Machine learning for dynamical systems

Seminar led by Weinan E
Scribe: Holden Lee

April 11, 2019

Contents

Chapter 1

Learning dynamical systems

1 2019-2-21 Kalman filter

1.1 Kalman filter

Consider a LDS with state variable x_k, inputs u_k, and outputs z_k. We assume:

1.
$$x_{k+1} = F_k x_k + B_k u_k \tag{1.1}$$
$$z_k = H_k x_k + n_k \tag{1.2}$$

2.
$$w_k \sim N(0, Q_k) \tag{1.3}$$
$$n_k \sim N(0, R_k). \tag{1.4}$$

The Kalman filter is an optimal estimator in this case.

The Kalman filter is computationally and memory efficient.
Given

$$P(x_{k-1}|x_{k-2}, z_{k-1}) \sim N(\widehat{x}_{k-1|k-1}, P_{k-1|k-1}) \tag{1.5}$$
$$p(x_k|x_k, z_k) \sim N(\widehat{x}_{k|k}, P_{k|k}) \tag{1.6}$$

and z_k, u_k, the Kalman filter can be conceptualized as a 2-step algorithm:

1. Prediction:

$$\text{State prediction (a priori estimation)} \qquad \widehat{x}_{k|k-1} = F_k \widehat{x}_{k-1|k-1} + B_k u_k \tag{1.7}$$
$$\text{Covariance prediction} \qquad P_{k|k-1} = F_k P_{k|k-1} F_k^\top + Q_k. \tag{1.8}$$

2. Update.

   We solve a MLE estimation.

   Lemma:

$$\begin{pmatrix} x \\ y \end{pmatrix} \sim N\left(\begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix}, \begin{pmatrix} \Sigma_x & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_y \end{pmatrix}\right) \tag{1.9}$$

$$\mu_{y|x} = \mu_y + \Sigma_{yx}\Sigma_x^{-1}(x - \mu_x) \tag{1.10}$$

$$\Sigma_{y|x} = \Sigma_y - \Sigma_{yx}\Sigma_x^{-1}\Sigma_{xy} \tag{1.11}$$

   The new covariance is the original covariance minus a correction term.

   Then the joint distribution is

$$\begin{pmatrix} x_k \\ z_k \end{pmatrix} \sim N\left(\begin{pmatrix} \widehat{x}_{k|k-1} \\ H_k\widehat{x}_{k|k-1} \end{pmatrix}, \begin{pmatrix} P_{k|k-1} & P_{k|k-1}H_k^\top \\ H_kP_{k|k-1} & H_kP_{k|k-1}H_k^\top + R_k \end{pmatrix}\right) \tag{1.12}$$

3. The Kalman gain K_k is how confident we are about our observations.

$$K_k = P_{k|k-1}H_k^\top(H_kP_{k|k-1}H_k^\top + R_k)^{-1} \tag{1.13}$$

4. State update:

$$\widehat{x}_{k|k} = \widehat{x}_{k|k-1} + K_k(Z_k - H_k\widehat{x}_{k|k-1}) \tag{1.14}$$

5. Covariance estimate

$$P_{k|k} = (I - K_kH_k^\top)P_{k|k-1} \tag{1.15}$$

   and find

$$\min_{x_k} \frac{1}{2}(x_k - \widehat{x}_{k|k-1})^\top P_{k|k-1}^{-1}(x_k - \widehat{x}_{k|k-1}) + \frac{1}{2}(z_k - H_kx_k)^\top R_k^{-1}(z_k - H_kx_k). \tag{1.16}$$

   The Kalman filter is also called linear quadratic estimation.

I'll talk about 3 extensions: Kalman smoother, high-dimensional data, nonlinear Kalman filter.

(Kalman filtering is not a machine learning problem. System identification is closer to ML.)

Applications in control theory: LQG (linear quadratic gaussian) control. Tracking problems: observations is GPS signal, we observe speed; predict next location. It's not accurate because odometry is not accurate; GPS has a few meters accuracy. Google maps is more accurate than GPS, 1m accurate. When you first open Google maps, you have a big circle representing uncertainty, when you get more observations it shrinks. Macroeconomic model: can include expectation terms.

What are the challenges? System identification, nonlinear systems, non-gaussian noise. Linear gaussian are the simplest assumptions.

Subspace problem: PCA. LQR: simultaneously learn and control.

You can treat F_k, B_k as state and use a nonlinear filter. This is called an adaptive filter.

1.2 Kalman smoothing

Assume we have observations up to $N > k$, we want to estimate $\widehat{x}_{k|N}$ and $P_{k|N}$. Writing out a huge optimization problem is not efficient. Kalman is a forward propagation; Kalman smoothing is a backwards propagation. Analogue of Viterbi algorithm. (Getting the state is Baum-Welch.)

Given

$$P(x_{k+1}|z_{0:N}) \sim N(\widehat{x}_{k+1|N}, P_{k-1|N}) \tag{1.17}$$

and $\widehat{x}_{k|k}, P_{k|k}$, solve

$$P(x_k|z_{0:N}) \sim N(\widehat{x}_{k|N}, P_{k|N}). \tag{1.18}$$

Treat this as a Kalman filtering problem with $z'_k = x_{k+1}$.

$$\begin{pmatrix} x_k \\ x_{k+1} \end{pmatrix} \sim N\left(\begin{pmatrix} \widehat{x}_{k|k} \\ \widehat{x}_{k+1|N} \end{pmatrix}, \begin{pmatrix} P_{k|k} & P_{k|k}F_k^\top \\ F_k P_{k|k} & P_{k+1|k} \end{pmatrix}\right) \tag{1.19}$$

$$\widehat{x}_{k|N} = \widehat{x}_{k|k} + C_k(\widehat{x}_{k+1|N} - \widehat{x}_{k+1|k}) \tag{1.20}$$

$$\widehat{P}_{k|N} = P_{k|k} + C_k(P_{k+1|N} - P_{k+1|k})C_k^\top \tag{1.21}$$

$$C_k = P_{k|k}F_k^\top P_{k+1|k}^{-1} \tag{1.22}$$

1.3 High-dimensional data: Ensemble Kalman Filter (EnKF)

We assume $x_k \in \mathbb{R}^{n\times 1}$, $z_k \in \mathbb{R}^{m\times 1}$. Consider $n \gg m$. Observations are limited, but data are high-dimensional.

This algorithm is empirical. It's widely used in geostatistics. There are several mines; they have limited data from mines. They want to predict petroleum reserves everywhere, like superresolution. Data assimilation.

The expensive part is the multiplication $H_k P_{k|k-1} H_k^\top$, $O(mn^2)$. The inverse has complexity $O(m^3)$. Try to simplify the computation of the Kalman gain K_k.

$$\widehat{x}_{k|k} = \widehat{x}_{k|k-1} + K_k(Z_k - H_k\widehat{x}_{k|k-1}) \tag{1.23}$$

$$K_k = P_{k|k-1}H_k^\top(H_k P_{k|k-1} H_k^\top + R_k)^{-1} \tag{1.24}$$

It's a Monte Carlo-based method. We have particles $X^{(i)}_{k-1|k-1}$ for $i = 1, \ldots, s$ for $s \ll n$. The algorithm: Initially sample from the initial gaussian distribution.

1. State update

$$\widehat{x}^{(i)}_{k|k-1} = F_k \widehat{x}^{(i)}_{k-1|k-1} + Bu_k \tag{1.25}$$

2.

$$\widehat{P}_{k|k-1} = \mathrm{Var}(\widehat{x}_{k|k-1}) = \frac{AA^\top}{s-1} \tag{1.26}$$
$$A = [\widehat{x}^{(i)}_{k|k-1} - \mathbb{E}[\widehat{x}^{(i)}_{k|k-1}] \tag{1.27}$$

Propagate the samples through time. Replace the matrix in the inverse by $(H_kA)(A^\top H_k)$, in the computation for K_k. (Think $n \sim 10^7$, $m \sim 10^5$, $s \sim 10^3$.)

Linear problem for which dimensionality high? Most things are nonlinear, do linearization. "Different copies of weather."

Keep track of variance: in linear setting we can compute exact optimal solution. In nonlinear, it's no longer correct. Is using the KF formalism still advantageous? Particle filters are more expensive. EnKF not pure MC; it solves some optimization problem. It's biased but can be close.

In weather forecasting, sometimes they use a nonlinear transition but still use EnKF, it still works to some extent. Tricks: Iterated extended KF, uncertainty transform...

(In order to estimate variance need $s > m$? If R_k is large, not as important.)

