Gradient Boosting for Inverse Problems

Yuri F. Saporito

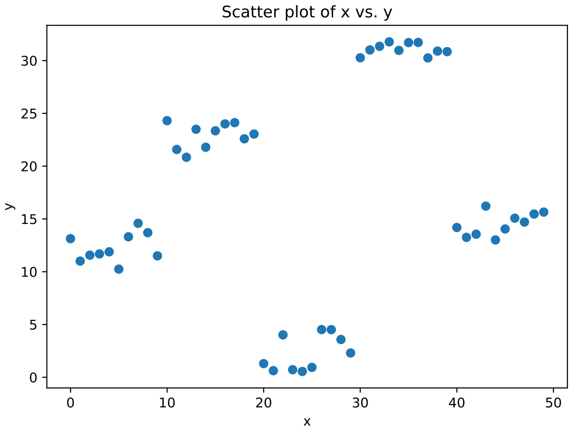
FGV EMAp
ESCOLA DE MATEMÁTICA APLICADA

Workshop on Data Science
April 1st to 5th, 2019

The first references of gradient boosting are

- Friedman, "Greedy function approximation: A gradient boosting machine", The Annals of Statistics **29** (5), 2001
- Mason, Baxter, Bartlett and Frean, "Boosting algorithms as gradient descent", Proceeding NIPS, 1999

Idea



We want to find $f^* = \arg\min_f \sum_{i=1}^{n}(y_i - f(x_i))^2$

Idea

- To approximate $f^* = \arg\min_f \sum_{i=1}^{n}(y_i - f(x_i))^2$ we use the idea of sequentially improve a initial, simple f_0 adding other simple estimators
- For instance, we could use Decision Trees with one node or very simple neural networks
- After the initial estimator f_0, we compute the residual $y_i - f_0(x_i)$ and find f_1 that such that

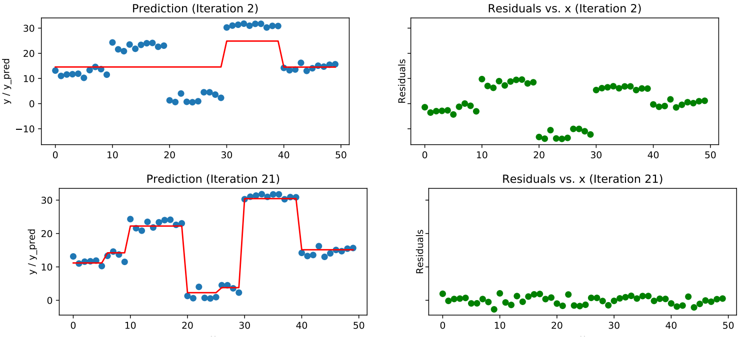
$$f_1 = \arg\min_h \sum_{i=1}^{n}(y_i - f_0(x_i) - h(x_i))^2$$

- Repeat until the total error is small. The final estimator will be given by

$$f^*(x) \approx f_0(x) + \sum_{i=1}^{k} \nu f_i(x),$$

where ν is a learning rate

Idea


Prediction (Iteration 2)
Residuals vs. x (Iteration 2)
Prediction (Iteration 21)
Residuals vs. x (Iteration 21)
y / y_pred
Residuals
30
20
10
0
−10
0
10
20
30
40
50

Let's formalize the ideas we have just seen:

- $\mathbb{X}$ denotes the input space and $\mathbb{Y}$, the output space
- $\mathbf{X}$ is the random input and $\mathbf{Y}$, the random output, both defined in a common probability space $(\Omega, \mathcal{F}, \mathbb{P})$
- their joint probability density by p; p_X and p_Y denote the marginal distributions
- $L : \mathbb{Y} \times \mathbb{Y} \longrightarrow \mathbb{R}$ is a point-to-point loss function and:

$$\mathcal{L}(f) = \mathbb{E}_{X,Y}[L(\mathbf{Y}, f(\mathbf{X}))] = \int L(\mathbf{y}, f(\mathbf{x}))p(\mathbf{x}, \mathbf{y})d\mathbf{x}d\mathbf{y},$$

for $f : \mathbb{X} \longrightarrow \mathbb{Y}$

Problem

- We are considering
$$f^* = \arg\min_{f \in Lin(\mathcal{S})} \mathcal{L}(f)$$

- $\mathcal{S}$ is the space of the so-called weak learners (or boosts), which are functions from $\mathbb{X}$ to $\mathbb{Y}$ and
$$Lin(\mathcal{S}) = \left\{ \sum_{j=1}^{m} w_j f_j \;;\; f_j \in \mathcal{S},\; w_j \in \mathbb{R},\; m \in \mathbb{N} \right\}$$

- Given $f \in Lin(\mathcal{S})$, we want to choose $\Delta f^* \in \mathcal{S}$ such that we decrease the loss function $\mathcal{L}$ as much as we can

- Clearly the desired direction must be the negative of the gradient of $\mathcal{L}$ at f

