CSE 446
Multi-armed bandits

Leo Maynard-Zhang

UNIVERSITY of WASHINGTON

Decision-making in the face of uncertainty

- In life we are faced with many difficult decisions
- Often these decisions have uncertain outcomes
  - “How much time should I spend studying for my exam?”
  - “Which stock should I invest in?”
  - “Should I pay the parking fee?”
- We have had to make such decisions likely more than once in our lifetime
- How do we find a way to make the optimal choice when we aren’t certain about the outcome?
- ML approach: What if we used the outcomes of our past decisions to help inform our future ones?

Slot machines

- Your friend who owns a casino says you can come play his 10 slot machines for free, but you only get 100 pulls
- Each slot machine has a different unknown expected payoff
  - Payoffs follow some distribution, for example, Gaussian
- Assume one of these slot machines has a higher expected payoff than all the others
- How will you maximize your winnings?

Slot machines

- One approach:
  - Play each of the 10 slot machines once
  - Then commit to the slot machine with the highest return, and play it 90 times
- What is the issue with this approach?
  - Very volatile!
  - The slot machines are random, and the one that initially gives the highest return might not actually be the best one
  - Worst case: Play suboptimal slot machines 91 + 8 = 99 times!!!

Slot machines

- Second approach:
  - Play each of the 10 slot machines 9 times
  - Then commit to the slot machine with the highest average return, and play it 10 times
- What is the issue with this approach?
  - Too safe!
  - Best case: Play best slot machine only 10 + 9 = 19 times!!!

Exploration vs Exploitation

- We must balance the following tradeoff:
  - Exploration: We want to find the slot machine that is optimal
    - This means we have to explore every slot machine, including those that are suboptimal
  - Exploitation: We want to play the slot machine that we believe to be optimal
    - This means we have to concentrate on the slot machines that have already given favorable returns
- These objectives are clearly conflicting

The bandit problem

- There are k possible actions enumerated as $\{1,2,\ldots,k\}$, with a distribution for each action dictating rewards: $(\mu_1, \ldots, \mu_k)$
  - k slot machines
  - μ_i denotes the expected return of the ith slot machine
- We are allowed to do n actions sequentially
  - n total pulls
- How do we make the right decisions in such uncertain environments?

First algorithm: Explore-then-commit

- First choose hyperparameter m
- Then run the following algorithm:
  - Input $k, n, m \in \mathbb{N}$
  - For $i = 1, \ldots, k$:
    - Play action i exactly m times, observe m rewards $r_1, \ldots, r_m$ and compute empirical mean $\hat{\mu}_i = \frac{1}{m} \sum_{j=1}^{m} r_j$
  - For $t = mk + 1, \ldots, n$:
    - Play action $a_* = \arg\max_{i \in [k]} \hat{\mu}_i$

This is just the slot machine algorithm from earlier defined formally.

It turns out we can do much better than this, see UCB and Thompson Sampling.

Today's lecture

- We will be focusing on the setting where our action space is featurized in some finite dimensional space
- That is, instead of our action set being $\{1,2,\ldots,k\}$, it will be of the form $A \subset \mathbb{R}^d$
  - A can be infinite!
- The reward distributions will be dictated by some common linear model
- Why?
  - Think about driving a car. Is there a way to describe the set of available actions in a finite and one-dimensional way?

Linear bandits

Recap: Linear regression

- Given fixed dataset $\{x_i, y_i\}_{i=1}^n$, with $y_i = x_i^\top w + \epsilon_i$, where $w \in \mathbb{R}^d$ is unknown and ϵ_i is zero-mean Gaussian noise.
- Goal: estimate w to make predictions on unseen data
- Least squares estimator:

$$\widehat{w} = \arg\min_w \sum_{i=1}^n (x_i^\top w - y_i)^2$$

- Closed form:

$$\widehat{w} = \left(\sum_{i=1}^n x_i x_i^\top \right)^{-1} \sum_{i=1}^n x_i y_i$$

Linear bandit setting

- Input action set $A \subset \mathbb{R}^d$, $n \in \mathbb{N}$, and unknown $w_* \in \mathbb{R}^d$
- For $t = 1, \ldots, n$:
  - Decision-maker chooses $x_t \in A$ based on $\{x_i, r_i\}_{i=1}^{t-1}$
  - Observe reward $r_t = x_t^\top w_* + \epsilon_t$

