Advanced Foundations of Machine Learning

Yash Savani
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
ysavani@cs.cmu.edu

Introduction

There are already several repositories and courses that cover the foundations of Machine Learning. Here are a few of them:

- https://github.com/jonkrohn/ML-foundations, and
- https://github.com/dontless/Machine-Learning-Foundations-A-Case-Study-Approach.

Most of these resources are designed to get you to the starting line. They provide just enough material for you to get a job doing machine learning, or for you to get started on machine learning projects. However, if you have ever tried reading a theoretical machine learning paper from a conference like COLT or one of the more advanced applied papers from a conference like ICML, it is usually clear that a lot is missing from these "foundational" curricula.

Some might argue that the best way to learn this missing material is to just dive in and start reading papers. Try to figure out the missing concepts on the fly. That is exactly what I have been doing over the past few years, and I am sad to say that this approach is very insufficient. While it may get you to a place where you can understand the research superficially, this top-down model leaves you with a deep sense of inadequacy; a feeling that you are lacking something base. With this approach, you eventually realize that you lack a cohesive narrative and many foundational concepts. You find it difficult to see the work in a wider theoretical and applied context, and have a hard time identifying nontrivial extensions. The need to go back and relearn some of the mathematical foundations then becomes imperative. This is exactly what I went through, and if you have had a similar experience, I hope this resource will be of some help to you.

If you have no experience with machine learning at all, then I would encourage you to take an intro class in machine learning to see just how cool this field

is. One example of such a course is (https://www.coursera.org/learn/machine-learning) by Andrew Ng. Machine learning is one of the most exciting fields to have blossomed over the past decade, and there are still a plethora of untapped applications to many of the existing techniques. I urge you to explore as much as you can, and also to start reading papers and thinking of extensions to contemporary work. This material should be seen more as a complement than a prerequisite to your machine learning journey. If you find yourself struggling with some of the advanced material, or you get a gnawing sense that you are missing something fundamental while reading the literature, then maybe something from this resource can help you.

For those of you who already have some experience with machine learning, I hope this resource can act as a good reference and review checklist. There are many topics and perspectives to consider any machine learning problem through. While you may be familiar with some of the ideas listed here, I hope this repository presents a few alternative viewpoints that you may have neglected to consider your problems through. Hopefully, these alternative lenses can highlight some key, deep insights into your problem that would otherwise have lied dormant. Furthermore, having a systematic checklist for review items can also help focus and prioritize future explorations. By going through the topics listed here, you can identify areas of high impact that you may not be completely comfortable with. Focusing on these areas has the potential to maximize the utility of your explorations.

My goal with this repository is to create a curated list of resources for those who want to do a deeper dive into some of the more advanced foundations of machine learning. I hope that anyone with a high school background in math and CS who goes through all the material here will successfully be able to read and carry out state-of-the-art research in both theoretical and applied machine learning feeling empowered and confident in their work.

The rest of this document is organized into several different sub-fields, with accompanying learning resources, that I think are necessary to gain this advanced foundation in machine learning.

Meta Tools

Before you start this journey, I would recommend making sure you have a solid arsenal of meta tools that will expedite your acquisition and integration of the new concepts you will be learning. These are tools like a good code and manuscript editor (I use neovim with plugins. See my dotfiles for my configuration), a good reference manager (I use Zotero, but I've heard good things about Mendeley as well), and a good concept management system (I use Zettelkasten: for an introduction to this system look at this blog post. For a deeper dive, read How to Take Smart Notes).

I also generally prefer videos over textbooks to learn new material. Unfortunately,

the speed of any video tends to vary a lot and does not necessarily match my comfort level with the material. I use the Video Speed Controller chrome extension to dial in the speed I want.

Core Math

Before diving into some of the more advanced math concepts, I would encourage you to at least cursorily go through some of the core math resources provided here. I recommend doing this even if you think you are comfortable with all the topics listed below. I can't count the number of times I thought I completely understood some core math concept, only to later realize that there was some subtlety I had neglected. These neglected subtleties can often have a cascading effect making it very hard to understand some of the more advanced material that rely on a solid core.

Problem Solving

Problems are ones that:

- Engage intellect
- Make connections to develop a coherent framework
- Can be solved in more than one way
- Should foster effective communication of mathematical ideas

Phases of problem solving:

- *Entry:* what do I know (question, experience), what do I want (paraphrase, ambiguities), what can I introduce (diagram, notation).
- *Attack:* brute force, look for patterns.
- *Review:* check, reflect, extend, understand why it works.

Propositional Logic

- Propositional Logic is the foundation for the language of reason.
- A *proposition* is a statement that has one and only one truth value, `True` or `False`.
- *Propositional variables* are letters used to stand in for actual propositional statements. They are usually p, q, or r.
- *Propositional connectives* are symbols used to connect propositional variables together.
- *Propositional expressions* are statements that link together propositional variables with potentially multiple propositional connectives.

- A *truth table* is a table that exhaustively lists all the possible truth values of the expressions.
- *Logical equivalence* (denoted by $\iff$ or $\equiv$) is when two propositional expressions are equivalent. That is, they take on the same truth value for every possible input.