Idea - Finite-Dimensional

- Reminder: if $\mathcal{L} : \mathbb{R}^n \longrightarrow \mathbb{R}$, then the directional derivative of $\mathcal{L}$ at f in the direction of Δf can be written $D\mathcal{L}(f)(\Delta f) = \nabla\mathcal{L}(f) \cdot \Delta f$, where $\nabla\mathcal{L} = (\partial_1\mathcal{L}, \ldots, \partial_n\mathcal{L})$ and $\cdot$ is the inner product in $\mathbb{R}^n$
- $\mathcal{L}$ decreases the most in the direction of $\Delta f = -\nabla\mathcal{L}(f)$

- Our setting is infinite-dimensional: the directional derivative of $\mathcal{L}$ at f in the direction of Δf is defined as:

$$D\mathcal{L}(f)(\Delta f) = \lim_{\varepsilon \to 0} \frac{\mathcal{L}(f + \varepsilon \Delta f) - \mathcal{L}(f)}{\varepsilon}$$

- Under mild assumptions, the gradient of $\mathcal{L}$, denoted by $\nabla \mathcal{L}(f)$ is a function in $L^2(\mathbb{X}, p_X)$ that satisfies, for all Δf,

$$D\mathcal{L}(f)(\Delta f) = \int_{\mathbb{X}} \nabla \mathcal{L}(f)(\mathbf{x}) \Delta f(\mathbf{x}) p_X(\mathbf{x}) d\mathbf{x} = \langle \nabla \mathcal{L}(f), \Delta f \rangle$$

- Hence, in some sense,

$$\nabla \mathcal{L}(f)(\mathbf{x}_0) p_X(\mathbf{x}_0) = D\mathcal{L}(f)(\delta_{\mathbf{x}_0}) = \lim_{\varepsilon \to 0} \frac{\mathcal{L}(f + \varepsilon \delta_{\mathbf{x}_0}) - \mathcal{L}(f)}{\varepsilon}$$

- What is the gradient of $\mathcal{L}$ in our infinite-dimensional case?
- Notice that (assuming certain regularity)

$$\begin{aligned} D\mathcal{L}(f)(\Delta f) &= \lim_{\varepsilon \to 0} \frac{\mathcal{L}(f + \varepsilon \Delta f) - \mathcal{L}(f)}{\varepsilon} \\ &= \mathbb{E}\left[\lim_{\varepsilon \to 0} \frac{L(\mathbf{Y}, f(\mathbf{X}) + \varepsilon \Delta f(\mathbf{X})) - L(\mathbf{Y}, f(\mathbf{X}))}{\varepsilon}\right] \\ &= \mathbb{E}_{X,Y}[\partial_2 L(\mathbf{Y}, f(\mathbf{X}))\Delta f(\mathbf{X})], \end{aligned}$$

where $\partial_2 L$ denotes the derivative with respect to the second argument of L. Then

$$D\mathcal{L}(f)(\Delta f) = \int_{\mathbb{X}} \mathbb{E}_{Y|X}[\partial_2 L(\mathbf{Y}, f(\mathbf{X}))\Delta f(\mathbf{X}) \mid \mathbf{X} = \mathbf{x}] p_X(\mathbf{x}) d\mathbf{x}$$

$$= \int_{\mathbb{X}} \underbrace{\mathbb{E}_{Y|X}[\partial_2 L(\mathbf{Y}, f(\mathbf{x})) \mid \mathbf{X} = \mathbf{x}]}_{= \nabla \mathcal{L}(f)(\mathbf{x})} \Delta f(\mathbf{x}) p_X(\mathbf{x}) d\mathbf{x} = \langle \nabla \mathcal{L}(f), \Delta f \rangle$$

- Hence $\nabla\mathcal{L}(f)(\mathbf{x}) = \mathbb{E}_{Y|X}[\partial_2 L(\mathbf{Y}, f(\mathbf{x})) \mid \mathbf{X} = \mathbf{x}]$
- Notice that $\nabla\mathcal{L}(f)$, viewed as a function on $\mathbb{X}$, might not belong to $\mathcal{S}$
- Therefore, we should choose $\Delta f^* \in \mathcal{S}$ as similar as possible to $\nabla\mathcal{L}(f)$, i.e. choose $\Delta f^* \in \mathcal{S}$ that maximizes

$$-\langle \nabla\mathcal{L}(f), \Delta f \rangle = -D\mathcal{L}(f)(\Delta f) = -\int_{\mathbb{X}} \nabla\mathcal{L}(f)(\mathbf{x}) \Delta f(\mathbf{x}) p_X(\mathbf{x}) d\mathbf{x}$$

- We then improve f by considering $f + \nu\Delta f^*$, where ν is a learning rate
- The Gradient Boosting algorithm is the iteration of the steps above starting from a given $f_0 \in \mathcal{S}$ and stopping when $-\langle \nabla\mathcal{L}(f_m), \Delta f \rangle \leq 0$ for all $\Delta f \in \mathcal{S}$

Convergence

Let $(f_m)_{m\in\mathbb{N}}$ be a sequence generated by the Gradient boosting algorithm

