Posterior agreement and stochastic optimization

Statistical learning theory 2020

Maximum entropy and posterior agreement

Motivation

$$X = (X_1, \dots, X_n) \qquad X_i \sim \mathcal{N}(i, \sigma^2)$$

↓ draw

instances

$$X' = (X'_1, \dots, X'_n) \in \mathbb{R}^n$$

$$X'' = (X''_1, \dots, X''_n) \in \mathbb{R}^n$$

Goal: $\min_i \mathbb{E}[X_i]$

Formalization

$$\min_{c} \mathbb{E}_X [R(c, X)]$$

$$X = (X_1, \ldots, X_n)$$

$$C = \{1, \ldots, n\}$$

$$R(c, X) = X_c$$

Problem:

$$\min_{c \in C} \mathbb{E}_X [R(c, X)]$$

C : solution space

X : random instance

$R(c, X)$: cost of solution c on instance X.

Max entropy + PA

Problem: find $\min_c \mathbb{E}_X[R(c, X)]$, given X', X''.

Approach:

1. Compute $p_{T^*}(\cdot|X') \propto \exp\left(-\frac{1}{T^*} R(\cdot, X')\right)$

   $p_{T^*}(\cdot|X'') \propto \exp\left(-\frac{1}{T^*} R(\cdot, X'')\right)$

2. $\hat{c} \overset{\$}{\leftarrow} \frac{1}{Z} p_{T^*}(\cdot|X') \odot p_{T^*}(\cdot|X'')$

$$T^* = \arg\max_T K_T(X', X'')$$

* Why ME ?

* What's the right value of T?

Why ME?

- Algos should output distributions
- $p(\cdot|x)$ should be the "most general" distribution such that

$$* R(c_1, x) \leq R(c_2, x) \iff p(c_1|x) \geq p(c_2|x)$$

low cost $\iff$ high prob.

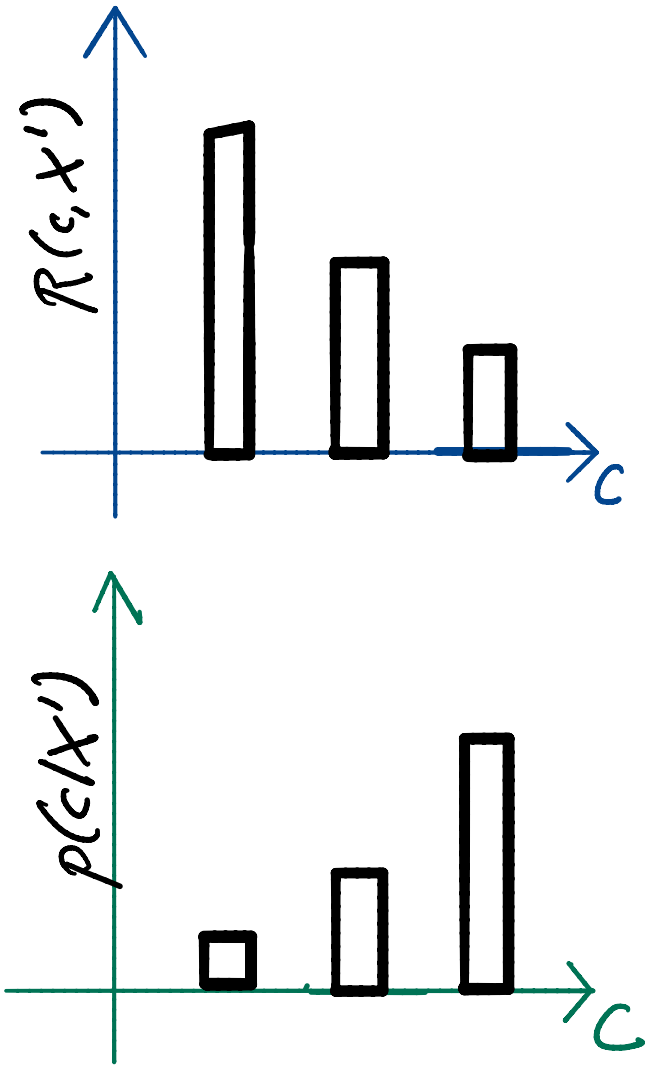
$$* \mathbb{E}_c[R(C, x)] = \text{const} \in [\min R(\cdot, x), \text{avg } R(\cdot, x)]$$