Reward Maximization

- The standard objective in sequential decision-making is to maximize reward. i.e. maximize $\sum_{t=1}^{n} r_t$
- We have to be careful about how we choose actions at each time step—every action we make contributes directly to our objective
  - Have to consider the "cost" of our actions
  - Cannot just explore freely
- Example: Advertising
  - Each ad we place generates some revenue
  - Want to maximize our total revenue

Exploration vs Exploitation

- We must balance the following tradeoff:
  - Exploration: We want to find the action that is optimal
    - This means we have to explore every action, including those that are suboptimal
  - Exploitation: We want to play the action that we believe to be optimal
    - This means we have to concentrate on actions that have already given favorable returns
- These objectives are clearly conflicting
- How to measure how well an algorithm balances this tradeoff?

Regret

- Let π be a bandit algorithm, i.e. a decision rule that at time step t maps history $\{x_i, r_i\}_{i=1}^{t-1}$ to action $x_t \in A$
  - We allow π to be deterministic or stochastic
- We define the (pseudo) regret of algorithm π on problem instance w_* with interaction length n as

$$R_n(\pi, w_*) = \sum_{t=1}^{n} \max_{x \in A} x^\top w_* - x_t^\top w_*$$

We will use R_n for shorthand

Max possible reward

Actual reward obtained by π

Regret

$$R_n = \sum_{t=1}^{n} \max_{x \in A} x^\top w_* - x_t^\top w_*$$

- What is a good value of regret?
- We want our regret to satisfy $R_n = o(n)$
  - $R_n = o(n) \iff \lim_{n \to \infty} \frac{R_n}{n} = 0$
- Average regret goes to 0 in the limit
  - Our algorithm eventually only plays optimal actions
- Examples: $O\left(\sqrt{n}\right), O\left(\log n\right), O\left(\sqrt{n}\log n\right)$
  - “sublinear”
  - $n \neq o(n)$

Linear bandit setting

- Input action set $A \subset \mathbb{R}^d$, $n \in \mathbb{N}$, and unknown $w_* \in \mathbb{R}^d$
- For $t = 1, \ldots, n$:
  - Decision-maker chooses $x_t \in A$ based on $\{x_i, r_i\}_{i=1}^{t-1}$
  - Observe reward $r_t = x_t^\top w_* + \epsilon_t$

Goal: Minimize regret $R_n = \sum_{t=1}^{n} \max_{x \in A} x^\top w_* - x_t^\top w_*$

Example: Spotify

- Suppose we are choosing songs to autoplay for a Spotify user. After the user is done listening to a song we choose a song to play next. Then we measure how long the user listened to the song and whether or not they saved it
  - n: number of songs the user plans to listen to
  - d: BPM, genre, popularity, release date, etc.
  - A: different songs we can recommend
  - r_t: percentage of song listened to + whether or not the user saved it
- Goal: recommend songs that the user enjoys listening to, so they continue to use our app

Optimism

- How do we tackle the exploration-exploitation problem?
- “Optimism in the face of uncertainty”
- Suppose you visit the same restaurant each week for Sunday brunch
  - If you only order the same dishes every week, you may never discover some dishes that you might love!
  - Solution: have an “optimistic” view of each menu item
    - Assume you’ll really like menu items you haven’t tried
    - This will lead to you trying every dish until you have sufficient reason to believe you don’t like them
- Key idea: Overestimating the true value of each action leads to efficient exploring