Common connectives

- *Tautology or* $\top$*:* $\top(p)$ is always `True` no matter what truth value p takes. Usually when we try to prove something, we want to show that it is equivalent to a tautology, which means it is true no matter what the input truth values.
- *Negation or NOT or* $\neg$ *or* $\bar{p}$*:* $\neg p$ is `True` if and only if p is `False`.
- *Conjunction or AND or* $\wedge$*:* $p \wedge q$ is `True` if and only if p and q are both `True`.
- *Inclusive Disjunction or OR or* $\vee$*:* $p \vee q$ is `True` if and only if either p or q or both are `True`.
- *Exclusive Disjunction or XOR or* $\oplus$*:* $p \oplus q$ is `True` if p and q have different truth values.
- *Implication or Conditional or IMPLIES or* $\rightarrow$*:* $p \rightarrow q$ is `False` if and only if p is `True` and q is `False`.
  - p is called the *antecedent*, and q is called the *consequent*.
  - The *converse* of $p \rightarrow q$ is $q \rightarrow p$. It is not equivalent to $p \rightarrow q$.
  - The *inverse* of $p \rightarrow q$ is $\neg p \rightarrow \neg q$. It is not equivalent to $p \rightarrow q$.
  - The *contrapositive* of $p \rightarrow q$ is $\neg q \rightarrow \neg p$. It is equivalent to $p \rightarrow q$.
- *Biconditional or IFF or* $\leftrightarrow$*:* $p \leftrightarrow q$ is `True` if and only if both p and q have the same truth value.

Truth table for common connectives

p	q	$\neg p$	$p \wedge q$	$p \vee q$	$p \oplus q$	$p \rightarrow q$	$p \leftrightarrow q$
0	0	1	0	0	0	1	1
0	1	1	0	1	1	1	0
1	0	0	0	1	1	0	0
1	1	0	1	1	0	1	1

Order of operations

As a convention, we evaluate any propositional expression in the following order:

1. Parantheses,
2. Negation,
3. Conjunction (left to right),
4. Inclusive disjunction (left to right),
5. Implication, and
6. Biconditional.

De Morgan's theorem

One important theorem in propositional logic that appears everywhere is De Morgan's theorem. The theorem states that:

$$\neg(p \wedge q) \equiv \neg p \vee \neg q, \text{and}$$
$$\neg(p \vee q) \equiv \neg p \wedge \neg q.$$

We can prove this by showing that the truth tables for the expressions are equivalent.

Set Theory

- A *set* is a collection of members / elements.
- *Roster notation* explicitly lists out every element of the set (e.g. $A = \{1, 2, 3, 4, 5\}, B = \{2, 4, 6, 10\}$).
- *Set builder notation* implicitly describes all the elements in a set (e.g. $A = \{x \mid x \text{ is an integer between 1 and 5 inclusive}\}$, here x is a dummy variable).
- To denote that an object is an element of a set we use the symbol $\in$ (e.g. $3 \in A$). Note that this is a valid proposition.
- To denote that on object is not an element of a set we use the symbol $\notin$ (e.g. $10 \notin A$). Note that this is also a valid proposition.
- Elements cannot appear more than once in a set, though they can appear more than once in a multiset.
- A set is not ordered.
- The *universal set* is the set of all the objects being considered. It is usually notated as $\mathcal{U}$. In our case we may assume that $\mathcal{U} = \{x \mid x \text{ is an integer between 1 and 10 inclusive }\}$
- The *null set or empty set* is the set containing no elements. It is notated as $\varnothing = \{\}$.
- Set theory has operators to combine sets.

- *Venn diagrams* are a visual representation of sets and the operations on sets.

Set operators

- *Complement or* ^{c}*:* $A^c = \{x \mid \neg(x \in A)\}$ (e.g. $A^c = \{6, 7, 8, 9, 10\}$).
- *Intersection or* $\cap$*:* $A \cap B = \{x \mid x \in A \wedge x \in B\}$ (e.g. $A \cap B = \{2, 4\}$). Note that the intersection is *associative*. That is, $(A \cap B) \cap C = A \cap (B \cap C) = A \cap B \cap C$.
- *Union or* $\cup$*:* $A \cup B = \{x \mid x \in A \vee x \in B\}$ (e.g. $A \cup B = \{1, 2, 3, 4, 5, 6, 8, 10\}$). Note that the union is *associative*. That is, $(A \cup B) \cup C = A \cup (B \cup C) = A \cup B \cup C$.
- *Difference or* $\setminus$*:* $A \setminus B = \{x \mid x \in A \wedge x \notin B\}$ (e.g. $A \setminus B = \{1, 3, 5\}$).
- *Symmetric difference or* Δ*:* $A \Delta B = \{x \mid x \in A \oplus x \in B\}$ (e.g. $A \Delta B = \{1, 3, 5, 6, 8, 10\}$).

Subsets, supersets, and equality

Three other propositional statements for sets are the subset, superset, and equality relations.

- We say that a set D is a subset *subset* of a set A, if every element $x \in D$ is also an element of A. Formally, we can write this as $\forall x, x \in D \to x \in A$. It is notated as $D \subseteq A$.
- We say that a set A is a *superset* of a set D, if every element $x \in D$ is also an element of A. the superset proposition is essentially the reverse of the subset proposition. We notate this by $A \supseteq D$.
- We say that a set A is *equivalent* to a set D if it is both a superset and a subset of D. Formally, we can write this as $\forall x, x \in D \leftrightarrow x \in A$. We notate this by $A = D$.

We say that D is a strict subset of A if $D \subseteq A$, but $D \neq A$. In this case we write $D \subset A$. We can reverse the statement to get strict supersets ($A \supset D$).

A trivial statement that is always true is that $\varnothing \in C$ for all possible sets C. We say that this is true vacuously or true in an empty sort of way.

Another such trivial statement is that every set is a subset of itself since it is equal to itself.

