# STAT 6900: Stochastic Processes in Modern Machine Learning
# Lecture 1: Course Vision

**Farzad Sabzikar**

Office: 2218 Snedecor Hall sabzikar@iastate.edu

January 20, 2026

# Course Vision

## Statistics/Probability → Modern ML

We use **stochastic processes** and **stochastic differential equations (SDEs)** as a **unifying mathematical framework** to understand selected modern machine learning topics such as **stochastic optimization**, **reinforcement learning (RL)**, and **continuous-time deep learning**.

## Important clarification

This course is not intended to be a full treatment of stochastic optimization, reinforcement learning, or continuous-time deep learning. Each of these topics could easily fill an entire semester on its own. Instead, our goal is to focus on the **stochastic-process foundations** that connect and unify these areas, and to develop a common mathematical language that helps us understand why modern learning algorithms behave as they do.

# Stochastic Gradient Descent (SGD): What Is It?

## Basic update rule

Stochastic Gradient Descent generates a sequence of parameter values $\theta_0, \theta_1, \theta_2, \ldots$ according to the recursion

$$\theta_{k+1} = \theta_k - \eta_k \nabla f(\theta_k) + \eta_k \, \xi_{k+1},$$

where

- $\theta_k$ is the current parameter value at iteration $k$.
- $f(\theta)$ is the loss function we want to minimize.
- $\nabla f(\theta_k)$ indicates the direction in which the loss function increases most rapidly near $\theta_k$.
- $\eta_k$ is the learning rate, controlling how large each update step is.
- In practice, $\nabla f(\theta_k)$ is not computed exactly. It is estimated using random data (mini-batches, subsamples, random transitions).
- The term $\xi_{k+1}$ captures this random error, called *stochastic gradient noise*.

# Why Study SGD? Goals and Approach

## Why SGD matters

- SGD is the main tool used to train most modern machine learning models. For example, large neural networks are trained by repeatedly applying SGD or its variants to update millions of parameters.
- Despite its simplicity, SGD works well even in very high-dimensional and highly nonconvex problems.
- In practice, SGD often converges quickly and finds good solutions.

## Our goal and approach

- Understand SGD as a stochastic process evolving over time.
- Study stability, long-time behavior, and escape from unfavorable regions.
- Use continuous-time models (ODEs and SDEs) to gain insight.

# Reinforcement Learning (RL): What Is It?

## Basic setup

Reinforcement Learning studies decision-making over time in uncertain environments. It is described by:

- a **state process** $X_t$, which represents the current situation of the system (for example, position, velocity, or system status),
- a **control or policy** $\pi$, which specifies what action to take in each state,
- a **cost function** $c(x, a)$, which measures how undesirable a state–action pair is.
- The goal is to choose a policy $\pi$ that minimizes the expected total cost

$$J(\pi) = \mathbf{E}\left[\int_0^T c(X_t, \pi_t)\, dt + g(X_T)\right],$$

where $g(X_T)$ represents a final cost at the terminal time.

# Why Study Reinforcement Learning?

## Why RL matters

- RL models situations where decisions must be made step by step while outcomes are uncertain.
- Examples include controlling a robot as it moves, managing a portfolio over time, or deciding how much inventory to keep when demand is random.
- In all cases, actions affect the future state of the system.

## Our goal and approach

- Study **value functions**, which describe how good or bad a situation is when we act optimally.
- Use the idea of **dynamic programming**: solving a long problem by breaking it into smaller decisions over time.
- Understand **Hamilton–Jacobi–Bellman equations** as equations that describe how optimal decisions evolve continuously over time.

# Neural ODEs: What Are They?

## Basic idea

Neural ODEs describe how a system changes **continuously over time**:

$$\frac{dX(t)}{dt} = f_\theta(X(t), t), \qquad X(0) = x.$$

Instead of applying many separate layer-by-layer updates, the model describes how the state changes smoothly over time.

## Interpretation

- Neural ODEs describe deep networks by treating many small layers as one smooth transformation.
- The parameters $\theta$ tell the system how to move at each moment.
- Adding randomness leads naturally to **Neural SDEs**, which model noise.

# Why Study Neural ODEs? Goals and Approach

## Why Neural ODEs matter

- Deep learning uses many discrete layers, while dynamical systems describe smooth change over time.
- Neural ODEs connect these ideas by viewing a deep network as a continuously evolving system.
- This allows us to use tools from dynamical systems to understand deep learning models.

## Our goal and approach

- Understand how Neural ODE models behave over time, such as whether their outputs are stable or sensitive to small changes.
- Study how randomness and noise influence learning and prediction.
- View training and prediction as continuous-time processes, which can simplify analysis and provide clearer insight.

# Background Assumptions and Starting Point

## Assumed background

Students are expected to be familiar with:

- basic probability ideas such as random variables and expectation,
- **conditional expectation**, used to reason about future outcomes given current information,
- **filtrations**, which describe how information is revealed over time,
- basic **Markov chains and Markov processes**, where the future depends mainly on the current state.