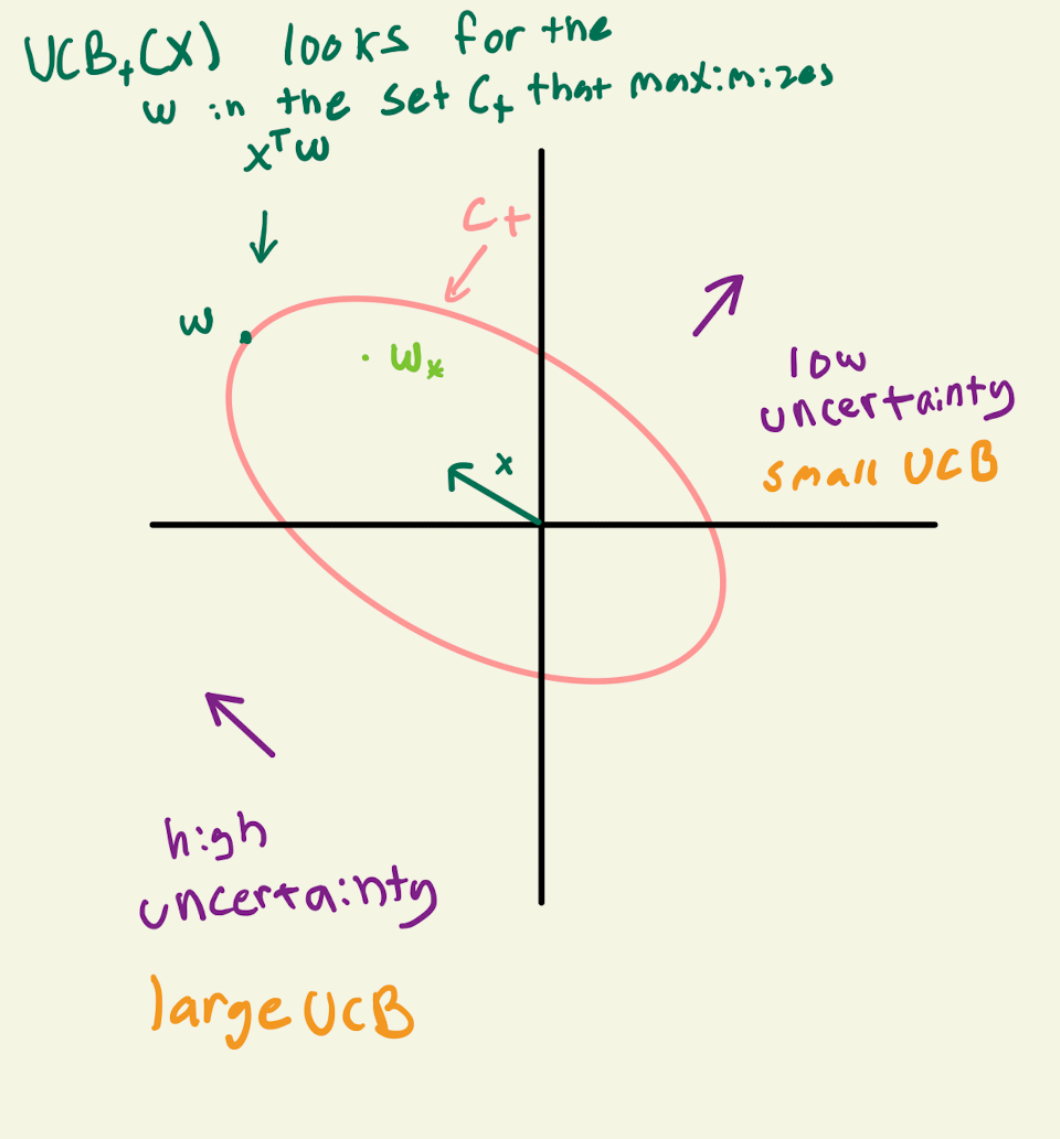
Optimism: Upper Confidence Bound

- Idea: at each time step t construct confidence set C_t that contains w_* with high probability
- For each $x \in A$, define

$$\text{UCB}_t(x) = \max_{w \in C_t} x^\top w$$

- Notice if $w_* \in C_t$ then $\text{UCB}_t(x) \geq x^\top w_*$ for all $x \in A$
  - $\text{UCB}_t(x)$ is an overestimate of the true value of each $x \in A$!