Special sets

Some special sets that we often consider are:

- The set of *natural numbers*, $\mathbb{N} = \{1, 2, 3, 4, \ldots\}$ (some people claim $0 \in \mathbb{N}$).

- The set of *integers*, $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$.
- The set of *rational numbers*, $\mathbb{Q} = \left\{x \mid x = \frac{p}{q}, \text{ where } p, q \in \mathbb{Z}\right\}$.
- The set of *real numbers*, $\mathbb{R}$ that are the completion of $\mathbb{Q}$.
- The set of *irrational numbers* given by $\mathbb{R} \setminus \mathbb{Q}$.
- The set of *complex numbers*, $\mathbb{C} = \{x \mid x = a + ib, \text{ where } a, b \in \mathbb{R} \wedge i = \sqrt{-1}\}$.

From the definitions we can see that $\mathbb{N} \subset \mathbb{Z} \subset \mathbb{Q} \subset \mathbb{R} \subset \mathbb{C}$.

Russell's paradox, families of sets, and index sets

A set cannot contain itself. If we let this be the case then we get a paradox. The paradox comes from the set $A = \{X \mid X \notin X\}$. That is, the set that contains all sets which do not contain themselves. If A is an element of A, then A must not be in the set, but then it must be in the set. This leads to a contradiction. To avoid this kind of paradox, we accept a system of 9 axioms governing everything we can call a set and everything we can do with a set. We call this set of axioms ZFC or Zermelo–Fraenkel set theory with the axiom of choice.

We call a set comprised of other sets a family of sets. We usually denote such families of sets with a uppercase script character like $\mathcal{F}$.

Index sets are sets that are used to enumerate the elements of a set, though they don't need to be a set of numbers. They are sometimes used with families of sets to enumerate the sets contained within the family. For example, $\mathcal{F} = \{L_1, L_2, \ldots\}$, where $L_i = \{p \mid p \in \mathcal{P} \wedge p \leq i\}$, so $L_{10} = \{2, 3, 5, 7\}$. An index set might be $\mathcal{I} = \{1, 2, \ldots, 100\}$ and it can be used to index the L_i sets of $\mathcal{F}$ for $i \in \mathcal{I}$.

Predicate Logic

- Linguistically a proposition needs to be a declarative sentence with a subject, which is a noun phrase, and a predicate, which is a verb phrase.
- We can represent the subject by an element of a set and store the predicate separately (e.g. $p(x)$ = "x is an even number", where x is a free variable and p is the predicate).
- The *truth set* of p is the set of all x for which $p(x)$ is `True`. That is, the truth set of p is $\{x \mid p(x)\}$.
- *Quantifiers* tell us how big our truth set is.
- The *universal quantifier or "for all x, p(x)" or* $\forall x, p(x)$ says that $p(x)$ is `True` for every possible element of our universal set.

  - $(\forall b \in B, p(b)) \iff (\forall b, (b \in B \to p(b)))$ is a useful identity for when we want to say something about every element in a non universal set B.

- The *existential quantifier or "there exists x such that p(x)" or* $\exists x,\ s.t.\ p(x)$ says that there exists an x in the universal set such that p(x) is `True`.
  - $(\exists a \in A,\ \text{s.t. } p(a)) \iff (\exists a,\ \text{s.t. } (a \in A \wedge p(a)))$ is a useful identity for when we want to say something about some element of a non universal set A.
  - Sometimes we want to say that there exists a unique element for which $p(x)$ is `True`. In this case we use the notation $\exists! x,\ \text{s.t. } p(x)$ to say that there exists exactly one element $x \in \mathcal{U}$ that is in the truth set of $p(x)$.
- We can see that $\forall b \in B, p(b) \implies \exists b \in B,\ \text{s.t. } p(b)$ is trivially true. This tells us that any statement with a universal quantifier is a stronger than one with an existential quantifier.
- We can also combine quantifiers to get compound quantifiers.
  - $\forall x, \exists y,\ \text{s.t. } (x < y)$ is `True` for all natural numbers.
  - $\exists y,\ \text{s.t. } \forall x,\ \text{s.t. } (x < y)$ is `False` for all natural numbers since there is no largest natural number.
- We can also negate quantifiers.
  - $\neg \forall x, p(x) \iff \exists x,\ \text{s.t. } \neg p(x)$.
  - $\neg \exists x,\ \text{s.t. } p(x) \iff \forall x, \neg p(x)$.
- For a family of sets $\mathcal{F}$, we say that $\bigcup_{A \in \mathcal{F}} A = \{a | \exists A \in \mathcal{F},\ \text{s.t. } a \in A\}$
- For a family of sets $\mathcal{F}$, we say that $\bigcap_{A \in \mathcal{F}} A = \{a | \forall A \in \mathcal{F}, a \in A\}$

Proof Techniques

- Axioms/postulates, conjectures, lemmas, theorems, corollaries,
- Completeness, consistency, and decidability (Godel's incompleteness theorems),
- Valid arguments (modus ponens, modus tollens, law of syllogism),
- Invalid arguments (affirming the consequent),
- Direct proof,
- Proof by contrapositive,
- Proof by contradiction,
- Proof of conjunction,

- Proof of inclusive disjunction,
- Universal: let x be arbitrary, prove p(x),
- Existential: find an x such that p(x) is true,
- Uniqueness: assume p(x), p(y), show x = y,
- Successor operation,
- Proof by induction (weak and strong), and
- Deductive vs inductive reasoning.