$$p(c|x') \propto \exp\left(-\frac{1}{T} R(c,x')\right)$$

low cost

$\Longleftrightarrow$

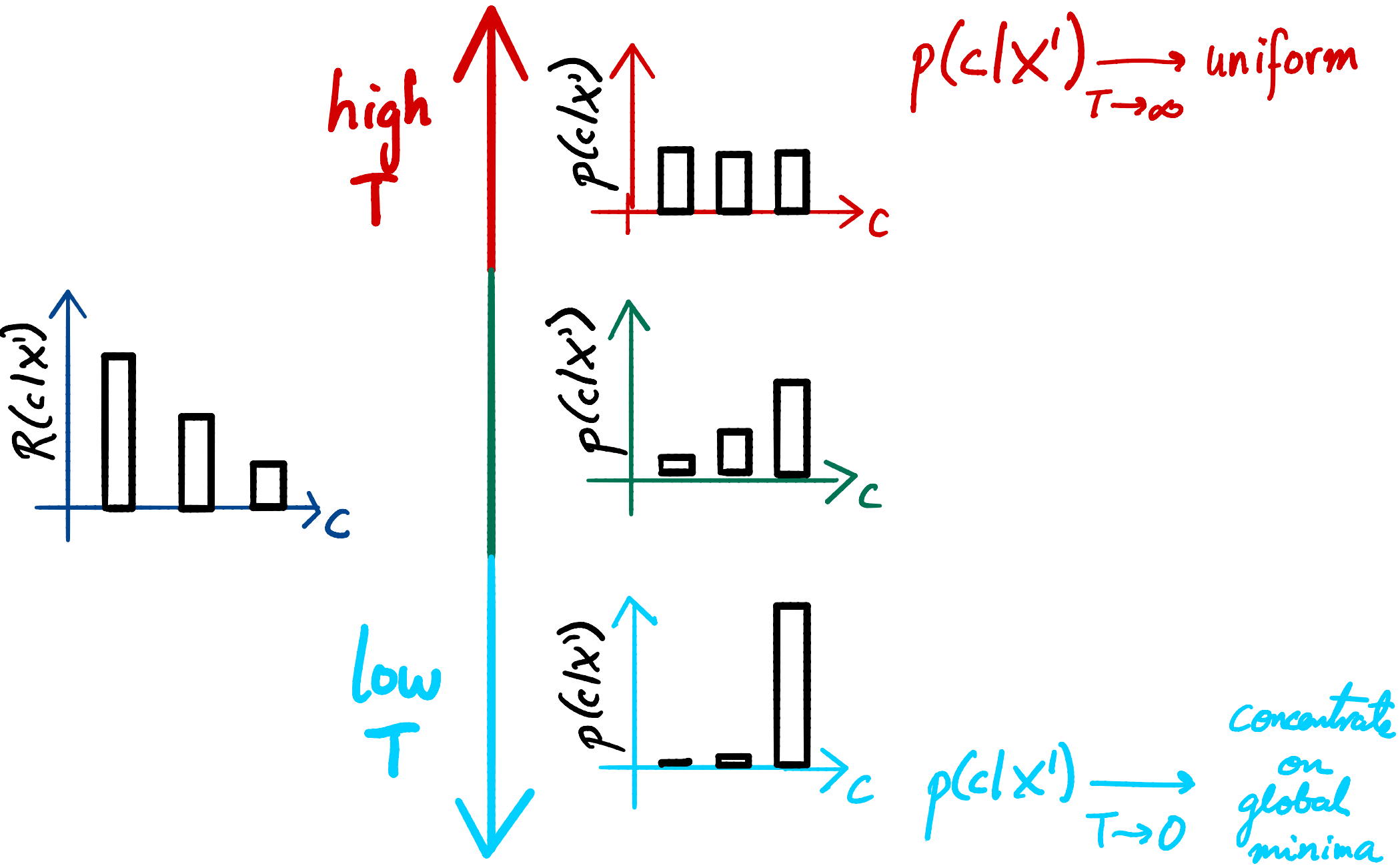
high prob.



