General Framework

Offline Learning:
- Observe data \( X_1, X_2, \ldots, X_n \in \mathcal{X} \), where \( \mathcal{X} \) is the state space.
- Make decision (control) \( u(X_1, X_2, \ldots, X_n) \in \mathcal{U} \), where \( \mathcal{U} \) is the decision (control) space.
- Goal: Minimize an objective function \( \sum_{i=1}^{n} l(X_i, u) \).

Online Learning:
For \( t = 1, 2, \ldots, n \):
- Observe data \( X_t \in \mathcal{X} \), where \( \mathcal{X} \) is the state space.
- Make decision (control) \( u_t \in \mathcal{U} \), where \( \mathcal{U} \) is the decision (control) space.
- Suffer loss \( l(X_t, u_t) \).
Goal: Minimize regret \( \sum_{t=1}^{n} l(X_t, u_t) - \min_u \sum_{t=1}^{n} l(X_t, u) \).
Regression

Given a set of input/output pairs, where the output is a continuous value. Find a good function to predict the output given input.

- $\mathcal{X} = \{(x_i, y_i)\} \subset \mathbb{R}^d \times \mathbb{R}$.
- $\mathcal{U} = \{ f : \mathbb{R}^d \to \mathbb{R} \}$.
- $l(u, (x_i, y_i)) = ||u(x_i) - y_i||$.

Example ($d = 1$):

- Constant least squares regression.
  $\mathcal{U} = \{ c | f : x \to c \}$. $l(c, (x_i, y_i)) = (c - y_i)^2$.

- Linear least squares regression.
  $\mathcal{U} = \{ (a, b) | f : x \to ax + b \}$.
  $l((a, b), (x_i, y_i)) = (ax_i + b - y_i)^2$.

- Quadratic least absolute regression.
  $\mathcal{U} = \{ (a, b, c) | f : x \to ax^2 + bx + c \}$.
  $l((a, b, c), (x_i, y_i)) = |ax_i^2 + bx_i + c - y_i|$.
Solving Constant Least Squares and Absolute Regression

Constant Least Squares Regression: Minimize $\sum_{i=1}^{n} (c - y_i)^2$. Take derivatives respect to $c$:

$$\sum_{i=1}^{n} 2(c - y_i) = 0.$$ 

Solving for $c$ we get

$$c = \frac{1}{n} \sum_{i=1}^{n} y_i.$$ 

Constant Absolute Regression: Minimize $\sum_{i=1}^{n} \vert c - y_i \vert$. Solution: $c$ is the median of $y$. 
Classification

Given a set of input/output pairs, where the output is a discrete value. Find a good function to predict the output given input.

- $\mathcal{X} = \{(x_i, y_i)\} \subset \mathbb{R}^d \times \mathbb{Z}$.
- $\mathcal{U} = \{ f : \mathbb{R}^d \to \mathbb{Z} \}$.
- $l(u, (x_i, y_i)) = 1_{u(x_i) \neq y_i}$.

Example ($d = 1$):

- Binary Linear Logistic Regression.
  In this case $\mathcal{X} = \{(x_i, y_i)\} \subset \mathbb{R} \times \{0, 1\}$. We relax the action space to be $\mathcal{U} = \{ f : \mathbb{R} \to \mathbb{R} \}$:
  - $\mathcal{U} = \{(a, b) | f : x \to \frac{1}{1 + e^{-(ax + b)}} \}$.
  - $l((a, b), (x_i, y_i)) = \left( \frac{1}{1 + e^{-(ax_i + b)}} - y_i \right)^2$. 
Solution to Binary Constant Logistic Regression

The loss function is

\[ \sum_{i=1}^{n} l(c, (x_i, y_i)) = \sum_{i=1}^{n} \left( \frac{1}{1 + e^{-c}} - y_i \right)^2. \]

Taking the derivative with respect to \( c \),

\[ \sum_{i=1}^{n} 2 \left( \frac{1}{1 + e^{-c}} - y_i \right) \frac{1}{(1 + e^{-c})^2} = 0. \]

That implies

\[ \frac{1}{1 + e^{-c}} = \frac{1}{n} \sum_{i=1}^{n} y_i. \]

Given a new data point, we will predict it belongs to the majority class.
Decision Boundary for Classification
Support Vector Machines

Idea: Find a line that best separates the two classes. A line is described by

$$ax + by + c = 0.$$ 

A line that separates the two classes should satisfy

- $$ax_i + by_i + c > 0$$ for points in the first class
- $$ax_i + by_i + c < 0$$ for points in the second class

What if there are infinitely many lines qualify? Answer: Choose the one that has the largest margin!