Relations

- Tuples,
- Cartesian products (cartesian square, cartesian power),
- Arity of operations,
- Domain and codomain,
- Inverse relations,
- Graphs and digraphs,
- Relations on a set,
- Pre-order relations (reflexive, transitive),
- Equivalence relations (reflexive, transitive, symmetric),
- Equivalence classes,
- Partial-order relations (reflexive, transitive, antisymmetric) (posets),
- Hasse diagrams,
- Total-order relations (complete, transitive, antisymmetric),
- Strict-order relations (reflexive -> irreflexive) (trichotomous),
- Upper bounds, lower bounds, maximum, minimum, and
- Least upper bound (supremum), greatest lower bound (infimum).

Functions

- Left-total, left-unique, right-total, right-unique,
- Function (left-total and right-unique),
- Prototype and definition,
- Images, ranges, and preimages,

- Injective/one-to-one (left-unique),
- Surjective/onto (right-total),
- Bijective/one-to-one correspondence (right-total and left-unique),
- Inverse functions,
- Function composition,
- Restricing domain and codomain, and
- Monotonicity.

Number Systems

- Cardinality,
- Pigeon-hole principle,
- Cantor-Bernstein-Schroder theorem,
- Finite Sets,
- Peano's axioms,
- Countable sets (Aleph-Null),
- Integers,
- Rationals (snaking argument),
- Arithmetic operations,
  - Addition with inverse (subtraction),
  - Multiplication (repeated addition) with inverse (division),
- Exponentiation (repeated multiplication) with inverses (logarithms and radicals/roots since exponentiation is not commutative),
- Algebraic numbers,
- Real Numbers
  - Zenos' paradoxes,
  - Transcendental numbers,
  - Dedekind cut,
  - Supremum property, and
  - Cantor's diagonalization argument,
- Cardinality of the continuum (continuum hypothesis),
- Hilbert's paradox of the Grand Hotel,

- Complex numbers (closure under roots) (no ordering),
  - Complex arithmetic,
  - Conjugate,
  - Modulus,
  - Argand diagram, and
- Quarternions (not commutative) and octonions (not associative).

Basic Number Theory

- Divisibility,
- Multiples and factors,
- Euclidean division (dividend = quotient divisor + remainder),
- Greatest common divisor, and lowest common multiple,
- Relative primes,
- Euclid's algorithm,
- Linear combination,
- Diophantine equations,
- Prime numbers,
- Unique prime factorization,
- Euler totient function,
- Prime counting function,
- Sieve of Eratosthenes,
- Mersenne primes,
- Mill's theorem, and
- Riemann Hypothesis.

Modular Arithmetic

- Remainders follow cyclic pattern,
- Congruence modulo m,
- Properties of congruence,
- Congruence classes,
- Cancelling when dividing by relative prime of modulus,

- Divisibility tests, and
- Multiplicative inverses.

Birkhoff Geometry (geometry based on real numbers)

- Undefined terms
  - point,
  - line (set of points),
  - distance (positive definite, symmetric, triangle inequality), and
  - angle (formed by 3 points).
- Definitions
  - Between,
  - Line segment,
  - Half-line/ray and endpoint,
  - Parallel,
  - Straight angle, right-angle, perpendicular,
  - Triangle, vertices, degenerate triangle, and
  - Similar and congruent.
- Postulates
  - Bijection between points of a line and real numbers,
  - Unique line containing two distinct points,
  - Bijection between rays and real numbers mod 2 pi, and
  - Triangle congruences.
- Theorems
  - Linear pair,
  - Vertical angles,
  - Triangle angles sum,
  - Corresponding angles (and converse),
  - Alternate interior/exterior angles (and converses),
  - Interiors on the same side (and converse),
  - Perimeter and Area,
  - Parallelogram,

  - Rectangle,
  - Rhombus,
  - Square,
  - Trapezoid,
  - Polygons (regular and irregular),
  - Circles (radius and diameter, chords, tangents, arcs, angles),
- 3D cylinders and volume,
- Planes and disks,
- Spheres,
- Polyhedrons are 3D object with flat faces and edges, and
- Dihedrons are a 2D object in 3D space with two faces and edges of neglegible thickness.
- Trigonometry
  - Right-angle triangle (opposite, adjacent, hypotenuse),
  - Sine, cosine, tangent,
  - Reciprocals (cosecant, secant, cotangent),
  - Inverses (arcsine, arccosine, arctangent)
  - Tan = sin / cos,
  - Unit circle,
  - Radians vs degrees,
  - Domains and ranges,
  - Periods,
  - Pythagorean identities ($\sin^2 + \cos^2 = 1$)
  - Sine is odd, cos is even,
  - Sum and difference formulae (cos together and flip, sin seperate and keep),
  - Product to sum and sum to product formulae,
  - Convert sine to cosine by phase shift,
  - Relation to complex numbers ($e^{ix} = \cos(x) + i \cdot \sin(x)$),
  - Add complex numbers using parallelogram rule, and
  - Multiply complex numbers by adding the angles and multiplying the lengths.

Systems of Equations, Matrices, Polynomials

- Algebraic expressions (rational exponents),
- Linear equations and inequalities,
- Linear graphs,
- Critical points,
- Multiple equations with multiple unknowns,
- Writing linear systems of equations as matrices,
- Matrix multiplication,
- Matrix inverses (singular matrices),
- Absolute value and piecewise equations,
- Monomials,
- Polynomials,
- Quadratic equation,
- Cubic equation,
- No closed form for quartic equation (Galois),
- Binomials,
- Binomial theorem (permutations and combinations) (Pascal's triangle),
  - $(a-b)^2 = (a-b)(a+b)$,
- Foil,
- Polynomial arithmetic,
- Polynomial factorization,
- Polynomial division,
- Partial fractions (Partial fraction decomposition),
- Polynomial graphs,
- Reciprocal graph,
- Complex polynomials, and
- Fundamental theorem of algebra (any polynomial of degree d has d, possibly complex, roots) (connects algebra with geometry).

