MIT 9.520/6.7910
Statistical Learning Theory and Applications

Class 11: Sequential prediction as learning dynamical systems

Lorenzo Rosasco

So far

- Theory and algorithms for learning from i.i.d. data.

- What about non i.i.d. data?

Sequential prediction

$$s_0, s_1, \ldots, s_T \quad \mapsto \quad s_{T+1} \, ?$$

Forecasting Boston temperature

$$79^\circ\ 72^\circ\ 63^\circ\ 53^\circ\ 73^\circ \quad \mapsto \quad \ldots?$$

Forecasting Boston temperature

$$\underbrace{79^\circ}_{s_0}\ \underbrace{72^\circ}_{s_1}\ \underbrace{63^\circ}_{s_2}\ \underbrace{53^\circ}_{s_3}\ \underbrace{73^\circ}_{s_4} \quad \mapsto \quad \underbrace{\ldots?}_{s_5}$$

Next token prediction

The cat sat on the $\mapsto$ ...?

Next token prediction

$$\underbrace{\text{The}}_{s_0}\ \underbrace{\text{cat}}_{s_1}\ \underbrace{\text{sat}}_{s_2}\ \underbrace{\text{on}}_{s_3}\ \underbrace{\text{the}}_{s_4} \quad \mapsto \quad \underbrace{\ldots?}_{s_5}$$

Non i.i.d. data

$$s_0, s_1, \ldots, s_T \quad \mapsto \quad s_{T+1} \, ?$$

The i.i.d. assumption breaks down.

Outline

Data models

Non i.i.d. data

Dynamic learning problems

Autoregressive approaches

Hidden state models

Discrete time dynamical systems

$$\begin{cases} s_0 \in \mathcal{S}, \\ s_{t+1} = f(s_t). \end{cases}$$

- $\mathcal{S}$ state space (tokens).
- $f : \mathcal{S} \to \mathcal{S}$ evolution map.

State space

$$\mathcal{S}$$

- $\mathcal{S} = \mathbb{R}^d$ continuous state space.
- $\mathcal{S} = \{w_1, \ldots, w_N\}$ finite state space. In this case f is a *"table"*.

Discrete vs continuous time

$$s_{t+1} = f(s_t)$$

- Discrete time evolution can arise from discretization of continuous time evolutions

$$\dot{s}(t) = f(s(t)).$$

- But many problems are inherently discrete in time!

Non-autonomous systems

$$s_{t+1} = f(s_t)$$

- The above evolution is called autonomous.

- For non-autonomous systems, the evolution depends on time

$$s_{t+1} = f(s_t, t).$$

Higher order systems

$$s_{t+1} = f(s_t)$$

- The above system is first order.
- A system of order p is

$$s_{t+1} = f(s_t, s_{t-1}, \ldots, s_{t-p+1}).$$

In next token prediction, the order p is called the **context**.

Stochastic dynamical systems

$$s_{t+1} = f(s_t, \omega_t)$$

- The variables ω_t are i.i.d. random variables, independent of s_0.
- The evolution is stochastic.
- It can be seen as a perturbation of a deterministic evolution, or as an inherently stochastic one.

The above system is equivalent to a Markov process.

Markov processes

$$\mathbb{P}(s_{t+1} \mid s_t, s_{t-1}, \ldots, s_0) = \mathbb{P}(s_{t+1} \mid s_t)$$

- The next state depends only on the current one.

Stochastic dynamical systems & Markov process

$$s_{t+1} = f(s_t, \omega_t) \quad \Longleftrightarrow \quad \mathbb{P}(s_{t+1} \mid s_t, s_{t-1}, \dots, s_0) = \mathbb{P}(s_{t+1} \mid s_t)$$

- Indeed,

$$s_{t+1} = f(s_t, \omega_t) \;\Rightarrow\; \mathbb{P}(s_{t+1} \mid s_t) = \mathbb{P}_{\omega_t}\big(f(s_t, \omega_t)\big)$$

- Conversely,

$$\mathbb{P}(s_{t+1} \mid s_t) = \mathbb{P}(s_{t+1} \mid s_t, \dots, s_0) \;\Rightarrow\; s_{t+1} = f(s_t, \omega_t)$$

Transition kernels

Let $\mathcal{S} = \mathbb{R}^d$.

Transition kernel

$$P : \mathcal{S} \times \mathcal{B}(\mathcal{S}) \to [0, 1].$$

For each $s \in \mathcal{S}$, $P(s, \cdot)$ is a probability distribution on $\mathcal{S}$.