Assumption

- $\mathcal{L} : Lin(\mathcal{S}) \longrightarrow \mathbb{R}$ *is convex, lower bounded and Lipschitz differentiable loss functional, i.e. there exists* B *and* L *positive numbers such that* $\mathcal{L}(f) \geq B$ *and* $\|\nabla\mathcal{L}(f) - \nabla\mathcal{L}(g)\| \leq L\|f - g\|$*, for any* $f, g \in Lin(\mathcal{S})$
- $f \in \mathcal{S} \Rightarrow -f \in \mathcal{S}$ *and the algorithm always finds the function* $\Delta f_m \in \mathcal{S}$ *that maximizes* $-\langle\nabla\mathcal{L}(f_{m-1}), \Delta f\rangle$

Theorem

If the assumptions above are satisfied, then

$$\lim_{m\to+\infty} \sup_{\Delta f\in\mathcal{S}} -\langle\nabla\mathcal{L}(f_m), \Delta f\rangle = 0 \text{ and } \mathcal{L}(h) = \inf_{f\in Lin(\mathcal{S})} \mathcal{L}(f),$$

for any accumulation point h *of* $(f_m)_{m\in\mathbb{N}}$

Given a finite sample $\mathcal{S} = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\}$ from $(\mathbf{X}, \mathbf{Y})$, we use the following approximation

$$\mathcal{L}(f) \approx \frac{1}{N} \sum_{j=1}^{N} L(\mathbf{y}_i, f(\mathbf{x}_i)),$$
$$\nabla \mathcal{L}(f)(\mathbf{x}_i) \approx \partial_2 L(\mathbf{y}_i, f(\mathbf{x}_i)),$$
$$\langle \nabla \mathcal{L}(f), \Delta f \rangle \approx \frac{1}{N} \sum_{i=1}^{N} \partial_2 L(\mathbf{y}_i, f(\mathbf{x}_i)) \Delta f(\mathbf{x}_i)$$

Zhang and Yu, "Boosting with Early Stopping: Convergence and Consistency", The Annals of Statistics **33** (4), 2005

What is an Inverse Problem?

- To understand the type of problems we will try to solve, consider the following simple example

$$-(gu_x)_x = f, \text{ for } x \in (0,1),$$
$$u(0) = a_0 \text{ and } u(1) = a_1,$$

  where f, a_0, a_1 are fixed and known.

- The direct problem is, knowing g, finding u: various numerical methods for that, e.g. finite difference
- The inverse problem is, knowing u, finding g
- What exactly do we mean by "knowing u"? Usually, we know u in a mesh of its domain and there might exist noise in the observation

- Let $\mathcal{G} \subset L^2(\mathbb{X}, p_X)$ be a space of functions and consider an operator $A : \mathcal{G} \times \mathbb{X} \longrightarrow \mathbb{Y}$
- The operator A denotes the direct problem; in the previous example $A(g, x) = u(x)$
- We are interest in solving the following inverse problem related to A: given $\mathbf{x} \in \mathbb{X}$ and $\mathbf{y} \in \mathbb{Y}$, find $g \in \mathcal{G}$ such that $A(g, \mathbf{x}) = \mathbf{y}$ in the sense of minimizing

$$\mathcal{C}(g) = \mathcal{L}(A(g, \cdot)) = \mathbb{E}_{X,Y}[L(\mathbf{Y}, A(g, \mathbf{X}))]$$

- There are various ways of solving this. Here we will modify the Gradient Boosting technique to fit the particularities of this problem

Gradient of $\mathcal{C}$

- We need to compute $\nabla\mathcal{C}$.
- For this, we assume the exists the gradient $\nabla A(g,\cdot) : \mathbb{X} \longrightarrow \mathbb{R}$ in the sense:

$$DA(g,\mathbf{X})(\Delta g) = \int_{\mathbb{X}} \nabla A(g,\mathbf{X})(\mathbf{x})\Delta g(\mathbf{x})p_X(\mathbf{x})d\mathbf{x}$$

- By the chain rule:

$$D\mathcal{C}(g)(\Delta g) = \mathbb{E}_{X,Y}[\partial_2 L(\mathbf{Y}, A(g,\mathbf{X}))DA(g,\mathbf{X})(\Delta g)]$$

- Using the definition of $DA(g,\mathbf{X})(\Delta g)$

$$D\mathcal{C}(g)(\Delta g) = \int_{\mathbb{X}} \mathbb{E}_{X,Y}[\partial_2 L(\mathbf{Y}, A(g,\mathbf{X}))\nabla A(g,\mathbf{X})(\mathbf{x})]\Delta g(\mathbf{x})p_X(\mathbf{x})d\mathbf{x}$$

- This implies that

$$\nabla\mathcal{C}(g)(\mathbf{x}) = \mathbb{E}_{X,Y}[\partial_2 L(\mathbf{Y}, A(g,\mathbf{X}))\nabla A(g,\mathbf{X})(\mathbf{x})]$$

Comparison

- Notice the difference of gradients:

$$\nabla\mathcal{L}(F)(\mathbf{x}) = \mathbb{E}_{Y|X}[\partial_2 L(\mathbf{Y}, F(\mathbf{x})) \mid \mathbf{X} = \mathbf{x}]$$
$$\nabla\mathcal{C}(g)(\mathbf{x}) = \mathbb{E}_{X,Y}[\partial_2 L(\mathbf{Y}, A(g, \mathbf{X}))\nabla A(g, \mathbf{X})(\mathbf{x})]$$