1.4 Extended Kalman filter (EKF) for nonlinear systems

$$x_{k+1} = f(x_k, u_k) + w_k \tag{1.28}$$
$$z_k = h(x_k) + n_k. \tag{1.29}$$

The iterated extended Kalman filter (IEKF) has

$$\widehat{x}_{k|k-1} = f(\widehat{x}_{k|k-1}, u_k) \tag{1.30}$$
$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^\top + Q_k, \qquad F_k = \frac{\partial}{\partial f} x|_{x_k = \widehat{x}_{k|k-1}, u_k = u_k} \tag{1.31}$$
$$K_k = P_k H_k^\top (H_k P_{k|k-1} H_k^\top + R_k)^{-1} \tag{1.32}$$
$$\widehat{x}_{k|k} = \widehat{x}_{k|k-1} + K_k(z_k - h(x_{k|k-1})) \tag{1.33}$$
$$P_{k|k} = (I - K_k H_k^\top) P_{k|k-1}. \tag{1.34}$$

Ex. adaptive optics problems. $x_{k+1} = x_k + f(x_k, u_k) + w_k$, $z_k = x_k^\top x_k + n_k$. z_k intensity observed, x_k electric field. Like phase retrieve. Fourier (nonlinear) optics.

Possible directions:

1. When the system is unknown (hidden Markov model, including system identification). Ex. HMM is diffusion process, but don't know the model, what do we learn?

2. Nonlinear filtering.

2 2019-2-28 Nonlinear filtering

Outline:

- Zakai and Kushner equiations
- Girsanov theory
- Derivation
- Particle filtering
- Deep learning to solve Zakai

2.1 Zakai and Kushner equation

The hidden state evolves as

$$dX_t = f_t(X_t)\,dt + \sigma\,dW_t. \tag{1.35}$$

The observation is

$$dY_t = h_t(X_t)\,dt + \eta dB_t. \tag{1.36}$$

Here, B_t, W_t are independent Brownian motion. The goal is to calculate the posterior

$$p_t(x) = p(X_t = x | Y_s = y_s, 0 \le s \le t). \tag{1.37}$$

The first term is from Fokker-Planck, the second term is variance reduction from the observations. For the Kalman filter, covariance matrix is subtracted by a positive definite matrix. The same thing is going on here.

The Kushner equation solves for the posterior density.

$$dp_t(x) = \mathcal{L}^*[p_t]\,dt + p_t(x)[h_t(x) - \mathbb{E}h_t]^\top \eta^{-\top}\eta^{-1}[dy_t - \mathbb{E}h_t]. \tag{1.38}$$

Here

$$\mathcal{L} = f\frac{\partial}{\partial x} + \frac{1}{2}(\sigma\sigma^\top)\frac{\partial^2}{\partial x^2} \tag{1.39}$$

$$\mathcal{L}^*[p_t] = -\nabla\cdot[f_t\cdot p_t] + \frac{1}{2}(\sigma\sigma^\top)\cdot\partial^2 p_t \tag{1.40}$$

$$dg(x_t) = \mathcal{L}[g]\,dt \tag{1.41}$$

$$\mathbb{E}_t h_t = \int h_t(x)p_t(x)\,dx \tag{1.42}$$

We have to update both; this is numerically not easy. We introduce an unnormalized verision that is easier.

Zakai gives an unnormalized density,

$$d\widetilde{p}_t(x) = \mathcal{L}^*[\widetilde{p}_t]\,dt + \widetilde{p}_t(x)h_t^\top(x)\eta^{-\top}\eta^{-1}\,dy_t. \tag{1.43}$$

To say it is an unnormalized density means

$$\widetilde{p}_t(x) = \alpha(t)p_t(x) \tag{1.44}$$

$$\int \widetilde{p}_t(x) \neq 1 \text{ in general.} \tag{1.45}$$

Remark 1.2.1: Assume Y_t/η is Brownian motion; let Q be its law. Let $\frac{dQ}{dP} = M_T^{-1}$. Define

$$\varphi_t[g] := \mathbb{E}_Q[g(X_t)M_t|\mathcal{F}_t^Y] \tag{1.46}$$

$$\psi_t[g] := \mathbb{E}_P[g(X_t)|\mathcal{F}_t^Y] = \frac{\varphi_t[g]}{\varphi_t[1]}. \tag{1.47}$$

$\alpha(t) := \varphi_t[1]$.

The forward form of the Kushner equation is

$$\psi_t[g] = \mathbb{E}[g(X_t)|\mathcal{F}_t^Y] \tag{1.48}$$

$$d\psi_t[g] = \psi_t[\mathcal{L}g]\,dt + \frac{\psi_t[gh] - \psi_t[g]\psi_t[h]}{\eta^2}(dY_t - \psi_t[h]\,dt) \tag{1.49}$$

To show the variance is smaller than in the Fokker-Planck equation we can take $g(x) = x^2$.

Remark 1.2.2: If there is no noise, and f is unknown, this is a ML problem (cf. RNN). For Kalman filtering if we know the model, it's not ML; if we don't know the model, we have to do system identification, this is ML.

Suppose $f_t, h_t = 0$, and we only have noise. It's just Brownian motion. If we have observations up to time t, that means the system picked a particular choice of Brownian motion. We don't know the particular path. If we have observations, we know something about the path. The variance is smaller, we solve a filtering problem.

Filtering by itself is variance reduction (through observations). Estimation is machine learning.

2.2 Girsanov theory

We use a change of measure to make the integration easier. Let's consider a simple example first.

Example 1.2.3: Let $(\Omega, \mathcal{F}, P)$ be a probability space and $\xi \sim N(0,1)$ under P. For $a \in \mathbb{R}$, then $\xi' = \xi + a \sim N(a,1)$ under P. We want to find Q such that $\xi' \sim N(0,1)$ under Q.

For any $A \in B(\mathbb{R})$,

$$P(\xi' \in A) = \int_A \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(x-a)^2}\, dx \tag{1.50}$$

$$Q(\xi' \in A) = \int_A \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}\, dx \tag{1.51}$$

$$\frac{dQ}{dP} := e^{-\frac{1}{2}\xi'^2 + \frac{1}{2}(\xi'^2 - a)^2} = e^{-a\xi' + \frac{1}{2}a^2} = e^{-a\xi - \frac{1}{2}a^2}. \tag{1.52}$$

Then

$$\mathbb{E}_Q[f(\xi')] = \mathbb{E}_Q[f(\xi + a)] = \mathbb{E}_p[f(\xi + a)e^{-a\xi - \frac{1}{2}a^2}] = \mathbb{E}_p[f(\xi)]. \tag{1.53}$$

We can also do the change-of-variable for a random process, using Girsanov. We will use it to remove drift $h(X_t)$, and make Y_t Brownian motion.

Theorem 1.2.4 (Girsanov)**.** *Let $(\Omega, \mathcal{F}, \{\mathcal{F}_i\}, P)$ be a probability space with filtration, B be n-dimensional Brownian motion, $X \in \mathcal{H}^2_{loc}[0, T]$. Define*

$$\frac{dQ}{dP} = M_T := \exp\left(-\int_0^T X_s\, dB_s - \frac{1}{2}\int_0^T \|X_s\|^2\, ds\right) \tag{1.54}$$

$$B'_t = B_t + \int_0^t X_s\, ds. \tag{1.55}$$

If $\mathbb{E}_P M_T = 1$, then $(B'_t)_{t \le T}$ is Brownian motion under Q. Also, M_t is a P-martingale.

Remark 1.2.5 (Novikov)**:** If $\mathbb{E}\exp\left(\frac{1}{2}\int X_s^2\, ds\right) < \infty$, then $\mathbb{E}M_T < 1$.

2.3 Derivation

Omitted.

2.4 Particle filtering

Particle filtering is Monte Carlo sampling with modifications based on observations.

We know $p(X_0)$, $p(X_t|X_{t-1})$, $p(Y_t|X_t)$, and have observations $Y_{1:T} = y_{1:T}$. The goal is to sample $\xi_t \sim p(X_t|Y_1, \dots, Y_t)$.

The algorithm is as follows.

Algorithm 1.2.6 (Particle filtering)**:**

- Let $\pi_0 = p(X_0)$.
- For $t = 1, \dots, T$, do:
  - (Resampling) Sample $\xi^{(i)}_{t-1} \sim \pi_{t-1}$, iid, for $i = 1, \dots, N$.
  - Sample $\xi^{(i)}_{t-\frac{1}{2}} \sim p(X_t|X_{t-1} = \xi^{(i)}_{t-1})$ (from the model).

– Compute

$$w_t^{(i)} = \frac{p(Y_t = y_t | x_t = \xi^{(i)}_{t-\frac{1}{2}})}{\sum_{j=1}^N p(Y_t = y_t | X_t = \xi^{(j)}_{t-\frac{1}{2}})} \tag{1.56}$$

– Let $\pi_t = \sum_{i=1}^N w_t^{(i)} \delta^{(i)}_{\xi_{t-\frac{1}{2}}}$.