Markov processes and transition kernels

$$P(s, A)$$

- Each Markov process has an associated kernel

$$P(s, A) = \mathbb{P}(s_{t+1} \in A \mid s_t = s).$$

- Conversely, given P and an initial distribution μ_0, we can define a Markov process by

$$\mathbb{P}(s_0 \in A) = \mu_0(A), \quad \mathbb{P}(s_{t+1} \in A \mid s_t = s) = P(s, A).$$

Transition matrices

Let $\mathcal{S} = \{w_1, \dots, w_N\}$.

Transition matrix $P \in \mathbb{R}^{N \times N}$ such that $P_{ij} \in [0, 1]$.

For each $i \in \{1, \dots, N\}$, $P(i, \cdot)$ is a probability distribution on $\mathcal{S}$, that is

$$\sum_{j=1}^{N} P_{ij} = 1.$$

Markov processes and transition matrices

$$P_{ij}$$

- Each Markov process with a finite state space has an associated transition matrix

$$P_{ij} = \mathbb{P}(s_{t+1} = w_j \mid s_t = w_i).$$

- Conversely, given P and an initial distribution μ_0, we can define a Markov process by

$$\mathbb{P}(s_0 = w_i) = \mu_0(w_i), \quad \mathbb{P}(s_{t+1} = w_j \mid s_t = w_i) = P_{ij}.$$

Invariant distribution

Given a Markov process, π is an *invariant distribution* if [1]

$$\pi P = \pi.$$

There might be no, one, or multiple invariant distributions.

[1] For $\mathcal{S} = \mathbb{R}^d$, this means $\int P(s, A)\, d\pi(s) = \pi(A)$ for all measurable A.

Ergodic Markov processes

Let μ_t denote the distribution of s_t. The process is *ergodic* if for any initial distribution μ_0,

$$\mu_t \;\rightarrow\; \pi \quad \text{as } t \rightarrow \infty.$$

Then π is unique.

Mixing and burn-in time

The *mixing time* quantifies how fast the distribution μ_t converges to π.

The *burn-in time* refers to the initial transient phase before the process is close to π.

Dynamical systems & Markov processes

$$s_{t+1} = f(s_t, \omega_t) \quad \iff \quad \mathbb{P}(s_{t+1} \mid s_t) = \mathbb{P}(s_{t+1} \mid s_t, s_{t-1}, \ldots, s_0)$$

- Autonomous systems: *time-homogeneous* Markov processes.
- Higher-order systems: p*-th order* Markov processes

$$\mathbb{P}(s_{t+1} \mid s_t, s_{t-1}, \ldots, s_0) = \mathbb{P}(s_{t+1} \mid s_t, \ldots, s_{t-p+1}).$$

- Continuous-time stochastic systems: *stochastic differential equations*, or equivalently, *continuous-time Markov processes*.

Markov processes and language models

$$\underbrace{\text{The}}_{s_0}\ \underbrace{\text{cat}}_{s_1}\ \underbrace{\text{sat}}_{s_2}\ \underbrace{\text{on}}_{s_3}\ \underbrace{\text{the}}_{s_4} \quad \mapsto \quad \underbrace{\ldots ?}_{s_5}$$

This Markov process perspective on next-token prediction dates back to Shannon[2].

In language models, it relates to p-grams (n-grams).

[2] Claude Shannon's 1948 paper "A Mathematical Theory of Communication".

Outline

Data models

Non i.i.d. data

Dynamic learning problems

Autoregressive approaches

Hidden state models

Data collection

Data can be seen as sample trajectories.

Single trajectory:

$$(s_0, s_1, \dots, s_T).$$

Multiple trajectories:

$$\begin{aligned}
&(s_0^{(1)}, s_1^{(1)}, \dots, s_T^{(1)}) \\
&(s_0^{(2)}, s_1^{(2)}, \dots, s_T^{(2)}) \\
&\vdots \\
&(s_0^{(N)}, s_1^{(N)}, \dots, s_T^{(N)}).
\end{aligned}$$

Dependent data

$$s_0, s_1 \ldots, s_T$$

- We cannot expect independent data.
- It can be shown that the *mixing time* quantifies dependence.
- Independence can be reinstated by collecting multiple trajectories, e.g. for $p = 1$:

$$(s_0^{(1)}, s_1^{(1)}),\ (s_0^{(2)}, s_1^{(2)}),\ \ldots,\ (s_0^{(N)}, s_1^{(N)}).$$

(Not very practical.)