- The usual gradient boosting can be retrieved by assuming $A(g, \mathbf{x}) = g(\mathbf{x})$

- In this case, $\nabla A(g, \mathbf{X})(\mathbf{x}) = 1_{\{\mathbf{X}=\mathbf{x}\}}$, which implies:

$$\begin{aligned}\nabla\mathcal{C}(g)(\mathbf{x}) &= \mathbb{E}_{X,Y}[\partial_2 L(\mathbf{Y}, g(\mathbf{X}))1_{\{\mathbf{X}=\mathbf{x}\}}] \\ &\propto \mathbb{E}_{Y|X}[\partial_2 L(\mathbf{Y}, g(\mathbf{x})) \mid \mathbf{X} = \mathbf{x}] = \nabla\mathcal{L}(g)(\mathbf{x})\end{aligned}$$

Algorithm

- As before, $\nabla\mathcal{C}(g)$, viewed as a function on $\mathbb{X}$, might not belong to $\mathcal{S}$
- Therefore, we should choose $\Delta g^* \in \mathcal{S}$ as similar as possible to $\nabla\mathcal{C}(g)$, i.e. choose $\Delta g^* \in \mathcal{S}$ that maximizes

$$-\langle \nabla\mathcal{C}(g), \Delta g\rangle = -D\mathcal{C}(g)(\Delta g) = -\int_{\mathbb{X}} \nabla\mathcal{C}(g)(\mathbf{x})\Delta g(\mathbf{x}) p_X(\mathbf{x}) d\mathbf{x}$$

- We then improve g by considering $g + \nu\Delta g^*$, where ν is a learning rate
- The Gradient Boosting for inverse algorithm is the iteration of the steps above starting from a given $g_0 \in \mathcal{S}$ and stopping when $-\langle \nabla\mathcal{C}(g_m), \Delta g\rangle \leq 0$ for all $g \in \mathcal{S}$

Convergence

Let $(g_m)_{m\in\mathbb{N}}$ be a sequence generated by the Gradient boosting algorithm

Theorem

If $\mathcal{C}$ is convex, lower bounded and Lipschitz differentiable, $\mathcal{S}$ is symmetric the algorithm always finds the function $\Delta g_m \in \mathcal{S}$ that maximizes $-\langle \nabla\mathcal{C}(g_{m-1}), \Delta g\rangle$, then

$$\lim_{m\to+\infty} \sup_{\Delta g\in\mathcal{S}} -\langle \nabla\mathcal{C}(g_m), \Delta g\rangle = 0 \textit{ and } \mathcal{C}(h) = \inf_{g\in Lin(\mathcal{S})} \mathcal{C}(g),$$

for any accumulation point h of $(g_m)_{m\in\mathbb{N}}$

Finite Data

Given a finite sample $\mathcal{S} = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_N, \mathbf{y}_N)\}$ from $(\mathbf{X}, \mathbf{Y})$, we use the following approximation

$$\mathcal{C}(g) \approx \frac{1}{N} \sum_{j=1}^{n} L(\mathbf{y}_j, A(g, \mathbf{x}_j)),$$

$$\nabla \mathcal{C}(g)(\mathbf{x}_i) \approx \frac{1}{N} \sum_{j=1}^{n} \partial_2 L(\mathbf{y}_j, A(g, \mathbf{x}_j)) \nabla A(g, \mathbf{x}_j)(\mathbf{x}_i),$$

$$\langle \nabla \mathcal{C}(g), \Delta g \rangle \approx \frac{1}{N^2} \sum_{i,j=1}^{n} \partial_2 L(\mathbf{y}_j, A(g, \mathbf{x}_j)) \nabla A(g, \mathbf{x}_j)(\mathbf{x}_i) \Delta g(\mathbf{x}_i)$$

Moreover, the gradient $\nabla A(g, \mathbf{x}_j)(\mathbf{x}_i)$ could be computed using the limit:

$$\nabla A(g, \mathbf{x}_j)(\mathbf{x}_i) = \frac{1}{p_X(\mathbf{x}_i)} \lim_{\varepsilon \to 0} \frac{A(g + \varepsilon \delta_{\mathbf{x}_i}, \mathbf{x}_j) - A(g, \mathbf{x}_j)}{\varepsilon}$$

One could improve the estimation by considering

- $\rho_m = \arg\min_{\rho} \frac{1}{N} \sum_{j=1}^{N} L(\mathbf{y}_j, A(g_{m-1} + \rho \Delta g_m^*, \mathbf{x}_j))$
- we then update $g_m = g_{m-1} + \nu \rho_m \Delta g_m^*$

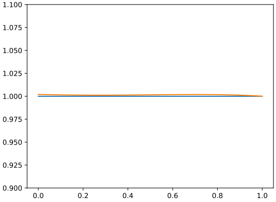
$A(g, \cdot)$ is the solution u of the PDE

$$\begin{cases} -(gu_x)_x = f, \text{ for } x \in (0,1), \\ u(0) = a_0 \text{ and } u(1) = a_1, \end{cases}$$