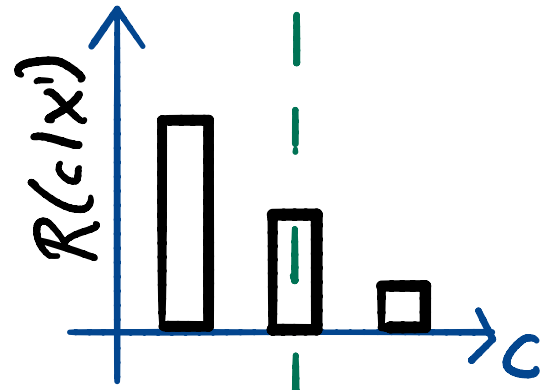
What is the right value for T?

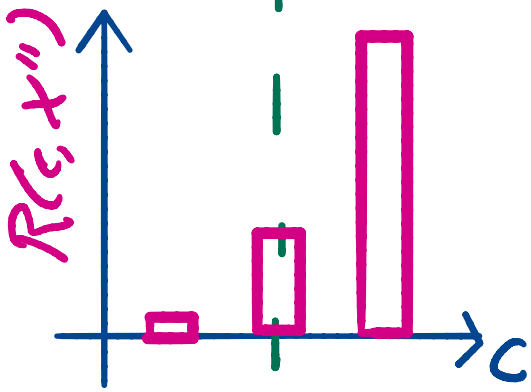
Temp	Max ent. as info extraction	Max ent. as info transmission
high	robust & poor fit ⇒ not info	low noise & few codewords ⇒ low cap
	robust & fit ⇒ info	low noise & many codewords ⇒ high cap
low	not robust & fit ⇒ not info	too noisy & many codewords ⇒ low cap