Optimism: Upper Confidence Bound



LinUCB Algorithm

- Input action set $A \subset \mathbb{R}^d$, $n \in \mathbb{N}$
- For $t = 1, \ldots, n$:
  - Construct C_t based on $\{x_i, r_i\}_{i=1}^{t-1}$
  - Denote $\text{UCB}_t(x) = \max_{w \in C_t} x^\top w$
  - Choose $x_t = \arg\max_{x \in A} \text{UCB}_t(x)$
  - Observe r_t

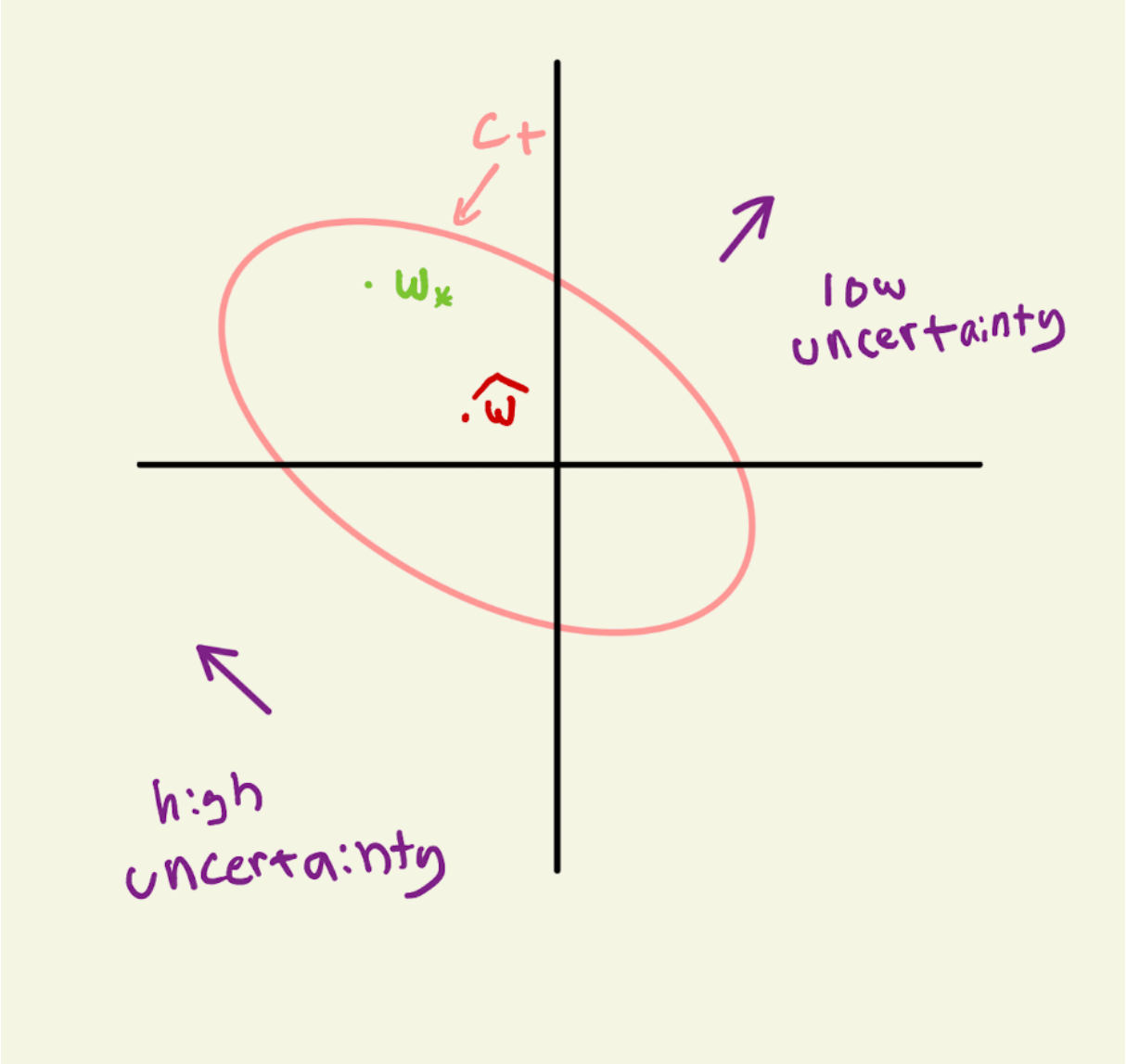
Confidence set

- Denote: $V_t = \sum_{i=1}^{t} x_i x_i^\top + I$
- Obtain ridge estimator $\widehat{w}_t = V_t^{-1} \sum_{i=1}^{t} x_i r_i$
- Choose confidence $\delta > 0$ and set $\gamma_t = O\left(\log\left(\frac{1}{\delta}\right) + d \log n\right)$
- Construct

$$C_t = \left\{ w \in \mathbb{R}^d : \left(\widehat{w}_{t-1} - w\right)^\top V_{t-1} \left(\widehat{w}_{t-1} - w\right) \leq \gamma_t \right\}$$