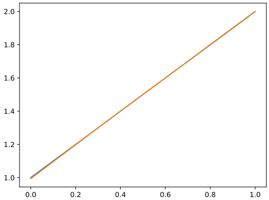
where f, a_0, a_1 are fixed and known. We consider the cases:

- $g(x) = 1 + \phi(x)$, with $\phi(x) = 1, x, x^2, \sin(2\pi x)$
- $u(x) = \sin^2(2\pi x)$ (and f accordingly), $a_0 = a_1 = 0$
- Compute $A(g, \cdot)$ using finite difference in a grid x with $\Delta x = 0.01$
- Learning rate $\nu = 0.1$, boosts were a neural network with 1 hidden layer with 5 neurons and activation `tanh`
- Noisy data: $u^\delta(x) = u(x) + \delta N(0, 1)$

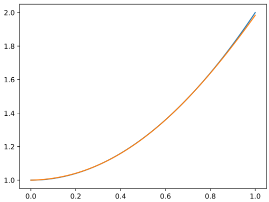
Example - PDE



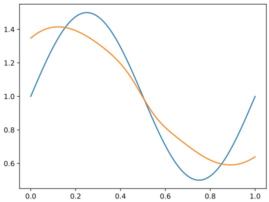
$g \equiv 1$



$g(x) = 1 + x$

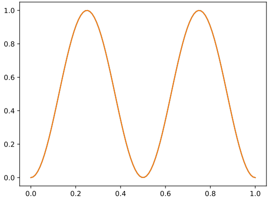


$g(x) = 1 + x^2$

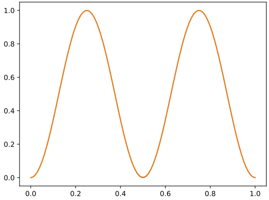


$g(x) = 1 + \sin(2\pi x)$

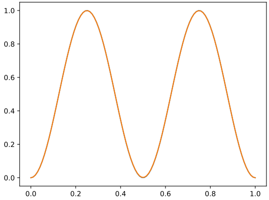
Example - PDE



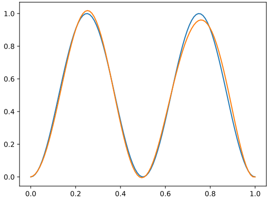
$g \equiv 1$



$g(x) = 1 + x$

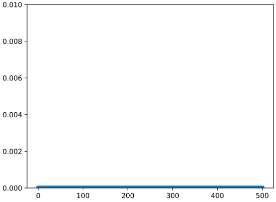


$g(x) = 1 + x^2$

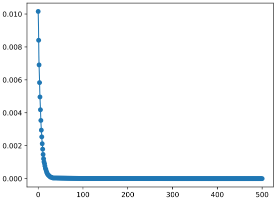


$g(x) = 1 + \sin(2\pi x)$

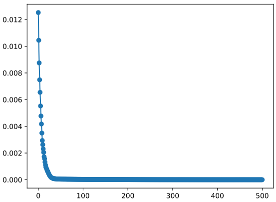
Example - PDE



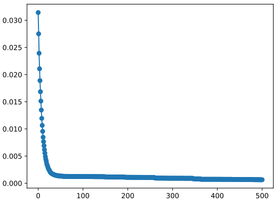
$g \equiv 1$



$g(x) = 1 + x$

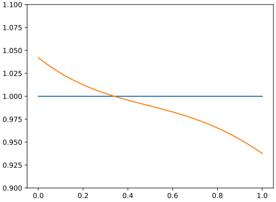


$g(x) = 1 + x^2$

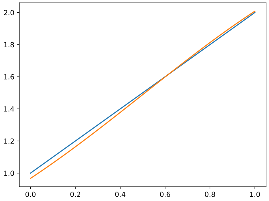


$g(x) = 1 + \sin(2\pi x)$

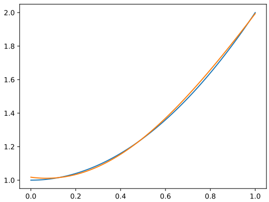
Example - PDE - Noisy Example



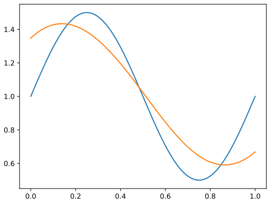
$g \equiv 1$



$g(x) = 1 + x$

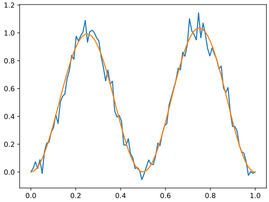


$g(x) = 1 + x^2$

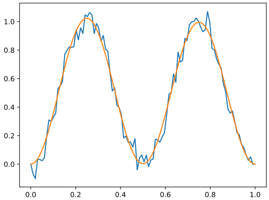


$g(x) = 1 + \sin(2\pi x)$

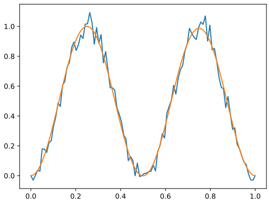
Example - PDE - Noisy Example



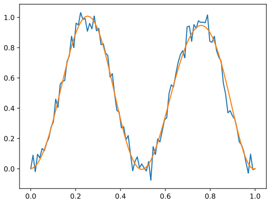
$g \equiv 1$



$g(x) = 1 + x$

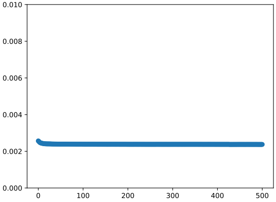


$g(x) = 1 + x^2$

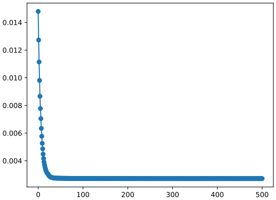


$g(x) = 1 + \sin(2\pi x)$

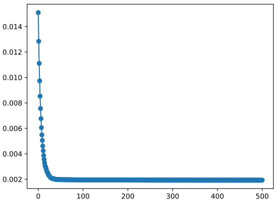
Example - PDE - Noisy Example



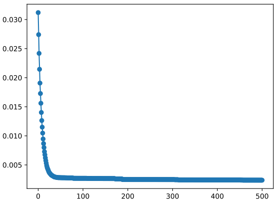
$g \equiv 1$



$g(x) = 1 + x$



$g(x) = 1 + x^2$

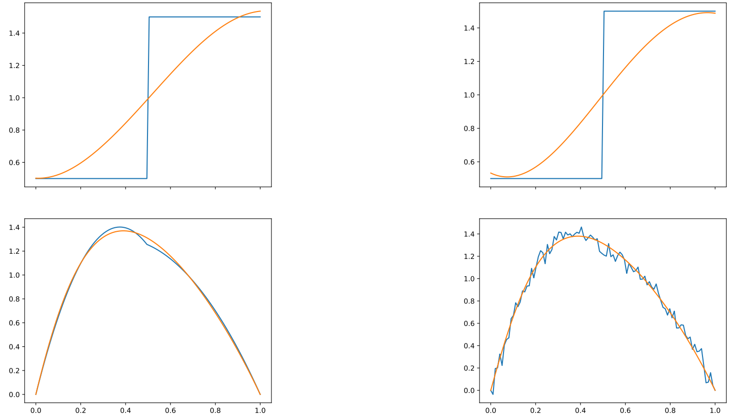