Group Theory

- Group (closed, associative, unique identity, unique inverse),
- Idempotence,
- Order of a group is the cardinality of the group,
- Finite and infinite groups,
- GLnR group of nonsingular matrices under matrix multiplication,
- Abelian groups (commutative group),
- Cayley table,
- Subgroup (show closure and inverse for infinite group, only closure for finite group),
- Exponentiation,
- Generated group,
- Order of an element,
- Cyclic group,
- Dihedral group (symmetries of a regular polygon), and
- Symmetric/permutation group (bijections from a set to itself) (transpositions, cycles, compostion).

Rings and Fields

- Rings are additive abelian groups that are closed under multiplication,
  - Addition and multiplication are linked through the distributive property,
- Exponentiation is repeated multiplication,
- 0a = a0 = 0,
- a (-b) = (-a) b = -(ab),
- -(-a)(-b) = ab,
- Infinite and finite rings,
- Polynomial rings,
- Commutative rings,
- Unital ring (multiplicative identity) (inverses and units),
  - 1 = 0 implies the ring is the trivial ring,
- Division ring (unital ring where every non-zero element is a unit),

- Zero divisor (non-zero element times another non-zero element gives 0),
- Intersection of units and zero divisors is empty,
- Integral domain (commutative, unital ring with no zero divisors),
- Can cancel factors from both sides of an integral domain,
- A field is an algebraic structure that is abelian under both addition and multiplication,
  - A field is an integral domain where every non-zero element is a unit,
- Rationals, reals, and complex numbers are infinite fields,
- Integers modulo a prime are finite fields, and
- All arithmetic operations $+, -, \times$, and $\div$ play nicely together.

Morphisms

- Symmetry is patterned self-similarity,
- Dihedral group D_n is a non-abelian, non-cyclic group with order $2n$,
- Homomorphisms (operation preserving functions),
- Isomorphisms (bijective homomorphisms) (equivalence relation),
- Endomorphisms and automorphisms (isomorphic endomorphism),
- Automorphism group,
- Group action,
- Any finite cyclic group of order m is isomorphic to the integers mod m.
- Cayley's theorem (every group is isomorphic to a subgroup of the symmetric group).
- Every permutation can be expressed as a composition of transpositions, and
- Every permutation has a fixed parity (odd or even based on number of transpositions).

Single Variable Calculus

- Zenos' paradoxes,
- Sequences (notation, inductive and recursive definition),
- Limits (intuitive definition),
  - Limits at infinity and infinite limits,
  - One sided limits,

- Convergence and divergence (periodic),
- Squeeze theorem,
- Properties of limits,
  - Constant scaling,
  - Closure under arithmetic operations ($+, -, \times, \div$, powers, and roots),
- Limit evaluation,
- Continuous functions and closure of contiuity under composition,
- Intermediate value theorem,
- Derivative,
  - Interpretation as relative changes in variables,
- L'Hopital's rule,
- Polynomial derivative,
- Basic properties of derivatives,
  - Constant derivative is zero,
  - Constant scaling,
  - Addition, subtraction,
  - Product rule,
  - Quotient rule,
  - Power rule,
  - Chain rule,
- Exponential derivative and Euler's number,
- Higher order derivatives (smooth functions),
- Implicit differentiation (logarthmic derivative),
- Common derivatives,
  - c, x, sin, cos, tan, arcsin, arccos, arctan, eˆx, aˆx, ln(x),
- Optimization (constrained and unconstrained),
- Critical points and extrema (local and global),
- Convex and concave functions,
- Extreme value theorem,
- 1st and 2nd derivative test,
- Lagrange multipliers,

- Newton's method,
- Mean value theorem,
- Riemann integral and anti-derivative,
- Fundamental theorem of calculus,
- Definite vs indefinite integral,
- Properties of anti-derivative,
  - Constant scaling,
  - Addition and subtraction,
- Common integrals,
  - k, polynomials, 1/x, ln(x), e^x, sin, cos, tan
- Standard integral techniques,
  - Substitution,
  - Integration by parts,
- Partial sums and infinite series,
  - Absolutely convergence,
  - Conditionally convergence,
- Arithmetic series (average of the terms times the number of terms),
- Geometric series (infinite limit = a/(1-r)),
- Telescoping series,
- Harmonic series (infinitely divergent) and p-series,
- Convergence tests,
  - Integral test,
  - (Limit) comparison test,
  - Alternating series test,
  - Ratio test,
  - Root test,
  - Power series (radius of convergence), and
- Taylor series,
  - 1/(1-x) = sum x^i,
  - e^x = sum x^i / i!,
  - cos x = sum (-1)^n x^{2n}/(2n)!,

- sin x = sum (-1)^n x^{2n+1}/(2n+1)!.

One excellent resource I have found for some of this material is the playlist Introduction to Higher Mathematics. I recommend going through this playlist at least once.