Max ent as info extraction



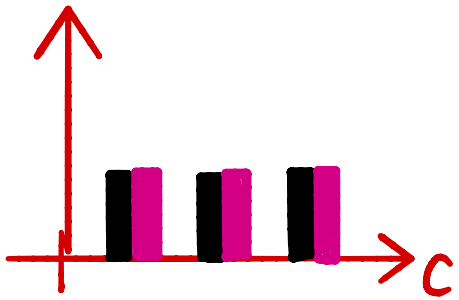
Max ent as info extraction

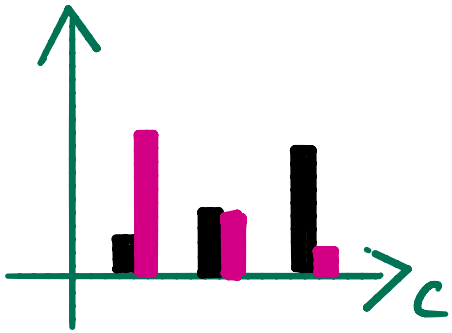




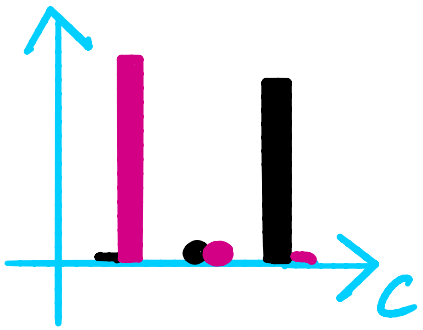
stable solution

high T





