
    }
}
```


Blank pages with the
“\blankpage” command.
Arbitrary positioning with
the “\pos” command.