A crucial tool in the machine learning toolbox is calculus. Most of the machine learning literature makes extensive use of multivariable calculus. However, before diving into the nuances of multivariable calculus, it's important to make sure you have a solid foundation and intuition for single variable calculus first. To gain the intuition I would recommend a this playlist by 3Blue1Brown, or this course on AP calculus from Khan Academy. If you already have a strong calculus foundation, you can safely skip this section. Though, I would still encourage you to go through the essence of calculus playlist by 3Blue1Brown at some point though. It helps fill in a lot of intuition that more formal classes may have missed. Who knows? You may learn something you didn't realize you had missed. Even small insights in the foundations can lead to major revelations later on.

Core CS

Many of the recent innovations in machine learning can be traced directly back to computer science roots. Furthermore, much of the language used in the contemporary machine learning literature is inherited almost entirely from the computer science literature. Also, ultimately most machine learning ideas will have to be transcribed into code that will run on computers. Therefore, having a solid foundation in computer science will be imperitave to your success in machine learning research. All the topics covered here should be part of any introductory CS curriculum.

Programming basics

- Types,
- Variables,
- Arrays and strings,
- Conditionals,
- Loops,
- Functions,
- Bits and bitwise operations,
- Recursion and backtracking,
- Classes and objects,
- Functional vs object oriented paradigms, and

- Pointers/references and memory organization (stack and heap).

Algorithms

- Search,
- Sort,
- Divide and Conquer,
- Bachmann-Landau/asymptotic notation (Big/Little O, Omega, Theta),
- Master theorem,
- Randomized algorithms,
- Graphs/networks,
- Minimum spanning trees,
- Balancing trees,
- Depth first search and breadth first search,
- Dijkstra,
- Dynamic programming (Bellman Ford, Floyd Warshall, Knapsack),
- Min-cut and max-flow (Ford-Fulkerson), and
- P vs NP (NP-completeness, 3 SAT, reductions).

Data Structures

- Lists,
- Stacks,
- Queues (priority queues),
- Trees,
- Heaps,
- Hashmaps (amortization), and
- Sets.

Systems and Additional Topics

- Hardware overview (transistors, logic gates, latches, memory, CPU, RAM, GPU, peripherals),
- Unix (filesystem, interrupts/signals, system calls, processess, interprocess communication, terminal, shell scripting),

- Multiprocessing vs multithreading, and asynchronous processing (mutexes, semaphores, threadpools, race conditions, deadlocks),
- Networking (TCP/IP, sockets, HTTP, HTML5, CSS3, bandwidth, latency, throughput),
- Standard libraries (string manipulation, file I/O, datetime, basic arithmetic and math operations),
- Regular expressions,
- Debugging techniques, and
- Profiling.

I learned most of this material through various online and in person classes over 13 years ago. As a result, most of the sources I used have since become antiquated and have been replaced with several better and friendlier versions. I am happy to add more resources to this section based on recommendations.

For anyone who has never programmed before, I recommend going through Programming Methodology by Prof. Mehran Sahami. While this playlist is on the older side, I think Prof. Sahami does a fantastic job of laying out the basics of programming in this course. This can safely be skipped if you already have some experience with programming.

Once you have whet your appetite for programming and basic computer science, CS106b is a good segue into the world of algorithms and abstractions. If you are already familiar with most of the basic algorithms and abstractions listed above feel free to skip this course as well. If you think your algorithms and abstraction skills may have gotten a little rusty take a look at the slides from the version of the course taught by Keith Schwarz.

Now that you have had some experience with programming and learning how to abstract some of the basic ideas from code into general algorithms, it's important to learn some of the most fundamental algorithms and algorthimic analysis techniques in computer science. To cover this material there are several fantastic resources. - CS161 by Prof. Tim Roughgarden is a great introduction to many of the most important algorithmic techniques you will need to know. - An equivalent alternative to the CS161 lectures, is a multi-part course by Prof. Roughgarden on Coursera split into part 1 (divide and conquer), part 2 (graph search, shortest paths), part 3 (greedy algorithms), and part 4 (NP-completeness) also taught by Prof. Roughgarden. If you prefer, playlist 1 and playlist 2 are the corresponding YouTube playlists. I recommend this option over the CS161 lectures because of the audio quality. - A course series on introductory algorithms from MIT that covers a little more than CS161 (6.006 and 6.046). I preferred the dynamic programming lectures in this series.

Regardless of whether you choose to pursue theoretical or applied research, you will inevitably come across some literature that requires you to have a working systems background. The modern revolution in machine learning was at least in

some part a consequence of the incredible advances we have made in hardware technology. Understanding the relationship between hardware and algorithms therefore becomes critical to an appreciation of modern machine learning.

For this basic foundation in systems, I would recommend CS107 (for an updated version see this playlist) and CS110. If you already feel comfortable with systems but want a recap, I would recommend going through this short series by Brian Will: - Hardware Basics Video, - Operating Systems Video, - Unix System Calls 1, Unix System Calls 2, and - This playlist on Unix terminals and shells.

Intermediate Math

Now we come to some of the more advanced topics in math that will almost definitely be used either directly or indirectly in any contemporary machine learning research you may come across.