## What we will build in this course

- **Brownian motion** as a basic model of randomness.
- **stochastic differential equations (SDEs)** as models for random evolution in time.
- Applications of these ideas to **SGD, reinforcement learning, and Neural ODE/SDE models**.

# Minimal Stochastic-Process Language

## Definition 1.1 Stochastic process

A **stochastic process** is a collection of random variables $\{X_t : t \in \mathcal{T}\}$ defined on a probability space $(\Omega, \mathcal{F}, \mathbf{P})$, where $\mathcal{T}$ is an index set (typically time, e.g. $\mathbb{N}$ or $\mathbb{R}_+$).

## Definition 1.2 Filtration and adaptedness

A **filtration** $\{\mathcal{F}_t\}$ is an increasing family of $\sigma$-algebras. A process $\{X_t\}$ is **adapted** to $\{\mathcal{F}_t\}$ if $X_t$ is $\mathcal{F}_t$-measurable for each $t$.

## Interpretation

In learning algorithms, $\mathcal{F}_t$ typically represents data and randomness observed up to iteration $t$.

# Why Filtrations Matter for Learning Algorithms

## Sequential learning and information over time

Learning algorithms such as SGD and reinforcement learning generate a sequence

$$\theta_0 \to \theta_1 \to \theta_2 \to \cdots$$

where $\theta_k$ represents the state of the algorithm at step $k$ (for example, model parameters or estimates).

- Randomness (data sampling, noise, random transitions) is revealed gradually over time.
- $\mathcal{F}_k$ represents everything the algorithm knows up to step $k$.
- Algorithm updates are **non-anticipative**: the next update $\theta_{k+1}$ uses only past and present information, not future data.

# Gradient Descent as Deterministic Dynamics

## Gradient descent as a dynamical system

Let $f : \mathbb{R}^d \to \mathbb{R}$ be a differentiable function. We wish to minimize $f(\theta)$ with respect to the parameter vector

$$\theta = (\theta_1, \ldots, \theta_d) \in \mathbb{R}^d,$$

and we assume $f$ is smooth enough so that its gradient exists.
For a step size (learning rate) $\eta > 0$, gradient descent generates a sequence

$$\theta_{k+1} = \theta_k - \eta \nabla f(\theta_k),$$

where the gradient is defined by

$$\nabla f(\theta) = \left( \frac{\partial f}{\partial \theta_1}(\theta), \ldots, \frac{\partial f}{\partial \theta_d}(\theta) \right),$$

which points in the direction of steepest increase of $f$ at $\theta$.
Once $\theta_0$ and $\eta$ are fixed, the entire path $\{\theta_k\}_{k \geq 0}$ is fully determined.

# Stochastic Gradient Descent (SGD)

## From gradient descent to stochastic gradient descent

Gradient descent updates parameters using the full gradient:

$$\theta_{k+1} = \theta_k - \eta \nabla f(\theta_k).$$

In many problems, the objective can be written as an average:

$$f(\theta) = \mathbf{E}[\ell(\theta; Z)] \quad \text{or} \quad f(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(\theta; Z_i),$$

where $Z$ denotes a generic random element (data, transition, or sample) and $\ell(\theta; Z)$ is the corresponding loss. At iteration $k$, we sample a mini-batch $B_k = \{Z_{k,1}, \ldots, Z_{k,m}\}$ with $m < n$ and define

$$\nabla \ell_{B_k}(\theta) = \frac{1}{m} \sum_{j=1}^{m} \nabla \ell(\theta; Z_{k,j}).$$

# Stochastic Gradient Descent (SGD)

## Stochastic Gradient Descent (SGD)

Stochastic gradient descent updates parameters as

$$\theta_{k+1} = \theta_k - \eta \nabla \ell_{B_k}(\theta_k).$$

Equivalently,

$$\theta_{k+1} = \theta_k - \eta \nabla f(\theta_k) + \eta \, \xi_{k+1}, \qquad \xi_{k+1} = \nabla f(\theta_k) - \nabla \ell_{B_k}(\theta_k),$$

showing that **SGD is gradient descent plus noise**.

# Stochastic Gradient Noise

## Stochastic Gradient Noise

Recall that stochastic gradient descent updates parameters as

$$\theta_{k+1} = \theta_k - \eta \nabla \ell_{B_k}(\theta_k),$$

where $\nabla \ell_{B_k}(\theta_k)$ is the mini-batch gradient. Since the mini-batch $B_k$ is random, $\nabla \ell_{B_k}(\theta_k)$ is a random approximation of the full gradient $\nabla f(\theta_k)$. We define the **stochastic gradient noise** by

$$\xi_{k+1} = \nabla f(\theta_k) - \nabla \ell_{B_k}(\theta_k).$$

With this notation, the SGD update can be written as

$$\theta_{k+1} = \theta_k - \eta \nabla f(\theta_k) + \eta\, \xi_{k+1},$$

showing that stochastic gradient descent is gradient descent plus a noise term.

