Computational Machine Learning

September 2, 2015
What types of problems are we solving?

- In data science problems, we generally need to:
  - Make a decision
  - Take an action
  - Produce some output
- Have some evaluation criterion
On page 3 of your document, the section titled "Actions" begins with a definition: "An action is the generic term for what is produced by our system." Below this, examples of actions are listed:

- Produce a 0/1 classification [classical ML]
- Reject hypothesis that $\theta = 0$ [classical Statistics]
- Written English text [speech recognition]
- Probability that a picture contains an animal [computer vision]
- Probability distribution on the earth [storm tracking]
- Adjust accelerator pedal down by 1 centimeter [automated driving]
Decision theory is about finding “optimal” actions, under various definitions of optimality.

Examples of Evaluation Criteria

- Is classification correct?
- Does text transcription exactly match the spoken words?
  - Should we give partial credit? How?
- Is probability “well-calibrated”? 
Real Life: Formalizing a Business Problem

- First two steps to formalizing a problem:
  1. Define the *action space* (i.e. the set of possible actions)
  2. Specify the evaluation criterion.

- Finding *the right formalization* can be an interesting challenge

- Formalization may evolve gradually, as you understand the problem better
Inputs

Most problems have an extra piece, going by various names:

- Inputs [ML]
- Covariates [Statistics]
- Side Information [Various settings]

Examples of Inputs

- A picture
- A storm’s historical location and other weather data
- A search query
Output / Outcomes

Inputs often paired with *outputs* or *outcomes*

Examples of outputs / outcomes

- Whether or not the picture actually contains an animal
- The storm’s location one hour after query
- Which, if any, of suggested the URLs were selected
Typical Sequence of Events

Many problem domains can be formalized as follows:

1. Observe input $x$.
2. Take action $a$.
3. Observe outcome $y$.
4. Evaluate action in relation to the outcome: $\ell(a, y)$.

Note

- Outcome $y$ is often **independent** of action $a$
- But this is **not always** the case:
  - URL recommendation
  - automated driving
Some Formalization

The Spaces

- \( \mathcal{X} \): input space
- \( \mathcal{Y} \): output space
- \( \mathcal{A} \): action space

Decision Function

A decision function produces an action \( a \in \mathcal{A} \) for any input \( x \in \mathcal{X} \):

\[
f : \mathcal{X} \rightarrow \mathcal{A}
\]

\[
x \mapsto f(x)
\]

Loss Function

A loss function evaluates an action in the context of the output \( y \).

\[
\ell : \mathcal{A} \times \mathcal{Y} \rightarrow \mathbb{R}^{\geq 0}
\]

\[
(a, y) \mapsto \ell(a, y)
\]
First two steps to formalizing a problem:

1. Define the action space (i.e. the set of possible actions)
2. Specify the evaluation criterion.

When a “stakeholder” asks the data scientist to solve a problem, she
may have an opinion on what the action space should be, and
hopefully has an opinion on the evaluation criterion, but
she really cares about your producing a “good” decision function.

Typical sequence:

1. Stakeholder presents problem to data scientist
2. Data scientist produces decision function
3. Engineer deploys “industrial strength” version of decision function
Evaluating a Decision Function

- Loss function $\ell$ evaluates a single action
- How to evaluate the decision function as a whole?
- We will use the standard statistical learning theory framework.
Setup for Statistical Learning Theory

Data Generating Assumption

All pairs \((X, Y) \in \mathcal{X} \times \mathcal{Y}\) are drawn i.i.d. from some unknown \(P_{\mathcal{X} \times \mathcal{Y}}\).
Definition

The expected loss or “risk” of a decision function $f : \mathcal{X} \rightarrow \mathcal{A}$ is

$$R(f) = \mathbb{E}_\ell(f(X), Y),$$

where the expectation taken is over $(X, Y) \sim P_{\mathcal{X} \times \mathcal{Y}}$.

Risk function cannot be computed

Since we don’t know $P_{\mathcal{X} \times \mathcal{Y}}$, we cannot compute the expectation. But we can estimate it...
Definition

A **Bayes decision function** $f^* : \mathcal{X} \rightarrow \mathcal{A}$ is a function that achieves the *minimal risk* among all possible functions:

$$R(f^*) = \inf_{f} R(f),$$

where the infimum is taken over all measurable functions from $\mathcal{X}$ to $\mathcal{A}$. The risk of a Bayes decision function is called the **Bayes risk**.