When sampling π_{t-1}, not all points will be sampled and some will be sampled more than once.

The rate is $\frac{1}{\sqrt{N}}$, but the constant depends on the dimension and increases exponentially. If we make some assumptions on the X, e.g. sparsity of interactions (and particles far away are not correlated), we can do better.

(Ramon van Handel's paper: can we overcome curse of dimensionality?Conclusion, not so clear.)

2.5 Deep learning for solving Zakai

Numerical method to solve Zakai or Kushner equations directly.

Zakai: $d\widetilde{p}_t(x) = \mathcal{L}^*[\widetilde{p}_t]\,dt + \widetilde{p}_t(x)\frac{h_t(x)}{\eta^2}\,dy_t$.

The method can help us calculate $\int g(x)p_t(x)\,dx$.

Given U_T, generate a backwards path

$$\begin{aligned}
U_t &= U_{t+1} - f_t(U_{t+1})\Delta t + \sigma \Delta W_t && (1.57)\\
V_t &= p_t(U_t) && (1.58)\\
V_{t+1} - V_t &= p_{t+1}(U_{t+1}) - p_t(U_{t+1}) + p_t(U_{t+1}) - p_t(U_t) && (1.59)\\
&= [-\nabla\cdot(f_t(U_{t+1})p_t(U_{t+1})) + \frac{1}{2}\sigma\sigma^\top \partial^2 p_t(U_{t+1})]\,dt && (1.60)\\
&\quad + p_t(U_{t+1})\frac{h_t(U_{t+1})}{\eta^2}\Delta Y_t + o(\Delta t) && (1.61)\\
&= -V_{t+1}\nabla\cdot f_t(U_{t+1}) - \partial p_t(U_{t+1})\cdot\sigma\Delta t + V_{t+1}h_t^\top(U_{t+1})\eta^{-\top}\eta^{-1}[\Delta Y_t - h_t(U_{t+1})\Delta t] + o(\Delta t) && (1.62)
\end{aligned}$$

The only unknown term is $-\partial p_t(U_{t+1})$. We can calculate $V_t = P_t(U_t)$ from T to 0. We know $V_0 = p_0(U_0)$. We minimize the loss function,

$$\min_{\varphi,\psi} \sum_{\text{samples } U_T, \Delta W_t} |V_0 - p_0(U_0)|^2. \tag{1.63}$$

We tested it on the Kalman filter in 2 dimensions.

3 2019-3-7 System identification (Linear gaussian state space model)

The model is

$$y_t = Cx_t + v_t \tag{1.64}$$
$$x_{t+1} = Ax_t + w_t \tag{1.65}$$

where $w_t \sim N(0, Q)$ and $v_t \sim N(0, R)$.

In economics, all macroscopic discrete time models can be reduced to this formulation.

The solution is by nature non-unique, we can replace C with $C\Omega^{-1}$ and A with $\Omega A \Omega^{-1}$. This is non-essential.

We want to estimate $P(A, C, Q, R|\{y_t\}_{t=1}^T)$. Let $\theta = (A, C, Q, R)$. By Bayes,

$$P(A, C, Q, R|\{y_t\}_{t=1}^T) \propto P(\{y_t\}_{t=1}^T|\theta)P(\theta) \tag{1.66}$$

We get this with the EM algorithm.

1. Initialize θ. Use Kalman filtering/smoother to get $P(x_{1:T}|y_{1:T,\theta})$ and $P(y_{1:T}|\theta) = \prod_{t=1}^{T-1} P(y_{t+1}|y_t, \theta)P(y_1)$. Let x_t be the conditional mean, $x_t = \mathbb{E}[x_t|y_{1:T}]$.

2. From $P(\theta|x_{1:T}, y_{1:T})$ obtain $\widehat{\theta}_{MLE}$.

The second (Bayesian) method is to do MCMC. Use the distributions instead: instead of using $\mathbb{E}[x_t|y_{1:T}]$, but take a lot of samples; likewise, instead of getting $\widehat{\theta}_{MLE}$, take a lot of samples. You need a prior; people use a conjugate prior to make things easier. Explicitly,

$$\ell(A, C, Q, R|x_{1:T}, y_{1:T}) = \frac{T}{2} \log |Q^{-1}| - \frac{1}{2} \operatorname{Tr}\left[Q^{-1}\left(\sum_{t=0}^{T-1} \cdots\right)\right] \tag{1.67}$$

(The variance is also estimated so $P(\theta, x_{1:T}, y_{1:T})$ is gaussian.)

The third method is principal component estimation, when $N \gg K$. People put a normalization condition called principal component normalization. Say $x_t \in \mathbb{R}^{F\times 1}$ and $y_t \in \mathbb{R}^{N\times 1}$. Then PC normalization has k^2 degrees of freedom, $\frac{1}{k}C^\top C = I_k$ and $\mathbb{E}[X_t^\top X_t]$ is diagonal.

Run PCA on y_t, get first K components. From the principal components you have C, and then you estimate A separately. If the second equation is nonlinear, can you still do it? You can use PCA when $\operatorname{Var}(y_t) = I$ (steady state) and $\operatorname{Var}(y_t) = C^\top C + R$.

The formulation of EM works in the nonlinear case (if you have functions for the dynamics depending on θ).

4 2019-3-21 Strategies for filtering turbulent systems: Particle filtering (Di Qi)

4.1 Strategies for filtering turbulent systems

Particle filtering suffers from the curse of dimensionality.

Fundamental challenges for real-time filtering of turbulent signals: model errors, limited ensemble size (50–100 for state space dimension 10^4 to 10^8), sparse noisy spatio-temporal observations for partial set of variables.

Filtering is the process of obtaining the best statistical estimate of a natural system from partial observations of the true signal.

$$p_{m+1,+}(u) = \frac{p_{m+1}(v_{m+1}|u)p_{m+1,-}(u)}{\int p_{m+1}(v_{m+1}|u)p_{m+1,-}(u)\,du}. \tag{1.68}$$

(Normalization: divide by sum.)

Step 1: forecast, step 2: correction. Observation reduces variance. Filtering is also called data assimiliation.

Example application: predicting path of hurricane. Large forecast changes for Hurricane Katrina in a few hours.

Theoretical and computational issues:

- Handling nonlinearity: convergence requires ensemble size growing exponentially with respec tto ensemble spread relative to observation errors (Bengtsson, Bickel, Li)
- Large systems
- Computational bottleneck: propagating covariance matrix of size $N \times N$.
- Strategies: Ensemble Kalman filters (ETKF, EAKF). Each requires computing SVD.
- Catastrophic filter divergence (go to machine ∞) when observations are sparse, even when true signal is dissipative system with "absorbing ball property."

Kalman filter optimal for linear system. We need to work with a high-dimensional covariance matrix.

Particle filter: with prior, get posterior weights and distribution

$$p_-(u) = \sum_j p_{j,-}\delta(u - u_j) \tag{1.69}$$

$$p_{j,+} \propto p(v|u_j)p_{j,-} \tag{1.70}$$

$$p_+(u) = \sum_j p_{j,+}\delta(u - u_j). \tag{1.71}$$

Capture non-gaussian statistics.

Problem: After running for several steps, weight concentrates on one particle. (Because exponential decay of probability from MLE.) This is why correction methods are proposed. EnKF maintains ensemble, but only uses it to get the mean and covariance; make a gaussian fit to the nongaussian distribution.

Particle filter vs. Kalman filter:

- PF based on MC approaches with various resampling strategies provides better estimates of low-dimensional systems than KF in presence of strong nonlinearity and highly non-gaussian distributions.
- Less feasible for high-dimensional systems. Computational constraints, particle collapsing.

Ideas for filtering high-dimensional turbulent systems: Blended method

- Decompose into 2 subspaces $u = (u_1, u_2)$ changing adaptively in time
- Subspace u_1 with dimensionality N_1 is low-dimensional enough, capturing non-gaussian structures; accurate nonlinear filters (PF) can be used.
- Orthogonal subspace u_2 where gaussian statistics assumed: cheap Gaussian filters (KF) can be used.

Problems:

- forecast: how to decompose space properly (time-dependently)
- analysis: observations mix the state variables in these two separate subspaces with different representations.

$$\begin{aligned}
p_-(u_1, u_2) &= p_-(u_1) p_-^T(u_2|u_1) && (1.72)\\
&= \sum_{j=1}^{Q} p_{j,-} \delta(u_1 - u_{1,j}^-) \mathcal{N}(u_{2,j}^-, R_{2,j}^-) && (1.73)\\
\overline{u}_{2,j}^- &= \int u_2 p^T(u_2, u_{1,j}^-)\, du_2 && (1.74)\\
R_{2,j}^- &= \cdots && (1.75)
\end{aligned}$$

Compute that the posterior distribution also has this form. Analytic pdf given by Bayes.