- Then with probability $\geq 1 - \delta$, $w_* \in C_t$ for all $t \in [n]$ (Theorem 2 Abbasi-Yadkori, 2011)

Confidence set



LinUCB regret

- (Theorem 19.2, Lattimore & Szepesvári 2020) With probability at least $1 - 1/n$, the worst-case regret of LinUCB satisfies:

$$R_n \leq O\left(d\sqrt{n}\log n\right)$$

Near-optimality of LinUCB

- (Theorem 24.2, Lattimore & Szepesvári 2020) Suppose our action set A is the unit ball. Then for any algorithm π, there exists a problem instance $w_* \in \mathbb{R}^d$ such that:

$$R_n(\pi, w_*) \geq \Omega\left(d\sqrt{n}\right)$$

LinUCB is (in general) optimal up to logarithmic terms!

Experimental design

Linear bandits with finitely many arms

- When the arm set is finite, i.e. $\left|A\right| = k$ we can improve further
- Idea: choose initial sequence of arms such that the variance of the least squares estimator is uniformly minimized across all arms
  - That is, choose $x_1, \ldots, x_t \in A$ to minimize:

$$\max_{x \in A} \mathsf{Var}\left(x^\top \widehat{w}_t - x^\top w_*\right)$$

- This will lead to stronger estimation of the value of each arm

Optimal design for least squares

- For fixed sequence $\mathbf{x}_t = x_1, \ldots, x_t \in A$, let:

$$V_{\mathbf{x}_t} = \sum_{i=1}^{t} x_i x_i^\top$$

- Least squares estimator:

$$\widehat{w}_t = V_{\mathbf{x}_t}^{-1} \sum_{i=1}^{t} x_t r_t$$

- Fact: for any $x \in \mathbb{R}^d$:

Prove at home

$$\mathrm{Var}\left(x^\top \widehat{w}_t - x^\top w_*\right) = x^\top V_{\mathbf{x}_t}^{-1} x$$

- So we aim choose sequence of arms $\mathbf{x}_t$ such that:

$$\mathbf{x}_t = \arg\min_{\mathbf{x}_t} \max_{x \in A} x^\top V_{\mathbf{x}_t}^{-1} x$$

  - Intractable
    - Discrete optimization over exponentially many options

Optimal design for least squares

- For sequence $\mathbf{x}_t$ define $T_{\mathbf{x}_t} : A \to \mathbb{N}$ as:

$$T_{\mathbf{x}_t}(x) = \sum_{i=1}^{t} \mathbf{1}\{x_i = x\}$$

  - $T_{\mathbf{x}_t}(x)$ returns the number of times x appears in $\mathbf{x}_t$

- Fact:

$$\sum_{x \in A} T_{\mathbf{x}_t}(x) = t$$

- Fact:

$$V_{\mathbf{x}_t} = \sum_{i=1}^{t} x_i x_i^\top = \sum_{x \in A} T_{\mathbf{x}_t}(x) x x^\top$$

Optimal design for least squares

- Note:

$$\sum_{x \in A} T_{\mathbf{x}_t}(x) = t \Rightarrow \sum_{x \in A} \frac{T_{\mathbf{x}_t}(x)}{t} = 1$$

- So $\lambda(x) := \dfrac{T_{\mathbf{x}_t}(x)}{t}$ is a PMF on A
- Notice:

$$\begin{aligned} V_{\mathbf{x}_t} &= \sum_{x \in A} T_{\mathbf{x}_t}(x) x x^\top \\ &= t \sum_{x \in A} \frac{T_{\mathbf{x}_t}(x)}{t} x x^\top \\ &= t \sum_{x \in A} \lambda(x) x x^\top \end{aligned}$$