- A Bayes decision function is often called the “target function”, since it’s what we would ultimately like to produce as our decision function.
Example 1: Least Squares Regression

- spaces: $\mathcal{A} = \mathcal{Y} = \mathbb{R}$
- square loss:
  \[ \ell(a, y) = \frac{1}{2}(a - y)^2 \]
- mean square risk:
  \[ R(f) = \frac{1}{2} \mathbb{E}[(f(X) - Y)^2] \]
  \[ = \frac{1}{2} \mathbb{E}[(f(X) - \mathbb{E}[Y|X])^2] + \frac{1}{2} \mathbb{E}[(Y - \mathbb{E}[Y|X])^2] \]
- target function:
  \[ f^*(x) = \mathbb{E}[Y|X = x] \]
Example 2: Multiclass Classification

- **Spaces:** \( \mathcal{A} = \mathcal{Y} = \{0, 1, \ldots, K - 1\} \)

- **0-1 Loss:**
  \[
  \ell(a, y) = 1(a \neq y)
  \]

- **Risk is Misclassification Error Rate**
  \[
  R(f) = \mathbb{E}[1(f(X) \neq Y)]
  = \mathbb{P}(f(X) \neq Y)
  \]

- **Target Function is the Assignment to the Most Likely Class**
  \[
  f^*(x) = \arg \max_{1 \leq k \leq K} \mathbb{P}(Y = k \mid X = x)
  \]
But we can’t compute the risk!

- Can’t compute $R(f) = \mathbb{E}_\ell(f(X), Y)$ because we don’t know $P_{X \times Y}$.

- Can we estimate $P_{X \times Y}$ from data?

- Under assumptions (e.g. comes from a parametric family), yes.
  - We’ll come back to these approaches later in the course.

- Otherwise, we’ll typically face a curse of dimensionality,
  - making $P_{X \times Y}$ very difficult to estimate.
A Curse of Dimensionality

The “volume” of space grows exponentially with the dimension.

Histograms

- Construct histogram for $X \in [0, 1]$ with bins of size 0.1
  - That’s 10 bins.
  - About 100 observations would be a good start for estimation.
- Construct histogram for $X \in [0, 1]^{10}$ with hypercube bins of side length 0.1
  - That’s $10^{10} = 10$ billion bins.
  - About 100 billion observations would be a good start for estimation...

Takeaway Message

To estimate a density in high dimensions, you need additional assumptions.
The Empirical Risk Functional

Can we estimate $R(f)$ without estimating $\mathcal{P}_{\mathcal{X} \times \mathcal{Y}}$?

Assume we have sample data

Let $\mathcal{D}_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ be drawn i.i.d. from $\mathcal{P}_{\mathcal{X} \times \mathcal{Y}}$.

Definition

The empirical risk of $f : \mathcal{X} \rightarrow \mathcal{A}$ with respect to $\mathcal{D}_n$ is

$$\hat{R}_n(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(X_i), Y_i).$$

By the Strong Law of Large Numbers,

$$\lim_{n \rightarrow \infty} \hat{R}_n(f) = R(f),$$

almost surely.

That’s a start...
Empirical Risk Minimization

We want risk minimizer, is empirical risk minimizer close enough?

Definition
A function \( \hat{f} \) is an \textbf{empirical risk minimizer} if

\[
\hat{R}_n(\hat{f}) = \inf_{f} \hat{R}_n(f),
\]

where the minimum is taken over all [measurable] functions.
Supervised learning

Learning by heart aka overfitting

Data: \((x_i, y_i)_{i=1, \ldots, n} \in \mathbb{R}^2 \times \{-1, +1\}\)
ERM led to a function $f$ that just memorized the data.

How to spread information or “generalize” from training inputs to new inputs?

- Need to smooth things out somehow...
- A lot of modeling is about spreading and extrapolating information from one part of the input space $X$ into unobserved parts of the space.
Aside: Notation for Function Spaces

**Notation**

Let $\mathcal{C}^\mathcal{D}$ denote the set of all functions mapping from $\mathcal{D}$ [the domain] to $\mathcal{C}$ [the codomain].
Hypothesis Spaces

Definition
A hypothesis space \( \mathcal{F} \subset \mathcal{A}^X \) is a set of decision functions we are considering as solutions.