# Why Continuous-Time Models?

## Small step sizes and continuous-time limits

Learning algorithms such as gradient descent and SGD update parameters in small steps:

$$\theta_{k+1} - \theta_k = -\eta \nabla f(\theta_k) \quad \text{(GD)},$$

$$\theta_{k+1} - \theta_k = -\eta \nabla f(\theta_k) + \eta\, \xi_{k+1} \quad \text{(SGD)}.$$

When the step size $\eta$ is small, the algorithm takes many small steps. This suggests describing the evolution of $\theta_k$ by a **continuous-time limit**.

Continuous-time models allow us to:

- approximate discrete algorithms by ODEs and SDEs,
- study stability, long-time behavior, and the effect of noise,
- gain intuition that is hard to see in discrete time.

# Optimization Terminology: Minima and Saddle Points

## Standard definitions

Let $f : \mathbb{R}^d \to \mathbb{R}$ be a **differentiable** function.

- A point $\theta^\star \in \mathbb{R}^d$ is a **global minimizer** if

$$f(\theta^\star) \leq f(\theta) \quad \text{for all } \theta \in \mathbb{R}^d.$$

- A point $\theta^\star \in \mathbb{R}^d$ is a **local minimizer** if there exists $r > 0$ such that

$$f(\theta^\star) \leq f(\theta) \quad \text{for all } \theta \text{ with } \|\theta - \theta^\star\| < r.$$

  Here $\|\cdot\|$ denotes the Euclidean norm on $\mathbb{R}^d$.

- A point $\theta^\star \in \mathbb{R}^d$ is a **saddle point** if

$$\nabla f(\theta^\star) = 0$$

  and $\theta^\star$ is neither a local minimizer nor a local maximizer.

# Optimization Terminology: Minima and Saddle Points

## Interpretation and course relevance

Here, "stationary" means that the gradient is zero, so the algorithm does not move. This notion is **different from stationarity in time series**.

Understanding whether a stationary point is a minimum or a saddle point is crucial, because gradient-based algorithms may get stuck at saddle points, and noise can help escape them.

# Learning Rate, Time Scale, and Continuous Dynamics

## Why continuous-time models appear

Learning algorithms such as gradient descent and stochastic gradient descent update parameters by taking small steps whose size is controlled by the learning rate $\eta$.

When $\eta$ is small, the algorithm evolves slowly and the sequence $\{\theta_k\}$ can be viewed on a rescaled time axis $t = k\,\eta$. In this sense, the learning rate sets the **time scale** of the dynamics.

Under this viewpoint:

- gradient descent is approximated by a continuous-time deterministic system (an ODE),
- stochastic gradient descent is approximated by a continuous-time random system (a diffusion).

This perspective explains why tools from dynamical systems and stochastic processes are natural for studying optimization, reinforcement learning, and neural ODEs.

# Course Roadmap

## Core modules

1. Foundations: Brownian motion, Markov processes, Itô calculus, SDEs.
2. Stochastic optimization: SGD for nonconvex losses (selected theory).
3. Continuous-time RL: stochastic control and Hamilton–Jacobi–Bellman (HJB) ideas.
4. Continuous-time deep learning: Neural ODEs/SDEs and controlled differential equations.

## Week 1 focus

Establish the unifying viewpoint and minimal language needed to begin SDE foundations.

# Session 1 Summary

## Main takeaways (Part I)

- **Modern learning algorithms evolve over time.** Methods such as gradient descent, stochastic gradient descent, and reinforcement learning do not give an answer in one step. Instead, they generate a sequence of updates that gradually change the parameters.
- **Randomness plays a central role in learning.** In practice, updates are noisy because they rely on sampling, partial data, or interaction with uncertain environments. This randomness strongly influences how learning behaves over long periods of time.

# Session 1 Summary

## Main takeaways (Part II)

- **The learning rate sets the speed of learning.** The learning rate controls how fast parameters change from one step to the next. Small learning rates lead to slow and smooth evolution, while larger learning rates lead to faster but potentially unstable behavior.
- **Continuous-time models help us understand learning behavior.** By viewing learning algorithms on a continuous time scale, we can use ideas from dynamical systems and stochastic processes to study stability, convergence, and long-term behavior. This viewpoint unifies optimization, reinforcement learning, and neural ODEs.

# Session 1 Summary

## What we build next (and why)

To analyze stochastic gradient descent, reinforcement learning, and continuous-time learning models, we will develop the following stochastic-process foundations:

- stochastic processes and their evolution over time,
- filtrations and information available to learning algorithms,
- conditional expectation and averaging of random updates,
- Brownian motion as a model of accumulated noise,
- stochastic differential equations and their solutions.

These tools form the mathematical language used throughout the course to study optimization dynamics, learning under uncertainty, and neural ODE/SDE models.