Non identically distributed data

$$s_t \sim \mu_t$$

- Different distributions at different times.
- After the burn-in time t_b,
$$\mu_t \approx \pi \quad \text{for } t \geq t_b.$$
- Stationary processes often assumed in time series analysis, i.e.
$$(s_t, s_{t+1}, \dots) \stackrel{d}{=} (s_{t+\tau}, s_{t+\tau+1}, \dots), \quad \forall \tau \geq 1.$$

Outline

Data models

Non i.i.d. data

Dynamic learning problems

Autoregressive approaches

Hidden state models

Different problems

Dynamical systems are studied by different communities with different focus.

- **Prediction / forecasting**: find $\widehat{f}$ s.t.

$$\widehat{f}(s_t) \approx s_{t+1}.$$

- Estimation: find $\widehat{p}$ s.t.

$$\widehat{p}(s_{t+1} \mid s_t) \approx p(s_{t+1} \mid s_t).$$

- Long-time behaviour: stability, chaos, attractors.

Loss functions

$$\widehat{f}(s_t) \approx s_{t+1}.$$

How shall we measure accuracy of $\widehat{f} : \mathcal{S} \to \mathcal{S}$?

Pick a loss:

- If $\mathcal{S} = \mathbb{R}^d$:
$$\ell(s, \widehat{f}(s)) = \|s - \widehat{f}(s)\|^2.$$

- If $\mathcal{S} = \{w_1, \ldots, w_N\}$:
$$\ell(s', \widehat{f}(s)) = \mathbf{1}_{\{s' \neq \widehat{f}(s)\}}.$$

Empirical risk

$$\widehat{L}(f) = \frac{1}{T}\sum_{t=0}^{T-1} \ell(s_{t+1}, f(s_t))$$

How good is a solution?

$$\widehat{f}$$

- **Practice: test error**[3]

$$\widehat{L}_{\mathsf{Test}}(\widehat{f}) = \frac{2}{T} \sum_{t=T/2}^{T-1} \ell(s_{t+1}, \widehat{f}(s_t))$$

- **Excess risk** (with ergodicity)

$$L(f) = \mathbb{E}_{s_t \sim \pi}[\ell(s_{t+1}, f(s_t))] = \int \ell(s', f(s))\, P(s, ds')\, d\pi(s).$$

This might not be the natural definition; other error metrics can be considered, e.g. the regret.

[3] Use $T/2$ points for $\widehat{f}$.

Target functions

$$f_*(s) = \arg\min_{\overline{s} \in \mathcal{S}} \int \ell(s', \overline{s})\, P(s, ds')$$

Analog of the inner risk.

- For $\mathcal{S} = \mathbb{R}^d$ and $\ell(s, \widehat{f}(s)) = \|s - \widehat{f}(s)\|^2$:

$$f_*(s) = \mathbb{E}[s_{t+1} \mid s_t = s].$$

- For $\mathcal{S} = \{1, \dots, N\}$ and $\ell(s, \widehat{f}(s)) = \mathbf{1}_{\{s \neq \widehat{f}(s)\}}$:

$$f_*(i) = \arg\max_{j=1,\dots,N} P_{ij}.$$

Remarks

- Next, we focus on $\mathcal{S} = \mathbb{R}^d$.
- The ideas can be extended to $\mathcal{S} = \{w_1, \ldots, w_N\}$ using ideas from multiclass learning.
- Also we could consider predicting $s_{t+\tau}$ rather than just s_{t+1}, e.g. by

$$\widehat{f}^{\tau}(s_t) \approx s_{t+\tau}.$$

Outline

Data models

Non i.i.d. data

Dynamic learning problems

Autoregressive approaches

Hidden state models

Basic idea

$$x_t = s_t \qquad y_t = s_{t+1}$$

This is a vector valued regression problem $y \in \mathbb{R}^d$.

Train your favorite model f_θ on

$$(x_0, y_0), \ldots, (x_{T-1}, y_{T-1}).$$

Adding context

$$x_t = (s_t, \ldots, s_{t-p+1}) \qquad y_t = s_{t+1}.$$

Still a regression problem, but the input is $x \in \mathcal{S}^p$, that is $x \in \mathbb{R}^{d \times p}$.

Each x is a collection of states (tokens).