Define

- $$ax_i + by_i + c \geq 1$$ for all points in the first class
- $$ax_i + by_i + c \leq -1$$ for all points in the second class

Objective: Maximize the distance between these two lines.
Distance Between Parallel Lines

How to calculate the distance between

\[ ax + by + c = 1 \quad \text{and} \quad ax + by + c = -1 \]

▶ Formulate it as an optimization problem.

\[
\begin{align*}
\minimize_{x_1, x_2, y_1, y_2} & \quad (x_1 - x_2)^2 + (y_1 - y_2)^2 \\
\text{subject to} & \quad ax_1 + by_1 + c = 1. \\
& \quad ax_2 + by_2 + c = -1.
\end{align*}
\]

▶ Use high school geometry.

The distance is

\[
\frac{1}{\sqrt{a^2 + b^2}}.
\]
Optimization for 2d Support Vector Machines

minimize $a^2 + b^2$

subject to $ax_i + by_i + c \geq 1$ for $z_i = 0$.

$ax_i + by_i + c \leq -1$ for $z_i = 1$.

What if the points are not linear separable?
Soft Margin and Nonlinear Support Vector Machines

\[
\begin{align*}
\text{minimize} & \quad a^2 + b^2 + \lambda \sum_{i=1}^{n} d_i \\
\text{subject to} & \quad ax_i + by_i + c \geq 1 - d_i \quad \text{for} \quad z_i = 0. \\
& \quad ax_i + by_i + c \leq -1 + d_i \quad \text{for} \quad z_i = 1.
\end{align*}
\]

\(\lambda\) is a parameter chosen a priori. It reflects the degree of tolerance.

If the points are not linear separable, they might be quadratic separable, etc. In general

\[
\begin{align*}
\text{minimize} & \quad d(f(x_i, y_i) = 1, f(x_i, y_i) = -1) \\
\text{subject to} & \quad f(x_i, y_i) \geq 1 \quad \text{for} \quad z_i = 0. \\
& \quad f(x_i, y_i) \leq -1 \quad \text{for} \quad z_i = 1.
\end{align*}
\]
**k Nearest Neighbors**

kNN is a simple classifier without a training process. A new point in the space is classified by a majority vote of its neighbors.

The action (control) it learns is an implicit function, and its functional form can be very complicated.
Linear Discriminant Analysis

Assumption: Data within each class is generated by a Normal distribution.

- For each of the $k$ classes, we compute its mean and variance.
- For a test data, we predict its class to be the one with the largest probability.
Artificial Neural Network

- Prediction: forward propagation
- Learning: backward propagation
Other Important Class of Offline Learning Models

- Ensemble learning (Bagging, Boosting, Random Forest)
- Hidden Markov models, the Kalman filter
- Probabilistic graphical models
- Bayesian nonparametrics (Gaussian Process, Dirichlet Process)
Online Learning and Regret

For \( t = 1, 2, \ldots, n \):

- Observe data \( X_t \in \mathcal{X} \), where \( \mathcal{X} \) is the state space.
- Make decision (control) \( u_t \in \mathcal{U} \), where \( \mathcal{U} \) is the decision (control) space.
- Suffer loss \( l(X_t, u_t) \).

Goal: Minimize regret

\[
R(n) = \sum_{t=1}^{n} l(X_t, u_t) - \min_u \sum_{t=1}^{n} l(X_t, u).
\]

Question: What is a good regret?

\[
\lim_{n \to \infty} \frac{R(n)}{n} = 0.
\]
Learnability of Online Classification

Take the decision space to be $\mathcal{U} = \{0, 1\}$. The loss is 

$l(X_t, u_t) = |u_t - y_t|$, where $y_t \in \{0, 1\}$ is the actual label for $X_t$.

$$R(n) = \sum_{t=1}^{n} |u_t - y_t| - \min_{u \in \{0,1\}} \sum_{t=1}^{n} |y_t - u|$$

Take the true answer $y_t = 1 - u_t$, then

$$\sum_{t=1}^{n} |u_t - y_t| = n.$$

But

$$\min_{u \in \{0,1\}} \sum_{t=1}^{n} |y_t - u| \leq n/2.$$ 

Therefore the regret is at least $R(n) = n/2$. 
Assumptions to Ensure Learnability

Idea: The adversary should not pick labels based on our actions.

Assumption:
- The adversary picks $h^*$ from a finite decision space $\mathcal{H}$.
- $y_t = h^*(X_t)$

An algorithm:

Initialize $\mathcal{U}_1 = \mathcal{H}$.
For $t = 1, 2, \ldots, n$:
- Observe data $X_t \in \mathcal{X}$.
- Pick any $h \in \mathcal{U}_t$, and make decision $u_t = h(X_t)$.
- Suffer loss $|h^*(X_t) - u_t|$.
- Update $\mathcal{U}_{t+1} = \{ h \in \mathcal{U}_t | h(X_t) = h^*(X_t) \}$.