Orthogonal subspace statistics with Gaussian mixture representation. For u_1, adaptively evolving basis to keep track of most energetic directions of phase space.

Lorenz 96 system: mimic large-scale behavior of mid-latitude atmosphere around circle of constant latitude. (First test problem for climate filtering problems.) $\frac{du_i}{dt} = u_{i-1}(u_{i+1} - u_{i-2}) - u_i + F_i$. Quadratic part conserves energy, $B(u, u) \cdot u = 0$. Simplest example of complex turbulent dynamical system with properties in realistic systems.

Increasing F breaks up the wave structure, and makes it more gaussian.

True signal generated by running model once to time $T = 250$. Initial data: perturb initial truth state with Gaussian noise with climatological variance R_∞. Ensemble size $K = 10000$, DO subspace $S = 5$, test 2 different observation noise amplitude r_0 and observation time Δt. (Ex. sparse infrequent high quality observations in the ocean, $r_0 = 0.01$, $\Delta t = 0.25$: often leads to catastrophic filter divergence.)

To decompose space, find Fourier transform of 14 grid points. $u(t) = \sum_{k=1}^{N_t} Z_k(t)\vec{v}_k(t)$, $\frac{d\vec{v}_k}{dt}$.

Two-layer quasigeostrophic equation.

What's a good realistic test problem?

4.2 Statistical reduced models and rigorous analysis for uncertainty quantification of turbulent geophysical flows

In $\mathbb{R}^N$

$$\frac{du}{dt} = F(u,t) + \sigma(u,t)\dot{W}(t). \tag{1.76}$$

Sources of uncertainty: internal instabilities, initial and boundary conditions, model approx, limited observations and data.

Challenges: non-gaussian, large dim phase space, non-stationary dynamics, wide range of scales.

Obtain accurate stat estimates such as change in mean and variance for key statistical quantities in nonlinear response to changes in external forcing or uncertain initial data.

Imperfect model u_M, F_M, σ_M. $u_M \in \mathbb{R}^M$, want $M \ll N$.

Examples:

- geophysical fluid flows, plasma flow, high Reynolds numbers.
- systems biology, structural systems, US power network.

General framework: with energy preserving quadratic part

$$\frac{du}{dt} = \mathscr{L}[u(t;\omega);\omega] = (L+D)u + B(u,u) + F(t) + \sigma(t)\dot{W}(t;\omega), \tag{1.77}$$

$L^* = -L$, $D \leq 0$, $u \cdot B(u,u) = 0$.

Model selection (ergodic theory, statistical measures in equilibrium), model calibration (empirical information theory, linear response theory, total statistical energy equations), model prediction (numerical stability).

Damping and forcing tems. Can find exact invariant measure

$$\frac{du}{dt} = \mathscr{L}[u] = B(u,u) + Lu - d\Lambda u + \Lambda^{\frac{1}{2}}\sigma\dot{W}(t;\omega). \tag{1.78}$$

Decompose state variable into mean state and fluctuation around mean, $u(t) = \overline{u}(t) + \sum Z_i(t)v_i$, $R_{ij} = \left\langle Z_i Z_j^* \right\rangle$. The system will never be closed, because there are 2nd order terms.

$$\frac{d\overline{u}}{dt} = (L+D)\overline{u} + B(\overline{u},\overline{u}) + R_{ij}\left\langle v_i, v_j \right\rangle + F(t) \tag{1.79}$$

$$\frac{dR}{dt} = L_v R + R L_v^* + Q_F + Q_\sigma. \tag{1.80}$$

Nonlinear flux. Energy is still conserved.

Example: one-layer barotropic flow with topography (simplest model in geophysical systems). Kinetic energy and large scale enstrophy conserved.

Two-layer, more complex.

4.2.1 Reduced-order statistical model with consistency and sensitivity

Huge dynamical system in background. Smaller dynamical system with similar behavior for what you're interested in.

New systematic approach for nonlinear flux Q_F^M combining detailed model energy mechanism and control over model sensitivity.

- Model fidelity: guarantees reduced model converence to same final unperturbed statistical equilibrium. Statistical energy principle offers a closed equation.
- Model sensitivity to external perturbations requires imperfect model to correctly reflect true system's memory.

  Linear response operator characterizes model sensitivity regardless of specific perturbations. For any function of state variable, the expectation can be expanded as unperturbed equilibrium measure, plus linear response. $\mathbb{E}^\delta A(u) = E_{eq}(A) + \delta E'_A + O(\delta^2)$

  Use relative entropy as distance between two probability densities.

Flow in low-latitude regimes with zonal jets. How do they respond to external forcing?

Reduce size 256x256x2 to 10x10x2.

4.2.2 Quantifying statistical reliability

Nonlinear saturation of instabilities estimates the max growth in one unstable flow situation.

Linear dependent solution is most probable state from maximm entropy principle.

Canonical statistical theory predicts invariant Gibbs measure for truncated barotropic equation.

Idea: use conserved total statistical energy in fluctuations.

Slaving principle: perturbed mean and variance in all high wavenumber modes are "slaved" by low wavenumber large-scale perturbations for all time.

Usually many more small-scale modes.

Instability: typically treat as initial value problem. Here, treated as statistical problem.

5 Hidden Markov Models

We've learned that filtering is not really machine learning, although we hope ML can help to do filtering. If we want to learn underlying dynamical systems, in the deterministic case, use can use RNN's. We understand very little about training RNN's. What would be a good formulation about RNN? In the stochastic case we learned about the linear problem.

It seems there is a good theory in the linear case. In the nonlinear case, there is a classical example, HMM.

If you have a general Markov process, and we want to learn the hidden Markov process, what is the mathematical model? 3 examples.

- Discrete Markov chain, discrete time.
- Continuous-time (jump process), useful for chemical kinetic systems.
- Diffusion process.

5.1 EM

First I introduce vanilla EM. The setup is as follows. There are hidden variables x_i, observations y_i, parameters θ, and samples $i = 1, \dots, N$.

The goal is to solve MLE: $\max_\theta L(\theta)$, where $L(\theta) = \sum_{i=1}^N \log p(y_i; \theta)$.

In EM we need to find $Q(\theta, \theta')$ such that $L(\theta) \geq Q(\theta, \theta')$ and $L(\theta') = Q(\theta', \theta')$.

In iteration k, let

$$\theta_{k+1} = \text{argmax}_\theta\, Q(\theta, \theta_k). L(\theta_k) \qquad = Q(\theta_k, \theta_k) \leq Q(\theta_{k+1}, \theta_k) \leq L(\theta_{k+1}). \tag{1.81}$$

The likelihood function will not decrease. We cannot prove convergence to local/global optima, but in practice, we can often get good solutions.

Consider

$$\gamma_i(x, \theta') = p(X_i = x | y_i, \theta') \tag{1.82}$$

We can rewrite the likelihood function as

$$L(\theta) = \sum_i \log p(y_i; \theta) = \sum_i \log \left(\sum_x p(X_i = x, y_i; \theta) \right) \tag{1.83}$$

$$= \sum_i \log \sum_x \gamma_i(x; \theta') \frac{p(X_i = x, y_i; \theta)}{\gamma_i(x; \theta')} \tag{1.84}$$

$$\geq \sum_i \sum_x \gamma_i(x; \theta') \log \frac{p(X_i = x, y_i; \theta)}{\gamma_i(x; \theta')} =: Q(\theta, \theta'). \tag{1.85}$$

Explanation: First we write the probability as a joint probability of x and y. For γ we use θ', but for p we have θ. We use Jensen's inequality to take γ_i out of the log. Note $L(\theta') = Q(\theta', \theta')$ since $\frac{p(X_i=x, y_i; \theta')}{\gamma_i(x;\theta)} = p(y_i; \theta')$ independent of X_i.

EM algorithm:

- In the E-step, calculate $\gamma_i(x;\theta_k) = p(X_i = x|y_i;\theta_k)$.
- In the M-step, set $\theta_{k+1} = \operatorname{argmax}_\theta Q(\theta, \theta_k)$.

5.2 Discrete HMM

For discrete HMM, denote $X = X_{0:T}$, $y = y_{0:T}$ (in the setting of discrete or continuous time). The MLE problem is $\max_\theta L(\theta)$, where

$$L(\theta) = p(y_{0:T};\theta) = \sum_{x_{0:T}} p(x_{0:T}, y_{0:T};\theta). \tag{1.86}$$

(In the continuous case, the probability is on path space and we need the path integral. We first consider the discrete version.)