Linear models aka AR(p)

For $d = 1$, that is $\mathcal{S} = \mathbb{R}$,

$$x = (s_1, \dots, s_p) \in \mathbb{R}^p$$

Consider,

$$f_\theta(x) = \sum_{j=1}^{p} a_j s_j, \qquad \theta = (a_1, \dots, a_p) \in \mathbb{R}^p.$$

Linear models aka VAR(p)

For $d > 1$, that is $\mathcal{S} = \mathbb{R}^d$,

$$x = (s_1, \ldots, s_p), \quad s_j \in \mathbb{R}^d.$$

Consider,

$$f_\theta(x) = \sum_{j=1}^{p} A_j s_j, \quad \theta = (A_1, \ldots, A_p), \quad A_j \in \mathbb{R}^{d \times d}.$$

Beyond linear models aka NARX

$$f_\theta(x) = \sum_{j=1}^{p} A_j s_j, \quad x = (s_1, \ldots, s_p), \quad s_j \in \mathbb{R}^d.$$

- Feature maps

$$f_\theta(x) = \sum_{j=1}^{p} A_j \Phi(s_j), \quad A_j \in \mathbb{R}^{d \times m}, \ \Phi(s_j) \in \mathbb{R}^m, \quad \theta = (A_1, \ldots, A_p).$$

- Neural networks, as above, but

$$\Phi(s_j) = \sigma(W_j s_j + b_j), \quad \theta = (A_1, \ldots, A_p, W_1, \ldots, W_p, b_1, \ldots, b_p).$$

Transformers

$$f_\theta(x) = \mathsf{MLP}\big(A(s_1, \ldots, s_p)\big), \quad x = (s_1, \ldots, s_p).$$

- MLP is a fully connected neural network, as before.

- A is the attention layer.

Attention

$$A(x) = (A_1(x), \ldots, A_p(x)), \quad x = (s_1, \ldots, s_p).$$

Where $A_j(x)$

$$A_j(x) = \sum_{i=1}^{p} \alpha_{ji}(x)\, s_i, \quad \alpha_{ji}(x) = \frac{\exp(s_j^\top s_i)}{\sum_{k=1}^{p} \exp(s_j^\top s_k)}.$$

Each token is represented as a weighted combination of all tokens.

Attention and embeddings

The inner product may not capture tokens similarity.

$$A_j(x) = \sum_{i=1}^{p} \alpha_{ji}(x)\, V s_i, \quad \alpha_{ji}(x) = \frac{\exp\big((Qs_j)^\top (Ks_i)\big)}{\sum_{k=1}^{p} \exp\big((Qs_j)^\top (Ks_k)\big)}.$$

- Tokens are embedded into so-called queries Qs_j, keys Ks_i, and values Vs_i.
- Q, K, and V are parameters to be learned.

Outline

Data models

Non i.i.d. data

Dynamic learning problems

Autoregressive approaches

Hidden state models

Non-autonomous systems

$$s_{t+1} = f(s_t, t)$$

- Can be approximated using **high**-order autonomous systems

- Next, we describe models based on hidden states.

Hidden state models

$$\begin{cases} h_{t+1} = g(h_t, s_t) \\ s_{t+1} = q(h_{t+1}) \end{cases}$$

- The evolution of the pair (s_t, h_t) is autonomous.
- The evolution of s_t alone depends on the entire past (non-autonomous).
- Also called state space models (SSM).

Linear state space models

$$\begin{cases} h_{t+1} = Ah_t + Bs_t, \\ s_{t+1} = Ch_{t+1}. \end{cases}$$

- Evolution maps are linear.

- Related to Kalman filtering, where noise is modeled explicitly.[4]

[4] Note that here we are only interested in predicting the observable state.

Recurrent neural networks (RNN)

$$\begin{cases} h_{t+1} = \sigma(Ah_t + Bs_t) \\ s_{t+1} = Ch_{t+1} \end{cases}$$

- Evolution maps are nonlinear.
- σ is an activation function.

Recurrent neural networks (RNN) (cont.)

$$\begin{cases} h_{t+1} = \sigma(Ah_t + Bs_t) \\ s_{t+1} = Ch_{t+1} \end{cases}$$

- Unrolling: RNNs are DNNs with # layers = #time steps, and same weights in each layer.
- Optimized through so-called backpropagation through time (BPTT).
- Vanishing gradients lead to variants, e.g. LSTM and GRU, etc.

Summary

- A dynamical system tour of sequential prediction.
- Focus on a simple next token prediction task.
- Basic algorithmic principles.

Forecasting Boston temperature

$$\underbrace{79^\circ}_{s_1}\ \underbrace{72^\circ}_{s_2}\ \underbrace{63^\circ}_{s_3}\ \underbrace{53^\circ}_{s_4}\ \underbrace{73^\circ}_{s_5} \quad \mapsto \quad \underbrace{55^\circ}_{s_6}$$

Time to go to Italy.