Linear Algebra

- Vector space (V,+, $\cdot$) over real or complex field (CANI-ADDU),
  - A vector is an element of a vector space,
  - Intuitive vector space is euclidean space (isomorphic to cartesian power of real numbers) with vectors as arrows, sum using parallelogram rule (element-wise sum), and s-multiplication by scaling arrow (scale all elements),
  - Non-intuitive abstract vector space is the set of polynomials of fixed degree with natural addition and scaling,
- Vector subspace,
- Linear combinations,
- Linear maps,
  - f(av + bw) = af(v) + bf(w),
  - Homomorphism between vector spaces,
  - Intuitively, keeps grid lines parallel and evenly spaced. Also doesn't move origin, (rotation, reflection, scaling, skewing),
- Compositions of linear maps are also linear maps,
- Set of linear maps between two vector spaces forms a vector space,
- Dual vector space to vector space V is the set of linear maps from V to the real number line,
  - A covector is an element of the dual vector space,

  - The dual vector space to the dual vector space to V is V when V is finite dimensional,
- Inner products (linearity in first argument, conjugate symmetry, positive definiteness),
  - Cauchy-Schwarz inequality,
  - Projection,
  - Angle definition,
  - Orthogonal vectors,
  - Induced norm (absolute homogeneity, positive definiteness, triangle inequality) -> induced metric/distance,
- Hamel-basis (finite dimensional basis),
  - Can represent a vector in uncountably many ways,
  - Linear independence (can form one of the vectors using linear combinations of the others),
  - Linear span (set of all linear combinations of the vectors),
  - Hamel-basis is a finite subset of the vector space that is linearly independent and spans the entire space,
  - The dimension of a vector space V is given by the cardinality of a basis on V (well-defined),
  - The dual basis is a set of covectors where applying the the ith dual basis element to the ith basis element gives 1, otherwise 0,
  - Standard/canonical basis for euclidean space,
- Matrices are linear maps expressed in a grid form based on the chosen basis,
  - Square and rectangle matrices,
  - Zero matrix,
  - Identity matrix (kronecker delta),
- Geometry of matrices (based on euclidean space),
- Systems of linear equations,
- Matrix transpose (AB)ˆT = BˆT AˆT,
- Vectors and covectors as matrices,
- Column space (vector space) and row space (dual space),
- Euclidean inner/dot product <v,w> = vˆT w,

  - Induced euclidean/l2 norm,
  - Induced euclidean/l2 distance,
- Vector norms
  - lp norms,
  - l1 norm, l_{} norm,
  - Unit ball,
  - Equivalence of vector norms (all norms are equivalent in euclidean space),
- Matrix multiplication (composition of linear maps),
  - Matrix-vector product as linear combination of row/column space,
  - Associativity and distributivity,
  - Non-commutativity,
- Rank and null space of a matrix,
- Determinant (volume of a parallelepiped),
- Matrix inverse,
- Cramer's rule,
- Gaussian elimination,
  - Diagonal matrices,
  - Elimination matrices,
  - Permutation matrices,
  - Triangular matrices,
  - LU factorization (with pivots QLUP),
- Fundamental theorem of linear algebra,
- Over and under determined systems of equations,
- Linear least squares (normal equations),
  - Gram matrix,
  - Symmetric matrices (antisymmetric matrices),
  - Quadratic forms,
  - Positive/negative (semi)-definite matrices,
- Trace of a matrix,
- Cholesky decomposition of a symmetric, positive definite matrix,

- Orthonormal/orthogonal matrices (Gram-Schmidt),
- Change of basis by matrix multiplication,
- Eigenvalues and Eigenvectors,
  - Characteristic polynomial,
  - Linear independence of eigenvectors,
  - Left and right eigenvectors,
  - Variational approach,
  - Rayleigh quotient,
  - Trace is sum of eigenvalues,
  - Determinant is product of eigenvalues,
- Spectrum and spectral radius,
- Spectral theorem (hermitian matrices),
  - Real eigenvalues,
  - Orthogonal eigenvectors,
- Non-defective or diagonalizable matrices,
- Similar matrices,
- Eigendecomposition,
- Singular value decomposition,
  - Covariance matrix,
  - Singular values,
  - Left and right singular vectors,
  - Moore-penrose pseudoinverse,
  - weighted sum of outer products,
  - truncated SVD,
- Matrix norms,
  - Submultiplicativity,
  - Frobenius norm (l2 norm of singular values),
  - Induced/operator matrix-p norms,
  - Spectral/matrix-2 norm (l_{} norm of singular values),
  - Matrix-1 norm (maximum l1 norm of the columns),
  - Matrix-norm (maximum l1 norm of the rows),

    - Nuclear/trace norm (l1 norm of singular values),
- Linear dimensionality reduction (principal components analysis),
- (r,s)-Tensors (multilinear [linear in every entry] map from cartesian product of r covectors and s vectors to reals),
    - Vectors are (1,0)-tensors for finite dimensional vector spaces,
    - Covectors are (0,1)-tensors,
    - Linear maps (matrices) are (1,1)-tensors,
    - Inner products (also matrices) are (0,2)-tensors,
- Components/coordinates of tensors,
    - Plug in the basis and dual basis elements to get all components/coordinates of the tensor,
- Kronecker/tensor product (relation to outer product), and
- Einstein notation (upper is up to down, lower is left to right).

Real Analysis

TODO @yashsavani: Add details

Multivariable Calculus Using Analysis

TODO @yashsavani: Add details

Probability Theory (Measure Theory and Stochastic Processes)

TODO @yashsavani: Add details

Statistics

TODO @yashsavani: Add details

Linear algebra is the foundation of almost all the math needed for machine learning. As my advisor says, "you can't ever know enough linear algebra". To develop a strong intuition for this material, I would first recommend going through this excellent playlist by 3Blue1Brown (Grant Sanderson). If you have never seen anything from his channel, you are in for a treat! Grant's expositions on math concepts are mesmerizing. Once you have developed some of the intuition for linear algebra, I would recommend going through this playlist by the esteemed Prof. Gilbert Strang from MIT for a more thorough coverage of the material.