The G-optimal design

- Idea: minimize over “smooth” PMFs instead of discrete allocations
- Call $\triangle_A$ the set of PMFs on A
- For $\lambda \in \triangle_A$ denote:

$$V_\lambda = \sum_{x \in A} \lambda(x) x x^\top$$

- G-optimal design of A:

$$\lambda^* = \arg \min_{\lambda \in \triangle_A} \max_{x \in A} x^\top V_\lambda^{-1} x$$

  - Tractable
    - Gradient based methods
      - Franke-Wolfe algorithm

The G-optimal design

- To convert λ^* to a discrete sequence of actions, an option is to play each arm $x \in A$ exactly $\lceil t\lambda^*(x) \rceil$ times
- There exists optimizer λ^* such that:

$$\left| \text{Supp}(\lambda^*) \right| \leq d(d+1)/2$$

  - (Theorem 21.1, Lattimore & Szepesvári, 2020)
- The above implies:

$$\sum_{x \in A} \lceil t\lambda^*(x) \rceil \leq t + d(d+1)/2$$

- That is, converting λ^* to a discrete sequence of actions costs us $O\left(d^2\right)$ extra samples

Phased elimination with G-optimal

- Input action set $A \subset \mathbb{R}^d$, $n \in \mathbb{N}$
- $A_1 \leftarrow A$
- For $l = 1, \ldots$:
  - Compute λ_l^* as the G-optimal design of A_l
  - Play each arm $x \in A_l$ exactly $\lceil t_l \lambda_l^*(x) \rceil$ * times to obtain dataset D_l
  - Train least squares estimator $\widehat{w}_l$ on D_l
  - $\hat{x}_l = \arg\max_{x \in A_l} x^\top \widehat{w}_l$
  - $A_{l+1} \leftarrow \left\{ x \in A_l : (\hat{x}_l - x)^\top \widehat{w}_l \leq 2^{-l+1} \right\}$

* t_l is a carefully chosen number determining how many arms are pulled each phase

Phased elimination regret

- (Theorem 22.1, Lattimore & Szepesvári, 2020) With probability at least $1 - 1/n$, the worst-case regret of phased elimination satisfies:

$$R_n \leq O\left(\sqrt{dn \log\left(kn\right)}\right)$$

Contextual bandits

Contextual bandits

- The assumption that there is a singular “one-size fits all” best action is not always sound
- At each time step we might have access to some contextual information that we can incorporate to make better decisions
  - These contexts can be thought of as a “state”
  - Some actions might be better than others depending on which “state” you are in

Recommender systems

- Contextual bandits are widely used by recommendation systems: Amazon, TikTok, etc.
- Context is the information available on each consumer
  - For example, demographic information: age, gender, geographic region, etc.
- We want to recommend things that the user will want to consume, but each user might have different preferences correlated with their demographic
  - For example younger folks might want to watch video game clips, while older folks might want to watch cat videos.
  - Recommending the same content to both groups of people would not be wise!

Contextual bandit setting: linear approach

- Input action set A, context set C, feature mapping $\phi : C \times A \to \mathbb{R}^d$, $n \in \mathbb{N}$, and unknown $w_* \in \mathbb{R}^d$
- For $t = 1, \ldots, n$:
  - Nature reveals context $c_t \in C$
  - Decision-maker chooses $x_t \in A$ based on $\{x_i, r_i, c_i\}_{i=1}^{t-1}$ and c_t
  - Observe reward $r_t = \phi(c_t, x_t)^\top w_* + \epsilon_t$

Goal: Minimize regret $R_n = \sum_{t=1}^{n} \max_{x \in A} \phi(c_t, x)^\top w_* - \phi(c_t, x_t)^\top w_*$

Reduces to a linear bandit. LinUCB is (an) option

Best action is changing at each time step

Summary

- Online decision-making
  - Data is no longer fixed
  - Action-feedback loop
- Reward maximization
  - Regret minimization
- Exploration vs. exploitation
  - Why is reward maximization hard?
- Optimism
  - UCB
- Experimental design
  - Optimal data collection
- Contextual bandits
  - Incorporate contextual information to make more informed decisions

Further learning

- CSE 541: Interactive Learning; with Professor Kevin Jamieson
- Bandit Algorithms by Lattimore & Szepesvári