$g(x) = 1 + \sin(2\pi x)$

Example - PDE - Discontinuous g

- $g(x) = \begin{cases} 0.5, & \text{if } x < 0.5, \\ 1.5, & \text{if } x \geq 0.5 \end{cases}$
- $f \equiv 10$, $a_0 = a_1 = 0$
- u computed with finite difference
- Compute $A(g, \cdot)$ using finite difference in a grid x with $\Delta x = 0.01$
- Learning rate $\nu = 0.1$, boosts were a neural network with 1 hidden layer with 5 neurons and activation `tanh`
- Noisy data: $u^{\delta}(x) = u(x) + \delta N(0, 1)$

Example - PDE - Discontinuous case



Future Research

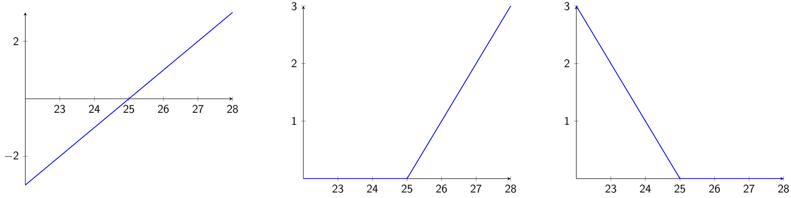
- Improve conditions in the gradient boosting theorem for inverse problem: convexity and Lipschitz differentiability of $\mathcal{C}$ might be too strong.
- Add restrictions to the estimator of g: we often know some information about g. e.g. it should be positive, etc.
- Apply the method to more complex problems, e.g. local volatility in Finance

Inverse Problems in Finance

Derivative Contracts

- A company wants to ensure its gasoline supply for next month
- How much should be paid today to guarantee the delivery of the agreed amount of gasoline in one month?
- This contract is a derivative:
  - the date the product will be delivered is the *maturity*
  - the agreed price for the product is the *strike*
  - the price to enter this contract is the *premium*
- There are some well-known types of derivatives:
  - Future: contract to buy or sell an asset at the maturity at the strike price
  - Options: contract that gives the holder the right, but not the obligation, to buy or sell an asset at the maturity at the strike price

- There are two important types of options: *calls* and *puts*
- Calls give the holder the option to buy; Puts, the option to sell.
- The payoff of an option is the amount at maturity is called *payoff*
- S_T is the stock price at maturity and K, the strike. The payoff of
  - a future is $S_T - K$
  - a call option is $S_T - K$, if $S_T > K$, and 0 otherwise
  - a put option is $K - S_T$, if $S_T < K$, and 0 otherwise