The regret bound of this algorithm $R(n) \leq |\mathcal{H}| - 1$. 
The Halving Algorithm

Initialize $\mathcal{U}_1 = \mathcal{H}$.
For $t = 1, 2, \ldots, n$:

- Observe data $X_t \in \mathcal{X}$.
- Make decision $u_t = \arg\max_{z \in \{0, 1\}} |\{h \in \mathcal{U}_t | h(X_t) = z\}|$.
- Suffer loss $|h^*(X_t) - u_t|$.
- Update $\mathcal{U}_{t+1} = \{h \in \mathcal{U}_t | h(X_t) = h^*(X_t)\}$.

When the algorithm makes an incorrect prediction, we have

$$|\mathcal{U}_{t+1}| \leq |\mathcal{U}_t|/2.$$ 

The regret bound of this algorithm $R(n) \leq \log_2(|\mathcal{H}|)$. 


Relaxing the Assumption

Assumption:

- At each round $t$ The adversary picks $h_t$ from a finite decision space $\mathcal{H}$.
- $y_t = h_t(X_t)$.

Randomization: Given data $X_t$, we predict 1 with probability

$$\sum_{i=1}^{\mid \mathcal{H} \mid} w_i h_i(X_t).$$

The expected loss is

$$\mid \sum_{i=1}^{\mid \mathcal{H} \mid} w_i h_i(X_t) - y_t \mid.$$
Weighted Majority

Idea: Maintain a trust for each hypothesis $h$. Update the trust according to its observed loss.

Initialize $w^{(0)} = (1/|\mathcal{H}|, 1/|\mathcal{H}|, \ldots, 1/|\mathcal{H}|)$.

For $t = 1, 2, \ldots, n$:

- Observe data $X_t \in \mathcal{X}$.
- Predict $u_t = \sum_{i=1}^{|\mathcal{H}|} w_i^{(t)} h_i(X_t)$.
- Suffer loss $|y_t - u_t|$.
- Update $w_i^{(t+1)} = \frac{w_i^{(t)} \exp(-\eta|y_t - h_i(X_t)|)}{\sum_i w_i^{(t)} \exp(-\eta|y_t - h_i(X_t)|)}$.

$\eta$ is the learning rate. One can prove the total loss is upper bounded by

$$\frac{1}{1 - \eta} \left( \min_{i \in \mathcal{H}} \sum_{t=1}^T |h_i(X_t) - y_t| + \frac{\log d}{\eta} \right).$$
Towards Infinity

In most machine learning applications the hypothesis space $\mathcal{H}$ is infinite.

Initialize $w^{(0)}$ to be some distribution in the hypothesis space. Update the weight $w^{(t+1)}(h)$ by

$$w^{(t)}(h) \exp(-\eta|y_t - h(X_t)|)$$

$$\frac{\int_h w^{(t)}(h) \exp(-\eta|y_t - h(X_t)|) dh}{\int_h w^{(t)}(h) \exp(-\eta|y_t - h(X_t)|) dh}$$

This shares the idea with that of the Bayesian approach to machine learning.
Example of Online Learning in Finance

\( V_t \): Value of investment at the end of trading period \( t \).

\( p_t(i) \): Proportion of investment in stock \( i \) during trading period \( t \).

\( i = 1, 2, \ldots, m \).

\( r_t(i) \): Relative change of stock price during trading period \( t \).

\[
V_t = \sum_{i=1}^{m} V_{t-1} p_t(i) r_t(i) = V_{t-1} \sum_{i=t}^{m} p_t(i) r_t(i) = V_{t-1} p_t^T r_t.
\]

The general solution for \( V_t \) is:

\[
V_t = V_0 \prod_{i=1}^{T} p_t^T r_t.
\]
Constantly Rebalanced Portfolio

Constantly rebalanced portfolio:

\[ p_t = p \quad \text{for} \quad t = 1, 2, \ldots \]

Denote the value using this strategy after time \( t \) as \( V_t^p \).

Relative performance to this benchmark:

\[
\max_p \frac{V_n}{V_n^p}
\]

Minimizing the negative of the logarithm of the above objective gives

\[
\sum_{t=1}^{n} - \log(p_t^T r_t) - \min_p \sum_{t=1}^{n} - \log(p^T r_t)
\]

This is a regret, with a loss function \( l(X_t, u_t) = -\log(u_t^T X_t) \).