Once you have a strong core, one of the most important math topics to cover is real analysis and multivariable calculus. Most of the theoretical foundation in machine learning relies on analyzing the convergence of processes in the limit. Real analysis is a tough topic, so it helps to have multiple sources to refer to if you get stuck. Some of the resources I can recommend are - Analysis II by the incredible Prof. Terrence Tao. - A playlist on Real Analysis by The Bright Side of Mathematics. - This video on the Jacobian. - This course on vector calculus - Another course on vector calculus. - Math 131 at Harvey Mudd College.

For a course in measure theory, with a specific interest in probability theory I recommend these notes by Prof. Amire Dembo. Another resource is this playlist on probability theory by The Bright Side of Mathematics. Yet another great resource is a Probability Primer by the mathematical monk.

For a background in statistics I recommend the 36-705 lecture notes By Prof. Sivaraman Balakrishnan.

These are some resources I have been recommended, but not had the time to go through and review myself. - https://www.youtube.com/playlist?list=PL05umP7R6ij1a6KdEy8PVE9zoCv6SlHR
- https://www.youtube.com/playlist?list=PL05umP7R6ij0Gw5SLIrOA1dMYScCx4oXT
- https://www.youtube.com/playlist?list=PLwJRxp3blEvaxmHgI2iOzNP6KGLSyd4dz

Numerical Methods and Optimization

- Matrix factorization (LU, Cholesky, QR, SVD),
- Eigenproblems (power iteration, shifting, deflation, QR Iteration, Krylov subspace methods),
- Fast Fourier Transform,
- Varational methods and normal equations,
- Sensitivity analysis,
- Fixed-point iteration,
- Newton's method,
- BFGS,
- Automatic differentiation,
- Gradient descent,
- Conjugate gradient descent,
- Proximal gradient descent,
- Stochastic gradient descent,
- Convexity,

- Duality,
- Karush Kuhn Tucker conditions,
- Convex problems (LP, QP, SDP, SOCP),
- Momentum,
- Frank Wolfe methods, and
- Mirror descent.

Here are some resources

- Numerical Algorithms playlist with accompanying textbook
- Convex optimization course.
- Linear algebra playlist
- Scientific computing course
- Numerical Linear Algebra online course.
- Convex optimization summer school

Core ML

- Perceptron (XOR),
- SVM (Slater),
- Kernels (Representer, Mercer),
- Realizable PAC,
- VC dimension,
- Sauer's lemma,
- No Free Lunch Theorem,
- Agnostic PAC,
- Rademacher Complexity,
- Fundamental theorem of statistical learning,
- Bias-variance tradeoff,
- double descent,
- Approximation algorithms,
- Clustering,
- Neural networks,

- Ensemble methods (bagging, boosting),
- Probabilistic graphical models,
- Multi-armed bandits, and
- Markov decision processes.

Note: no reinforcement learning here.

- The standard textbook Understanding Machine Learning From Theory to Algorithms by Prof. Shai Shalev-Shwartz and Prof. Shai Ben-David and the accompanying lecture playlist
- An overview of all the material and a good review if you are already familiar with these topics EPFL playlist.
- A course in statistical machine learning.
- A more classical approach of machine learning theory.

Recommended but not personally reviewed. - https://www.youtube.com/playlist?list=PLAPSKVSdi0oac6hwCkl
- https://www.youtube.com/playlist?list=PLTPQEx-31JXhguCush5J7OGnEORofoCW9

Seminal Research in ML

- Computer vision
  - classification,
  - segmentation,
  - OCR,
  - optical flow, and
  - reconstruction
- Natural language processing
  - syntax,
  - speech,
  - semantics,
  - translation,
  - QA,
  - retrieval
- Neural networks as universal function approximators,
- Backpropagation,
- batch normalization,

- Dropout,
- Momentum,
- NTK,
- CNN,
- ResNet,
- LSTM,
- Language models,
- Transformers,
- VAE,
- WGAN,
- NAS.

Some resources are: - The Deep Learning Book. - Repository of seminal deep learning papers - NYU Deep Learning course. - Foundations of Deep Learning course from University of Maryland, College Park.

Tools

- Python or Julia with conda,
- Numerical libraries (numpy, PyTorch, JAX),
- Data tools (SQL, NoSQL, redis, pandas, spark, S3),
- System tools (ssh, MPI, docker, kubernetes),
- Visualization tools (matplotlib, seaborn, PCA, t-SNE, UMAP),
- statistical analysis tools (scipy, statsmodel), and
- Notebooks (Jupyter, Pluto).

Some resources are: - Introduction to Computational Thinking A course on Julia at MIT.

Additional Topics

- Optimal transport,
- Algebraic/differential topology/geometry,
- Information theory,
- Game theory,

- Statistical mechanics,
- Psychology,
- Cognitive science,
- Graphics,
- Hardware architecture,
- Automata theory,
- Type theory,
- Category theory,
- Computational Fluid Dynamics (a visual masterpiece) and also introduces some important ideas about information reduction for computational feasibility.
- Intution for General Relativity.
- Breadboard computer.
- Advanced Algorithms Stanford CS261 by Prof. Tim Roughgarden.
- Advanced Algorithms Harvard COMPSCI 224 by Prof. Jelani Nelson.
- General Relativity and more generally differential topology.
- Engineering Mathematics.
- Functional Analysis.
- Five Miracles of Mirror Descent.

Acknowledgements

I want to thank Josh Cho for giving me advice on this repository.