Future $K - S_T$

Call $(S_T - K)^+$

Put $(K - S_T)^+$

- An arbitrage opportunity is a strategy that costs nothing today, it cannot lose money and it has a chance of profit in the future
- Mathematically, the value of this strategy, V_t, satisfies $V_0 = 0$, $\mathbb{P}(V_t \geq 0) = 1$ and $\mathbb{P}(V_t > 0) > 0$
- Notice that $(S_T - K)^+ - (K - S_T)^+ = (S_T - K)$, i.e. Call minus Put equal Future at T
- In a market without arbitrage opportunities, we would expect that this relation is true at any time $t \leq T$. Indeed, this is the Put-Call Parity:

$$C_t(T, K) - P_t(T, K) = F_t(T, K)$$

Black–Scholes Model

- Black–Scholes dynamics (under the historical measure):

$$dS_t = \mu S_t dt + \sigma S_t dW_t$$

- The parameter μ is the rate of return
  - This parameter is quite difficult to estimate/calibrate
  - But it is not used in the pricing formula
- The most important parameter in this model is the volatility, σ, assumed constant
- Under the risk-neutral measure, the dynamics is given by:

$$dS_t = rS_t dt + \sigma S_t dW_t,$$

where r is the risk-free interest rate

- Using Itô's formula with $f(x) = \log x$, one might show that

$$S_t = S_0 e^{\left(r - \frac{\sigma^2}{2}\right)t + \sigma W_t}$$

- So, S_t is log-normally distributed
- Using this, there exists a closed-formula for the price of a call with maturity T and strike K:

$$C(t, S, T, K, \sigma) = SN(d_1) - Ke^{-r(T-t)}N(d_2),$$

where

$$d_1 = \frac{\log(S/K) + (r + \sigma^2/2)(T - t)}{\sigma\sqrt{T - t}},$$
$$d_2 = d_1 - \sigma\sqrt{T - t}$$

- The implied volatility of the market price, C^M, of a call with maturity T and strike K is implicitly defined by

$$C_{BS}(t, S, T, K, \hat{\sigma}_t(T, K)) = C^M(T, K)$$

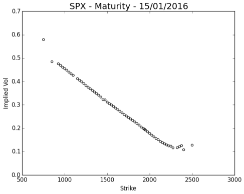
- The implied vol exists (given that $(S - e^{-r(T-t)}K)^+ \leq C^M \leq S$) and it is unique (since the Vega $\frac{\partial C_{BS}}{\partial \sigma}$ is strictly positive)
- The implied volatility is the language used in the market to quote option prices
- "implied volatility is the wrong number to put in the wrong formula to obtain the right price." (Rebonato)

Implied Volatility - The Smile

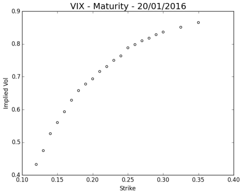
- If the Black–Scholes model were the correct market model, then we would expect that

$$\hat{\sigma}_t(T, K_1) = \hat{\sigma}_t(T, K_2)$$

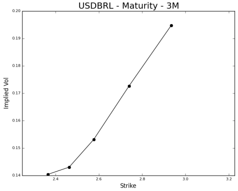
- Until the crash of 87 (Black Monday), this was actually verified in the American Equities market
- Since then, the *smile* is found in all liquid markets



S&P 500



VIX



USDBRL

- More complex models are needed!

$$dS_t = rS_t dt + \sigma S_t dW_t^S$$

  - Time dependent volatility: $\sigma(t)$
  - Local volatility: $\sigma(t, S)$
  - Stochastic Volatility (e.g. Heston Model): $\sqrt{V_t}$
  - Stochastic Local Volatility: $L(t, S)\sqrt{V_t}$

Stylized Facts about Asset and Volatility

- Volatility clustering - small moves in prices follow small moves and large moves follow large moves
- Heavy tails - distribution of asset returns have heavier tails that the normal distribution (as the Black–Scholes model assumes)
- Leverage effect - negative correlation between asset price and volatility (for some markets, mainly equities and index); when price goes down, vol rises
- Mean reversion - when volatility rises or falls, it tends to revert to some long-run level

Black–Scholes - First Generalization

- In the case where $\hat{\sigma}_t(T, K_1) = \hat{\sigma}_t(T, K_2)$, but $\hat{\sigma}_t(T_1, K) \neq \hat{\sigma}_t(T_2, K)$, the Black–Scholes model can be easily generalized to fit this behavior
- Consider:

$$dS_t = r(t)S_t dt + \sigma(t)S_t dW_t$$

- There is still a closed form solution for the price of a call option with strike K and maturity T:

$$C_{BS}(t, S, T, K, \sigma) = SN(\bar{d}_1) - Ke^{-\bar{r}(T-t)}N(\bar{d}_2),$$

where

$$\bar{d}_{1,2} = \frac{\log(S/K) + (\bar{r} \pm \bar{\sigma}^2/2)(T-t)}{\bar{\sigma}\sqrt{T-t}},$$

$$\bar{r} = \frac{1}{T-t}\int_t^T r(s)ds, \quad \bar{\sigma} = \sqrt{\frac{1}{T-t}\int_t^T \sigma^2(s)ds}$$