- In the E-step, given θ_k, we calculate the posterior $p(x_{0:T}|y_{0:T};\theta_k)$.
- In the M-step, given the posterior, update

$$\theta_{k+1} = \operatorname{argmax}_\theta \sum_{x_{0:T}} p(x_{0:T}|y_{0:T};\theta_k) \log p(x_{0:T}, y_{0:T};\theta). \tag{1.87}$$

For a discrete HMM, let the transition matrix and observation matrix be

$$A_{ij} = \mathbb{P}(X_{t+1} = j|X_t = i) \tag{1.88}$$
$$B_{ik} = \mathbb{P}(Y_t = k|X_t = i). \tag{1.89}$$

The parameters are $\theta = (A, B)$. We assume the initial distribution $\mathbb{P}(X_0 = i)$ is known. (We could also have included it in the parameters θ.) The MLE is $\max_\theta L(\theta)$, $L(\theta) = \log p(y_0, \ldots, y_T;\theta) = \sum_x \log p(x_{0:T}, y_{0:T};\theta)$ There are M^{T+1} terms, but we can use dynamic programming to solve this problem and avoid the exponential term.

Let

$$\alpha_t(i) = \mathbb{P}(X_t = i, y_{0:t}). \tag{1.90}$$

Note $\alpha_0(i) = \mathbb{P}(X_0 = i)B_{i,y_0}$. Then

$$\alpha_t(i) = \sum_j A_{ji}B_{i,y_t}\alpha_{t-1}(j) \tag{1.91}$$

Let the backwards probability be

$$\beta_t(i) = \mathbb{P}(y_{t+1:T}|X_t = i), \tag{1.92}$$

note $\beta_T(i) = 1$. Then

$$\beta_t(i) = \sum_j \beta_{t+1}(j) A_{ij} B_j y_{t+1}. \tag{1.93}$$

(This is the analogy of Kolmogorov forward/backward equation.) Then

$$\mathbb{P}(X_t = i, y_{0:T}) = \alpha_t(i)\beta_t(i) \tag{1.94}$$

$$\rho_t(i) = \mathbb{P}(X_t = i | y_{0:T}) = \frac{\alpha_t(i)\beta_t(i)}{\sum_i \alpha_t(i)\beta_t(i)}. \tag{1.95}$$

(α_t solves a filtering problem, ρ_t solves a smoothing problem.) ρ_t is not exactly what we want to calculate; we want the probability of the whole $X_{0:T}$. Ideally, we the posterior distribution of the whole pass. But $\rho_t(i)$ is almost enough. We need another function,

$$\xi_t(i,j) = \mathbb{P}(X_t = i, X_{t+1} = j | y_{0:T}) = \frac{\alpha_t(i) A_{ij} B_{j,y_{t+1}} \beta_{t+1}(j)}{\sum_i \alpha_t(i)\beta_t(i)}. \tag{1.96}$$

For the M-step, use the Baum-Welch Algorithm.

$$\max_\theta \sum_{x_{0:T}} \mathbb{P}(x_{0:T} | y_{0:T}, \theta_k) \log \mathbb{P}(x_{0:T}, y_{0:T} | \theta) = \max_\theta \sum_{x_{0:T}} \mathbb{P}(x_{0:T} | y_{0:T}; \theta_k) \Bigg[\log \mathbb{P}(x_{0:T}) + \sum_{t=0}^{T-1} \log A_{x_t, x_{t+1}} + \sum_{t=0}^{T} \log B \tag{1.97}$$

$$= \max_\theta \left[C + \sum_{x_{0:T}} \left[\sum_{t=0}^{T-1} \mathbb{P}(x_t, x_{t+1} | y_{0:T}; \theta_k) \right] \log A_{x_t, x_{t+1}} \right] \tag{1.98}$$

$$= \max_\theta \left[C + \sum_{i,j} \sum_{t=0}^{T-1} \xi_t(i,j) \log A_{ij} + \sum_i \sum_{t=0}^{T} \rho_t(i) \log B_{i,y_t} \right] \tag{1.99}$$

The constraint is $\sum_j A_{ij} = 1$, so the optimizer is seem to be

$$A^*_{ij} = \frac{\sum_{t=0}^{T-1} \xi_t(i,j)}{\sum_j \sum_{t=0}^{T-} \xi_t(i,j)} = \frac{\sum_{t=0}^{T-1} \xi_t(i,j)}{\sum_{t=0}^{T-1} \rho_t(i)} \tag{1.100}$$

$$B^*_{ik} = \frac{\sum_{t=0}^{T} \rho_t(i) \mathbb{1}[y_t = k]}{\sum_{t=0}^{T} \rho_t(i)}. \tag{1.101}$$

Alternate the E and M steps.

(Historical note: Baum was the first collaborator of Jim Simons. Viterbi started Qualcomm. CDMA technology used Viterbi.)

5.3 Continuous HMM

For a continuous HMM,

$$dX_t = f(X_t;\theta^f)\,dt + \sigma\,dW_t \tag{1.102}$$
$$dY_t = h(X_t;\theta^h)\,dt + \eta\,dB_t. \tag{1.103}$$

Suppose $p(X_0)$ is known and we observe y_t, $0 \le t \le T$. The MLE is

$$L(\theta) = \log p(y_{0:T};\theta) = \log \int p(x_{0:T}, y_{0:T};\theta)[Dx_{0:T}] \tag{1.104}$$

assuming we can do the integral over all paths.

For the E-step, we need to calculate something similar to the ρ_t.

$$\rho_t(x) = p(X_t = x|\mathcal{F}_T^Y) \tag{1.105}$$
$$\mathcal{F}_t^Y = \sigma(Y_s, s \le t) \tag{1.106}$$

We need something beyond the Kushner and Zakai equations. The Zakai equation is

$$\pi_t(x) = p(X_t = x|\mathcal{F}_t^Y)Z_t \tag{1.107}$$
$$Z_t = \mathbb{E}_Q[M_t|\mathcal{F}_t] \tag{1.108}$$
$$\pi_0(x) = p(X_0 = x) \tag{1.109}$$
$$d\pi = \mathscr{L}^*\pi_t(x)\,dt + \frac{1}{\eta^2}h(x)\pi(x)\,dY_t. \tag{1.110}$$

To get ρ_t, use Zakai smoothing.

$$\rho_t(x) = \frac{\pi_t(x)\kappa_t(x)}{Z_T} \tag{1.111}$$
$$\kappa_t(x) = \mathbb{E}_Q[M_T/M_t|\mathcal{F}_T^Y \cup [X_t = x]] \tag{1.112}$$
$$M_t = \exp\left(-\frac{1}{2\eta^2}\int_0^t h^2(X_s)\,ds + \frac{1}{\eta^2}\int_0^t h(X_s)\,dY_s\right) \tag{1.113}$$
$$dQ = M_T^{-1}\,dP \tag{1.114}$$
$$\kappa_T(x) = 1 \tag{1.115}$$
$$d\kappa_t(x) = \mathscr{L}\kappa_t(x)\,dt - \frac{1}{\eta^2}h(x)\kappa_t(x)\,dY_t. \tag{1.116}$$

For the M-step, discretizing $t = 0, \Delta t, 2\Delta t, \dots, T$,

$$\max_\theta \int p(x_{0:T}|y_{0:T}, \theta_k) \tag{1.117}$$

$$\log p(x_{0:T}, y_{0:T}; \theta) \tag{1.118}$$

$$p(x_{0:T}, y_{0:T}|\theta) = p(x_0) \prod_{t=0}^{T-\Delta t} p(\Delta x_t|x_t, \theta^f) \prod_{t=0}^{T} p(\Delta y_t|x_t; \theta^h) \tag{1.119}$$

$$p(\Delta x_t|x_t; \theta^f) = \frac{1}{Z_f} \exp\left[-\frac{(\Delta x_t - f(x_t; \theta^f)\Delta t)^2}{2\sigma^2 \Delta t}\right] \tag{1.120}$$

$$p(\Delta y_t|x_t, \theta^h) = \cdots \tag{1.121}$$

$$\int p(x_{0:T}|y_{0:T}, \theta_k) = \int p(x_{0:T}|y_{0:T}; \theta_k) \left[\log p(x_0) + \sum_{t=0}^{T-\Delta t} \log p(\Delta x_t|x_t; \theta^f) + \sum_{t=0}^{T} \log p(\Delta y_t|x_t; \theta^h)\right] dx_{0:T} \tag{1.122}$$

$$= C + \sum_{t=0}^{T-\Delta t} \int p(x_t, \Delta x_t|y_{0:T}; \theta_k) \log p(\Delta x_t|x_t; \theta^f)\, dx_t\, dx_{t+\Delta t} \tag{1.123}$$

$$+ \sum_{t=0}^{T} \int p(x_t|y_{0:T}; \theta_k) \log p(\Delta y_t|x_t; \theta^h)\, dx_t \tag{1.124}$$