Hypothesis Space Choice

- Easy to work with.
- Includes only those functions that have desired “smoothness”
Constrained Empirical Risk Minimization

- Hypothesis space $\mathcal{F} \subset \mathcal{A}^X$, a set of functions mapping $X \rightarrow \mathcal{A}$

- **Empirical risk minimizer** (ERM) in $\mathcal{F}$ is $\hat{f} \in \mathcal{F}$, where

\[
\hat{R}(\hat{f}) = \inf_{f \in \mathcal{F}} \hat{R}(f) = \inf_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(X_i), Y_i).
\]

- Risk minimizer in $\mathcal{F}$ is $f^*_\mathcal{F} \in \mathcal{F}$, where

\[
R(f^*_\mathcal{F}) = \inf_{f \in \mathcal{F}} R(f) = \inf_{f \in \mathcal{F}} \mathbb{E} \ell(f(X), Y)
\]
Error Decomposition

\[
f^* = \arg\min_{f} \mathbb{E} \ell(f(X), Y)
\]

\[
f_{\mathcal{F}} = \arg\min_{f \in \mathcal{F}} \mathbb{E} \ell(f(X), Y)
\]

\[
\hat{f}_n = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i)
\]

- **Approximation Error** (of \(\mathcal{F}\)) = \(R(f_{\mathcal{F}}) - R(f^*)\)

- **Estimation error** (of \(\hat{f}_n\) in \(\mathcal{F}\)) = \(R(\hat{f}_n) - R(f_{\mathcal{F}})\)
Error Decomposition

Definition

The **excess risk** of \( f \) is the amount by which the risk of \( f \) exceeds the Bayes risk

\[
\text{Excess Risk}(\hat{f}_n) = R(\hat{f}_n) - R(f^*) = R(\hat{f}_n) - R(f_f^*) + R(f_f^*) - R(f^*) .
\]

This is a more general expression of the bias/variance tradeoff for mean squared error:

- Approximation error = “bias”
- Estimation error = “variance”
Approximation Error

- Approximation error is a property of the class $\mathcal{F}$
- It’s our penalty for restricting to $\mathcal{F}$ rather than considering all measurable functions
  - Approximation error is the minimum risk possible with $\mathcal{F}$ (even with infinite training data)
- $Bigger$ $\mathcal{F}$ mean $smaller$ approximation error.
**Estimation Error**

- *Estimation error*: The performance hit for choosing $f$ using finite training data
  - *Equivalently*: It’s the hit for not knowing the true risk, but only the empirical risk.
- *Smaller $F$ means smaller estimation error.*
- *Under typical conditions*: “With infinite training data, estimation error goes to zero.”
  - Infinite training data solves the *statistical* problem, which is not knowing the true risk.”
Does unlimited data solve our problems?

There’s still the *algorithmic* problem of finding $\hat{f}_n \in \mathcal{F}$.

For nice choices of loss functions and classes $\mathcal{F}$, the algorithmic problem can be solved (to any desired accuracy).

- Takes time! Is it worth it?

**Optimization error:** If $\tilde{f}_n$ is the function our optimization method returns, and $\hat{f}_n$ is the empirical risk minimizer, then the optimization error is $R(\tilde{f}_n) - R(\hat{f}_n)$.

- NOTE: May have $R(\tilde{f}_n) < R(\hat{f}_n)$, since $\hat{f}_n$ may overfit more than $\tilde{f}_n$!
ERM Overview

- Given a loss function $\ell : y \times y \to \mathbb{R}^{\geq 0}$.
- Choose hypothesis space $\mathcal{F}$.
- Use an algorithm (an optimization method) to find $\hat{f}_n \in \mathcal{F}$ minimizing the empirical risk:
  \[
  \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(X_i), Y_i).
  \]
- (So, $\hat{R}(\hat{f}) = \min_{f \in \mathcal{F}} \hat{R}(f)$).
- Data scientist’s job: choose $\mathcal{F}$ to optimally balance between approximation and estimation error.