Local Volatility

- Let $\{C^M(T,K)\ ;\ \ T>0,\ \ K>0\}$ the set of all call option prices available in the market
- In what follows, we will assume there are a continuum of call option prices available
- The local volatility dynamics assumes

$$dS_t = rS_t dt + \sigma_L(t, S_t) S_t dW_t$$

- The local volatility for the prices C^M is the function σ_L such that the prices of call options under the model above is equal to prices C^M observed in the market, i.e. is the function that reprices the market
- In symbols:

$$C(T,K) = e^{-r_d T}\mathbb{E}[(S_T - K)^+] = C^M(T,K)\ , \quad \forall\ T, K > 0$$

FGV EMAp ESCOLA DE MATEMÁTICA APLICADA

Theorem

In the famous paper "Pricing with a smile", Bruno Dupire proved that the local volatility exists, it is unique and it is given by the formula:

$$\frac{\partial C}{\partial T}^M(T,K) + rK\frac{\partial C}{\partial K}^M(T,K) - \frac{1}{2}\sigma_L^2(T,K)K^2\frac{\partial^2 C}{\partial K^2}^M(T,K) = 0$$

or

$$\sigma_L^2(T,K) = \frac{\frac{\partial C}{\partial T}^M(T,K) + rK\frac{\partial C}{\partial K}^M(T,K)}{\frac{1}{2}K^2\frac{\partial^2 C}{\partial K^2}^M(T,K)}$$

- Many ways to prove the previous theorem.
- One of them is to use the Fokker-Planck equation and the *Breeden-Litzenberger formula*:

$$\frac{\partial^2 C}{\partial K^2}(t, S, T, K) = e^{-r(T-t)} p(t, S, T, K),$$

  where $p(t, S, T, \cdot)$ is the density of S_T given that $S_t = S$
- This follows from

$$C(t, S, T, K) = e^{-r(T-t)} \int_0^{+\infty} p(t, S, T, x)(x - K)^+ dx$$

- As Dupire wrote: "[the Local vol formula] holds only because the intrinsic value of a call happens to be the second integral of a Dirac function. It is very fortunate that the market trades this particular payoff!"

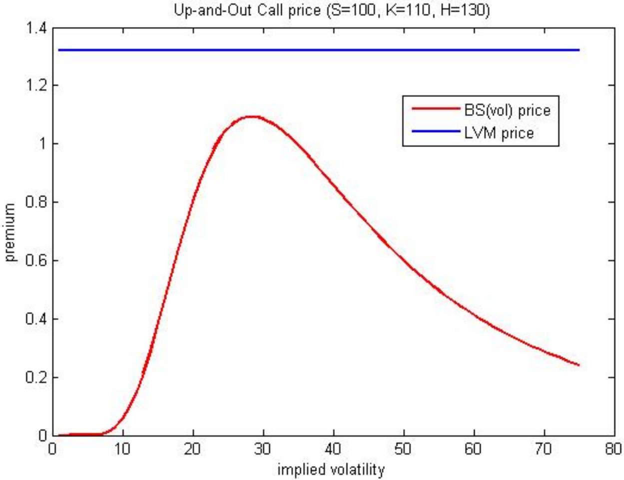
Local Volatility in Practice

- In practice, there are just some finite number of implied vols
- However, the formulas we have seen for the Local Volatility requires a continuum of implied vols
- This means we may need to interpolate the implied vols from the market
- Which must be done in a way that does not introduce arbitrage in the prices

Pros and Cons

- Pros:
  - in theory, perfectly calibrates the market
  - keeps the market complete (only one dW to be hedged)
  - it is a simple model to price exotic derivatives (barrier, forward-start, etc) - just use PDE methods or Monte Carlo simulation
- Cons:
  - requires the interpolation of the implied volatility surface
  - volatility is perfectly correlated with the spot
  - volatility is not mean-reverting
  - computation requires numerical differentiation (usually unstable)
  - it does not capture the correct price of exotic derivatives that depends on the realized volatility (the correlation becomes important)

Exotic Option - Barrier



Price of Barrier Option in Black-Scholes (various vols) and in Local Vol - taken from one of Dupire's talks

Exotic Option - Vol KO

- It is a call option with European barrier in the realized volatility during the life of the option
- The payoff can be written as $g(S, RV) = (S - K)^{+} 1_{\{RV \leq H\}}$
- The realized vol is usually (meaning contractually) given by

$$RV = \sqrt{\frac{AF}{N} \sum_{i=1}^{N} \left(\log \left(\frac{S_{t_i}}{S_{t_{i-1}}} \right) \right)^2},$$

onde $t_i = (i/N)T$

- The Stochastic Local Volatility (SLV) model is a hybrid model that captures important characteristics of both local and stochastic volatility models
- It can be mathematically described by the SDEs

$$\begin{cases} dS_t = rS_t dt + L(t, S_t)\sqrt{V_t} dW_t^S \\ \\ dV_t = \alpha(t, V_t)dt + \beta(t, V_t) dW_t^V \end{cases}$$

- The most important restriction is:

$$\sigma_L^2(t, S) = \mathbb{E}[L^2(t, S_t)V_t \mid S_t = S] \text{ or } L^2(t, S) = \frac{\sigma_L^2(t, S)}{\mathbb{E}[V_t \mid S_t = S]}$$

- Since the joint dynamics of (S, V) depends on L, the equation above is an implicit equation for L
- It is fairly difficult to implement this model

Thank you!