Taking the limit of $\sum_{t=0}^{T-\Delta t} \int p(x_t, \Delta x_t|y_{0:T}; \theta_k) \log p(\Delta x_t|x_t; \theta^f)\, dx_t\, dx_{t+\Delta t}$ gives

$$C + \int_\Omega dx \int_0^T \rho_t(x; \theta_k) \left[f^2(x; \theta^f) - 2f(x; \theta^f) \frac{d}{d\tau}|_{\tau=0} \mathbb{E}[X_{t+\tau}|\mathcal{F}_T^Y \cup [X_t = x]; \theta_k]\right] dt \tag{1.125}$$

$$+ \int_0^T dx \int_0^T \rho_t(x, \theta_k)[h^2(x; \theta^h)\, dt - 2h(x; \theta^j)\, dY_t]\, dt \tag{1.126}$$

(Note the exponent of $p(\Delta x_t|x_t; \theta^f)$ becomes $\left(\frac{dx}{dt}\right)^2 - 2f\, dx + f^2\, dt$.) Also

$$\frac{d}{d\tau}|_{\tau=0} \mathbb{E}[X_{t+\tau}|\mathcal{F}_T^Y \cup [X_t = x]; \theta_k] = f(x; \theta_k) + \frac{2x\mathscr{L}\kappa_t(x) + \sigma^2 \partial_x \kappa_t(x)}{\kappa_t(x)} - \frac{xh^2(x)}{\eta(x)^2} \tag{1.127}$$

5.4 Continuous time jump process

We assume X is a a jump process.

The E step is the same. For the M step we still get (1.124). We get

$$p(x_t, x_{t+\Delta t}|y, Q_k) \log p(\underbrace{x_{t+\Delta t}}_{j} | \underbrace{x_t}_{i}, Q)\, dx_t\, dx_{t+\Delta t} \tag{1.128}$$

$$Q^*_{ij} = \frac{\int_0^T \frac{d}{d\tau}|_{\tau=0} p(x_t = i, x_{t+\tau} = j|y; Q_k)\, dt}{\int_0^T \rho_t(i)\, dt} \tag{1.129}$$

Chapter 2

Reinforcement learning

For the rest of the weeks, we'll cover the 10 lectures in David Silver's course on reinforcement learning (compressed). Questions:

1. What happens in continuous time? Things simplify.

2. Math results that have been proved. Q-learning, policy gradient, value iteration

3. What happens when the state space is huge? Ex. sophisticated exploration techniques like Monte Carlo.

4. The formulation has a well-defined reward function and discount factor. This is not necessarily the right formulation for decision-making problems. Optimal decision-making is the ultimate goal. But everything is about the limited formulation.

Understand the SoTA for situations for situations for which the state-space (and maybe action space) is high-dimensional, and understand what's known mathematically.

1 2019-4-4 Introduction

I'll start with the most important topic: what are good examples?

1. Flying stunts (see Andrew Ng on flying a drone). As someone with a background in modeling physical system, I'd write down a PDE, how the turbines interact wth the flows, how to control the turbines so that it flies. If you learn how to ride a bike, I'd model the details of how it interacts with the surface.

   But we never thinik about the detailed interactions; I just try it; if I'm about to fall I change. I'm going through a model-free reinforcement learning procedure.

2. Games. This is a good example.

3. Investment portfolio. I'm not convinced. The main difference from a supervised learning problem is a time lag (I don't have iid examples; it's a dynamical system, I apply actions and later I get a reward). Aside from this there's no difference.

   Suppose you want to liquidate a huge amount of stock. If you sell it immediately, then you impact the market a lot.

4. Control of power station

5. Robot

I'd like to see more convincing examples. It's not convincing that it has caused such a big spectacle in China and the US.

How do we come up with good really convincing examples of reinforcement learning?

We're working for the cause of RL. It has to be compelling enough for us to spend time on.

David Silver uses the terminology

- RL for model-free learning
- planning for model-based learning.

We have a dynamical system with control, and assume it's a fully observed system, which means both the environment and the system we're interested in is fully observed (Markov model).

We could also assume a partially observed system. This is a hidden Markov model.

Find the simplest setting where all the critical issues are present: high-dimensionality, etc. I will work with deterministic continuous dynamical systems, for which everything can be written as PDE's.

1.1 Discrete setting

We need to discuss Markov process, Markov reward process, and Markov decision process.

1. Markov process: (S, P) where S is state space and P is transition probabilities. $P_{ss'} = P(S_{t+1} = s'|S_t = s)$.

2. Markov reward process: (S, P, R, γ) where R_{t+1} is a random variable of reward at time $t+1$ (depends on S_t)

   The total return is $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$.

   The reward of state s is $R_s = \mathbb{E}[R_{t+1}|S_t = s]$.

   This framework assumes you know the reward.

   (Riding a bike: don't need to know details of whole process. The policy function in stored in our brain.)

The value function is

$$\begin{aligned}
v(S) &= \mathbb{E}[G_t | S_t = s] && (2.1)\\
&= \mathbb{E}[R_{t+1} + \gamma R_{t+1} + \gamma^2 R_{t+3} + \cdots | S_t = s] && (2.2)\\
&= \mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) | S_t = s] && (2.3)\\
&= \mathbb{E}[R_{t+1} | S_t = s] + \gamma \mathbb{E}[v(S_{t+1}) | S_t = 1] && (2.4)\\
&= R_s + \gamma \sum_{s'} P_{ss'} v(s') && (2.5)
\end{aligned}$$

The Bellman equation follows from the Markovian property. We can write this as $(I - \gamma P)v = R$; solving this linear system gives $w = (I - \gamma P)^{-1} R$. Doing the iteration

$$v_{k+1}(s) = R_s + \gamma \sum P_{ss'} v_k(s') \tag{2.6}$$

this converges with error $\|e_{k+1}\| \le \gamma \|e_k\|$ for $\gamma < 1$.

3. Markov decision process: (S, A, P, R, γ) where the outcome depends on the action. We now have $P_{ss'}^a = \mathbb{P}(S_{t+1} = s | S_t = s, A_t = a)$ and $R_s^a = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$.

   The policy is $\pi(a|s) = \mathbb{P}(A_t = a | s_t = s)$. This describes the behavior of agents.

   We can define

$$\begin{aligned}
P_{ss'}^\pi &= \sum_a \pi(a|s) P_{ss'}^a && (2.7)\\
R_s^\pi &= \sum_a \pi(a|s) R_s^a && (2.8)\\
v_\pi(s) &= \mathop{\mathbb{E}}_\pi [G_t | S_t = s] && (2.9)\\
q_\pi(s, a) &= \mathop{\mathbb{E}}_\pi [G_t | S_t = s, A_t = a] && (2.10)
\end{aligned}$$

   The q function is the value at state s, taking action a, and then following π. (From $t + 1$ onwards, follow π.) We derive (linear) Bellman equations:

$$\begin{aligned}
v_\pi(s) &= \sum_a \pi(a|s) q_\pi(s, a) && (2.11)\\
v_\pi(s) &= R_s^\pi + \gamma \sum_{s'} P_{ss'}^\pi v_\pi(s') && (2.12)\\
q_\pi(s, a) &= R_\pi^a + \gamma \sum_{s'} P_{ss'}^\pi v_\pi(s') && (2.13)\\
q_\pi(s, a) &= R_s^a + \gamma \sum_{s'} P_{ss'}^a \sum_{a'} \pi(a'|s') q_\pi(s', a') && (2.14)
\end{aligned}$$

[1]

[1] There is a deterministic optimal policy. But in practice, when we play games, often we follow a probabilistic policy.

Related? Nash equilibrium is a mixed strategy (but this does not fall into this framework—how to modify so it does?).

When I say "optimal policy" I'm assuming there exists a policy with which the value function for all the states is maximized. It can be shown that an optimal policy does exist. It exists by dynamic programming. You can define a partial order in the space of policies, it has a maximum.

$$v_*(s) = \max_\pi v_\pi(s) = \max_{(\pi,a)} q_\pi(s,a) \tag{2.15}$$

$$q_*(s,a) = \max_\pi q_\pi(s,a) \tag{2.16}$$

$\pi^*(a|s)$ is supported on $\operatorname{argmax} q_*(s,a)$. If it's unique, we must have $\pi_*(a|s) = \begin{cases} 1, & a = \operatorname{argmax}_a q_*(s,a) \\ 0, & \text{else} \end{cases}$.
The way we find optimal policies is to do this operation.

I write down Bellman equations for the optimal policy.

$$v_*(s) = \max_a \left(R_s^a + \gamma \sum_{s'} P_{ss'}^a v_*(s') \right) \tag{2.17}$$

$$q_*(s,a) = R_s^a + \gamma \sum_{s'} P_{ss'}^a \max_{a'} q_*(s',a'). \tag{2.18}$$

How to solve this? Note that in the q function R_s^a is out of the max; this is another reason why the q function is useful.

Once we solve this, the optimal policy is given by the argmax of q_*.

Obvious methods to solve: policy iteration (do maximization at each step). This can be proven to converge. The problem is that if the state space is so large that you cannot even do one iteration; you have to do Monte Carlo sampling. This is the situation we're interested in.

1.2 Continuous setting

I want to discuss the ODE setting, and derive the Hamiltonian-Jacobi-Bellman equations in this setting. In the continuous setting, a Markov process is an ODE, $z(0) = x$, $\frac{dz}{dt} = f(z)$. For a Markov reward process, cumulative reward is

$$v(x) = \int_0^\infty e^{-\gamma t} R(z(t))\, dt \tag{2.19}$$

$$v(z(t)) = \int_t^\infty e^{-\gamma(s-t)} R(z(s))\, ds \tag{2.20}$$

Differentiating,

$$\nabla v(z(t)) f(z(t)) = \nabla v \cdot \frac{dv}{dt} = \frac{dv}{dt} = \gamma e^{\gamma t} \int_t^\infty e^{-\gamma s} R(z(s))\, ds - e^{\gamma t} e^{-\gamma t} R(z(t)). \tag{2.21}$$

At $t = 0$,

$$\nabla v(x) \cdot f(x) = \gamma v(x) - R(x). \tag{2.22}$$

It's the analogue of the first Bellman equation.

Now let's look at the analogue of a decision process. Now $\frac{dz}{dt} = f(z, a)$. The policy is a function of z, $a = \pi(z)$. The dynamical system becomes $\frac{dz}{dt} = f(z, \pi(z))$. (For simplicity we consider a deterministic policy. We could also consider $\pi(a|s)$.)

The q-function for the discrete setting is: take one step with a, then follow π. In the continuous setting, how do we define it? Follow a for time Δt, and then follow π. Let

$$\widetilde{a}(t) = \begin{cases} a, & 0 \le t \le \Delta t \\ \pi(z(t)), & t \ge \Delta t \end{cases} \tag{2.23}$$

$$q_\pi(x, z) - v_\pi(x) = \Delta t Q_\pi(x, a) + o(\Delta t). \tag{2.24}$$

The difference is the sum of the differences on the two intervals. On the interval Δt, the difference is also Δt, so the contribution is $O(\Delta t^2)$, negligible.

$$\int_0^{\Delta t} e^{-\gamma t} \underbrace{(R_\pi(\widetilde{z}_\pi(t)) - R(z_\pi(t)))}_{\approx \nabla R \cdot (\widetilde{z}_\pi - z)} \, dt. \tag{2.25}$$

(If there is dependence on a, there is another term $\Delta t(R(x, a) - R(x, \pi(x)))$.) We need to calculate

$$\int_{\Delta t}^\infty e^{-\gamma t}(R(\widetilde{z}_\pi(t)) - R(z_\pi(t))) \, dt. \tag{2.26}$$

The initial condition at Δt differs by $\Delta t(f(x, a) - f(x, \pi(x)))$:

$$\widetilde{z}_\pi - z_\pi = \Delta t(f(x, a) - f(x, \pi(x)))(\nabla_x z)(t) \tag{2.27}$$

$$\text{where } \frac{dz}{dt} = f(z, \pi(z)) \tag{2.28}$$

At the end we have

$$Q(x, a) = R(x, a) - R(x, \pi(x)) + \int_0^\infty (f(x, a) - f(x, \pi(x))) \underbrace{\nabla_x z(t)}_{\delta z} e^{-\gamma t} \nabla R(z_\pi(t)) \, dt. \tag{2.29}$$

Taking an infinitesimal perturbation of the original path, the difference between the two is the gradient $\nabla z_x(t)$. "Backpropagation."

$$\frac{d\delta_z}{dt} = \nabla_x \frac{dz}{dt} = \nabla_x f(z, \pi(z)) = \nabla_z f \delta z + \nabla_a f \nabla_z \pi \delta z \tag{2.30}$$

$$= (\nabla_z f + \nabla_a f \nabla_z \pi) \, \delta z \tag{2.31}$$

...

$$= (f(x, a) - f(x, \pi(x))) \nabla v(x) \tag{2.32}$$

$$= f(x, a) \nabla v - f(x, \pi(x)) \nabla v. \tag{2.33}$$

If you want to maximize something you need to maximize $f(x,a)\nabla v$.

Balance between short and long term.

This is the Q-function for a fixed policy. Let's derive the Q function for the optimal policy. ($\frac{dz^*}{dt} = f(z^*, \pi^*(z^*))$)

$$v_*(x) = \max_\pi v_\pi(x) = \max \int_0^\infty e^{-\gamma t} R(z_*(t))\, dt \tag{2.34}$$

To leading order

$$x_a = x + \Delta t f(x,a) \tag{2.35}$$

so

$$v_*(x) = \max_{a(t)} \left(\int_0^{\Delta t} e^{-\gamma t} R(z(t), a(t))\, dt + \int_{\Delta t}^\infty e^{-\gamma t} R(z(t), a(t))\, dt \right) \tag{2.36}$$

We have

$$\int_{\Delta t}^\infty e^{-\gamma t} R(z(t), a(t))\, dt = e^{\gamma \Delta t} \int_0^\infty e^{-\gamma t} R(z(t+\Delta t), a(t+\Delta t))\, dt \tag{2.37}$$

so

$$v_*(x) = \max_{a(t)} \left(\int_0^{\Delta t} e^{-\gamma t} R(z(t), a(t))\, dt + e^{\gamma \Delta t} v_*(x_a) \right) \tag{2.38}$$

Using $z(t) = x + tf(x,a)$,

$$\max_a \left(\int_0^{\Delta t} e^{-\gamma t} R(z(t), a)\, dt + e^{\gamma \Delta t} v_*(x_a) \right) = \max_a \left(\Delta t R(x,a) + (e^{\gamma \Delta t} - 1) v_*(x + \Delta t f(x,a)) \right) \tag{2.39}$$

$$v_*(x + \Delta t f(x,a)) = v_*(a) + \Delta t \nabla v_* f(x,a) \tag{2.40}$$

$$\max_a \left(R(x,a) - \gamma v_*(x) + \nabla v_*(x) f(x,a) \right) = 0 \tag{2.41}$$

$$\gamma v_*(x) = \max_a \left(R(x,a) + \nabla v_*(x) f(x,a) \right). \tag{2.42}$$

Defining

$$H(p,x) := \max_x (R(x,a) + p \cdot f(x,a)), \tag{2.43}$$

we get

$$\gamma v_*(x) = H(\nabla v_*, x). \tag{2.44}$$

Our dynamical system is $\frac{dz}{dt} = f(z,a)$. Imagine it describes the Tokamak where z is applied magnetic field. This is an unstable process. The system is billions of dimensions. The magnetic field—there are many knobs. This really depends on technology but it also depends on your ability to control. If you can't control a million knobs, then having them there doesn't help. Just to evaluate the Hamiltonian at one value is already costly. This is the kind of problem we should have in mind. Can we handle this kind of problem?

Turbulence: apply polymer to wing to reduce drag. We don't know how to apply control to reduce the drag. Any industrial processing problem is a problem of this type.

Maote - strong liquor in China. Experience-based. When old masters retire, experience retires with them. They can put a lot of sensors, control moisture. Reward function is also experience-based!

If you want to produce the best-quality steel, manufacturing process—temperature, etc. Silification (?) process, complicated PDE.

Before we couldn't imagine solving these kinds of problems. When numerical algorithms could only handle 3 variables, talking about this is useless.

How do we solve this PDE? We can't put a grid in a billion-dimensional space.

1.3 4-11

$$v(s) = \mathbb{E}(R + \gamma v(s')) \tag{2.45}$$
$$= R + \gamma \sum_{s'} P_{ss'} v(s') \tag{2.46}$$
$$v_\pi(s) = R^\pi(s) + \gamma \sum_{s'} P^\pi_{ss'} v_\pi(s') \tag{2.47}$$
$$Q(s,a) = \mathbb{E}(R + \gamma Q(s',a')|s,a) \tag{2.48}$$
$$Q_\pi(s,a) = \mathbb{E}(R + \gamma \max_{a'\in A} Q(s',a')|s,a) \tag{2.49}$$
$$v_t(s) = \mathbb{E}(R + \gamma \max_a \sum Q_*(s',a)|s). \tag{2.50}$$

We are trying to solve the Bellman equation, in two different contexts:

1. Dynamic programming (model-based): The simplest iterative methods for solving these equations are the following

   Iterative Policy Evaluation:

$$v(s) \hookleftarrow \mathbb{E}(R + \gamma v(s')|s) \tag{2.51}$$
$$v_{k+1}(s) = R(s) + \gamma \sum_{s'} v_k(s') P_{ss'} \tag{2.52}$$

   Q Policy iteration:

$$Q(s,a) \hookleftarrow \mathbb{E}(R + \gamma Q(s',a')|s,a) \tag{2.53}$$
$$(Q_{k+1}(s,a) = R(s,a) + \gamma \sum_{s',a'} P^a_{s,s'} Q_k(s',a')) \tag{2.54}$$
$$Q_{k+1}(s,a) = \mathbb{E}(R + \gamma \max_{a'} Q_k(s',a')|s,a) \tag{2.55}$$

2. Model-free: Sample the outcomes

$$v_{k+1}(s) = R(s) + \gamma v_k(S_{t+1}) \tag{2.56}$$
$$Q_{k+1}(s,a) = R(s,a) + \gamma Q_k(s_{t+1}, a_{t+1}) \tag{2.57}$$
$$Q^*_{k+1}(s,a) = R(s,a) + \gamma \max_{a'} Q^*_k(s_{t+1}, a) \tag{2.58}$$
$$\text{or } Q_{k+1}(s,a) = \alpha\left(\mathbb{E}(R + \gamma \max_{a'} Q^*_k(s_{t+1}, a')|s,a)\right) + (1-\alpha)Q^*_k(s,a). \tag{2.59}$$

   If $\alpha = 1$ this may not converge. α is the relaxation parameter. (This is classical. One of the classical methods is overrelaxation, when $\alpha \notin [0,1]$.)

Question: Do you consider AlphaGo model-based or model-free?

Those are simple methods. Next time we'll talk about more sophisticated methods: Actor-critic, policy gradient.

First we'll consider the continuous analogue. Last time we derived

$$\gamma v_*(x) = H(x, \nabla v_x) \tag{2.60}$$

This is a first order scalar nonlinear PDE.

$$H(x,p) = \max_a \{R(x,a) + f(x,a)p\} \tag{2.61}$$
$$Q_\pi(x,a) = R(x,a) + R(x,\pi(x)) + (f(x,a) - f(x,\pi(x)))\nabla v_\pi(x). \tag{2.62}$$

You can solve this by

1. the method of characteristics
2. and the maximal principle (Pontryagin) in control theory
3. variational formulation

They're all the same thing, in different forms. The equation was defined by a variational formulation.

I'll discuss how to solve the cleaner problem (Hamilton-Jacobi equation).

$$\partial_t u + H(x, \nabla u) = 0 \tag{2.63}$$
$$u(x,0) = u_0(x). \tag{2.64}$$

The variational formulation is (least action principle in mechanics)

$$u(z,t) = \inf_{x(0)=y, x(t)=z} \left\{u_0(y) + \int_0^t L(x(s), \dot{x}(s))\, ds\right\}, \tag{2.65}$$

where L is the Lagrangian.

$$L(x,\dot{x}) = \sup_p(\dot{x}p - H(x,p)) \tag{2.66}$$
$$H(x,p) = \sup_{\dot{x}}(p\dot{x} - L(x,\dot{x})) \tag{2.67}$$

Usually we start with a Lagrangian; Hamilton-Jacobi theory says that u satisfies the Hamilton-Jacobi equation. Here we go backward from the HJ equation to derive the least action principle in mechanics and define the Hamiltonian.

Look at all the path going backwards and calculate the action; that gives the solution.

In principle there should not be exponential dependence for calculating the value at one point (z, t). This is an ODE problem.

Write down the Euler-Lagrange equation, that's the method of characteristics. Pontrayagin maximal principle is more general. a is dual to p, $\dot{x}$ is dual to p. In RL, $\dot{x}(s)$ can be discrete; equation (2.61) still works.

In my view the simplest way to solve (2.61) is to pick a sample point and compute the solution by solving (2.65), and parameterize your policy using deep learning. The algorithm discussed is Jacobi method; this is at least as good as conjugate gradient.

If you want to solve $Ax = b$, write A as $D + (L + U)$ and iterate:

$$(D + (L + U))x = b \tag{2.68}$$

$$Dx_{k+1} = (L + U)x_k + b. \tag{2.69}$$

A simple situation: what if H is independent of x?

$$H(x, p) = H(p) \tag{2.70}$$

For example, in classical mechanics this is $\frac{1}{2}|p|^2$. (Classical mechanics without a force.) Then in (2.65), the integral becomes

$$\int_0^t L(\dot{x}(s))\, ds. \tag{2.71}$$

The solution is $x(s) = y + s\frac{z-y}{t}$, $\dot{x}(s) = \frac{z-y}{t}$. The solution is

$$u(z, t) = \inf_y \left\{ u_0(y) + tL\left(\frac{z-y}{t}\right)\right\}. \tag{2.72}$$

This is Huygen's principle.

To solve the problem there are 2 ideas.

1. Indirect method (method of characteristics). Solve

$$\frac{dz}{dt} = \nabla_p H \tag{2.73}$$

$$\frac{dp}{dt} = -\nabla_x H \tag{2.74}$$

$$x(0) = z \tag{2.75}$$

$$p(0) = \nabla u_0(y) \tag{2.76}$$

   Usually the method of characteristics goes forward. The variational formulation goes backwards, look at all the paths which reach (z, t).

   (2.65) is a boundary value problem; the indirect method is an initial value problem. Use a "shooting method". Iteratively try different values of y to try to end up at (z, t).

2. The direct method is to solve the variational problem.

In both cases I need to discuss 2 problems: how to discretize, and how to accelerate convergence.

To discretize, use finite element methods, spectral methods, pseudo-spectral methods, or collocation methods.

Spectral element method: divide into intervals, apprxomate on each with polynomial. A vector function of 1 variable, no curse of dimensionality. If the interval is large, convergence is slow. It's too far to shoot, especially when the time horizon is infinite. The idea is to do parallel shooting. Divide into smaller intervals. Target in between is to make sure trajectory is continuous and differentiable.

The second idea which is not implemented seriously is to use multigrid.

For model-free there's not much you can do, observe data coming in. You won't do better than Monte Carlo. The biggest obstacle is that it's not as good because the data is correlated. If it's model-based there's a lot more one can do. The room for improvement is in the model-based.

AlphaGo is model-free (?). In principle we can make Go model-based. Backgammon: use supervised learning to learn moves, give to opponent.

The room is in model-based; develop simulation platforms. If you don't have simulation platforms... I don't think investment is a good example, if you just wait for data to come in, it's not efficient. You need to model and simulate the market.

Since we can make Go model-based, can we do better than AlphaGo? Model-free is easy, just use Jacobi. Is there room to improve with a model?

The RL community is heavily model-free.

Policy-gradient method: Policy-gradient theorem. It's like the Hellman-Feynman theorem.

We want to compute $\frac{\delta v_\pi(x)}{\delta \pi}$, to maximize $\max_\pi v_\pi(x)$.

$$v_\pi = \int_0^\infty e^{-\gamma t} R(z_\pi(t), \pi(z_\pi(t)))\, dt \tag{2.77}$$

Taking the gradient, the dependence on z vanishes. We have $(\nabla_z R + \nabla_u R \nabla_z \pi) \frac{\delta z_\pi}{\delta \pi}$.

Pontryagin maximal principle:

$$\frac{dx}{dt} = f(x,u) \tag{2.78}$$

$$\frac{dp}{dt} = -\nabla_x H \tag{2.79}$$

$$\inf_{x(0)=y, x(t)=x} \left\{ u_0(y) + \int_0^t L(x, \dot{x}, u)\, dx \right\} \tag{2.80}$$

$$H(x,p,u) = \sup\{p\dot{x} - L(x, \dot{x}, u)\} \tag{2.81}$$

$$x(t) = x \tag{2.82}$$

$$p(0) = -\nabla u_0(x(0)). \tag{2.83}$$

The action is given by this maximal principle

$$u(s) = \text{argmax}_u\, H(x(s), p(s), u) \tag{2.84}$$

This maximization is done independently for each time. Naively, you do the following iteration; solve the adjoint system

$$\dot{x}_{k+1} = f(x_{k+1}, u_k) \tag{2.85}$$

$$\dot{p}_{k+1} = \nabla_x H(x_{k+1}, p_{k+1}, u_k) \tag{2.86}$$

$$0 \leq s \leq t \quad u_{k+1}(s) = \text{argmax}_u\, H(x_{k+1}(s), p_{k+1}(s), u) \tag{2.87}$$

This is MSA, method of successive approximation. This is the basic idea; you can use shooting, multigrid, introduce relaxation, etc.