low T



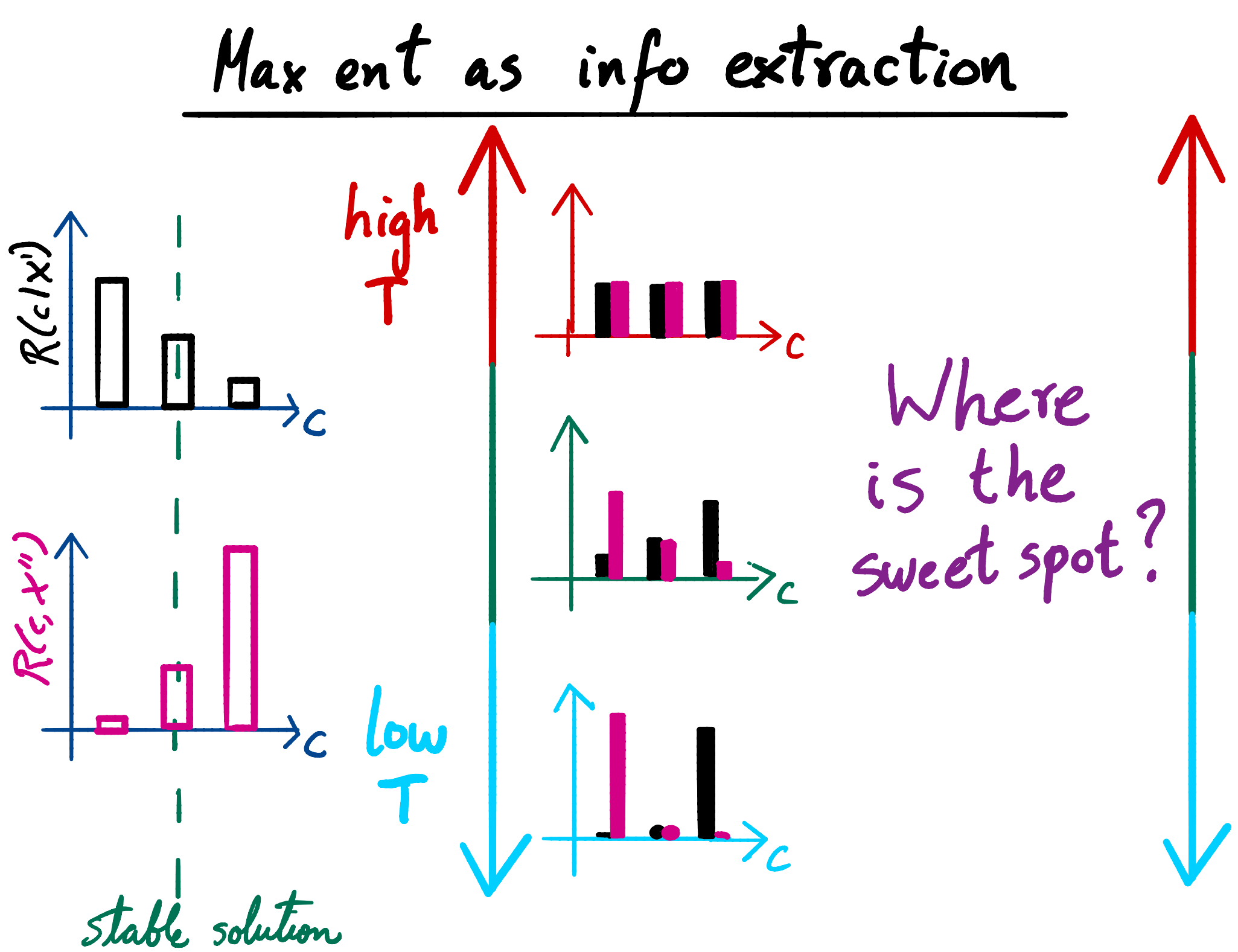
similar posteriors

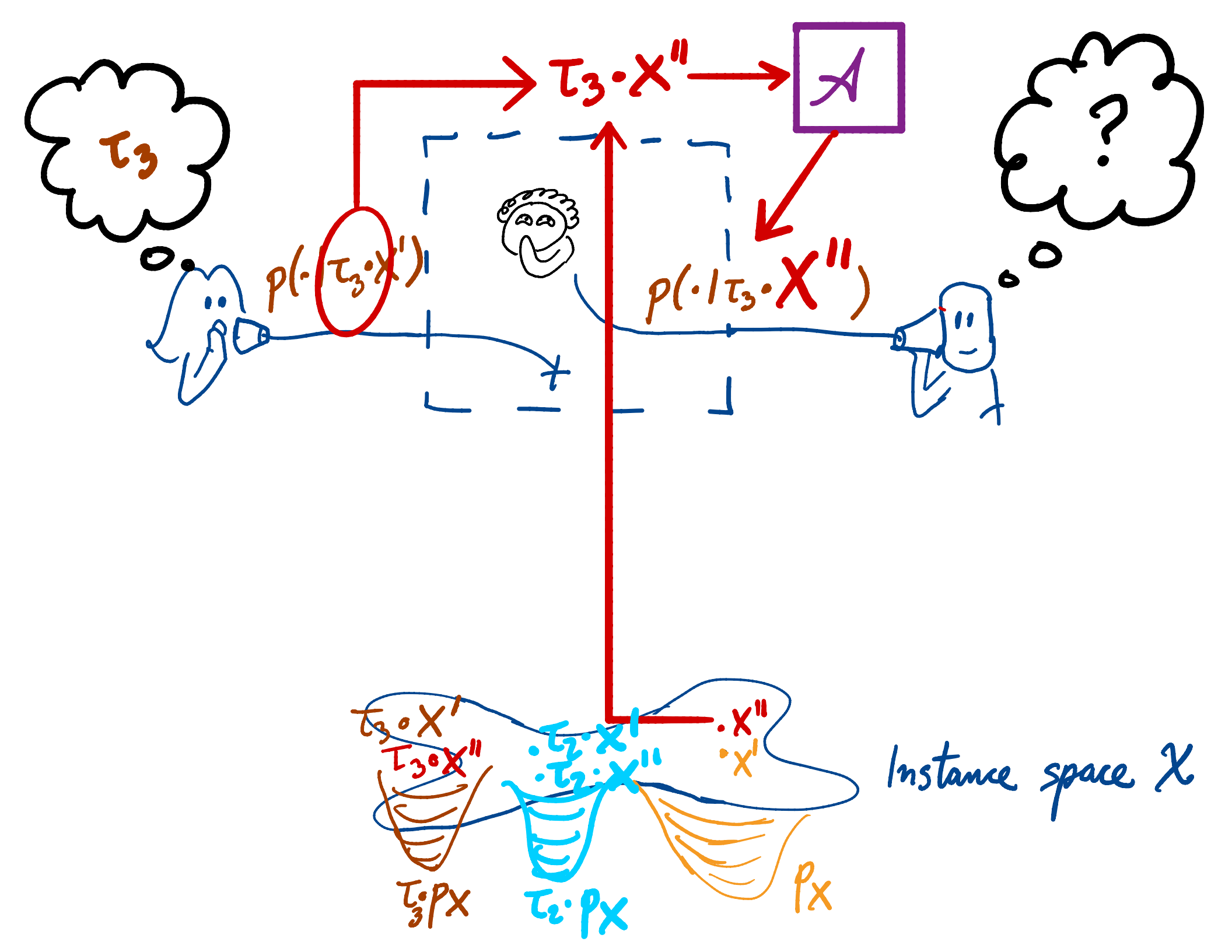
robust & poor fit / not info

different posteriors

not robust & fit / not info

Max ent as info extraction

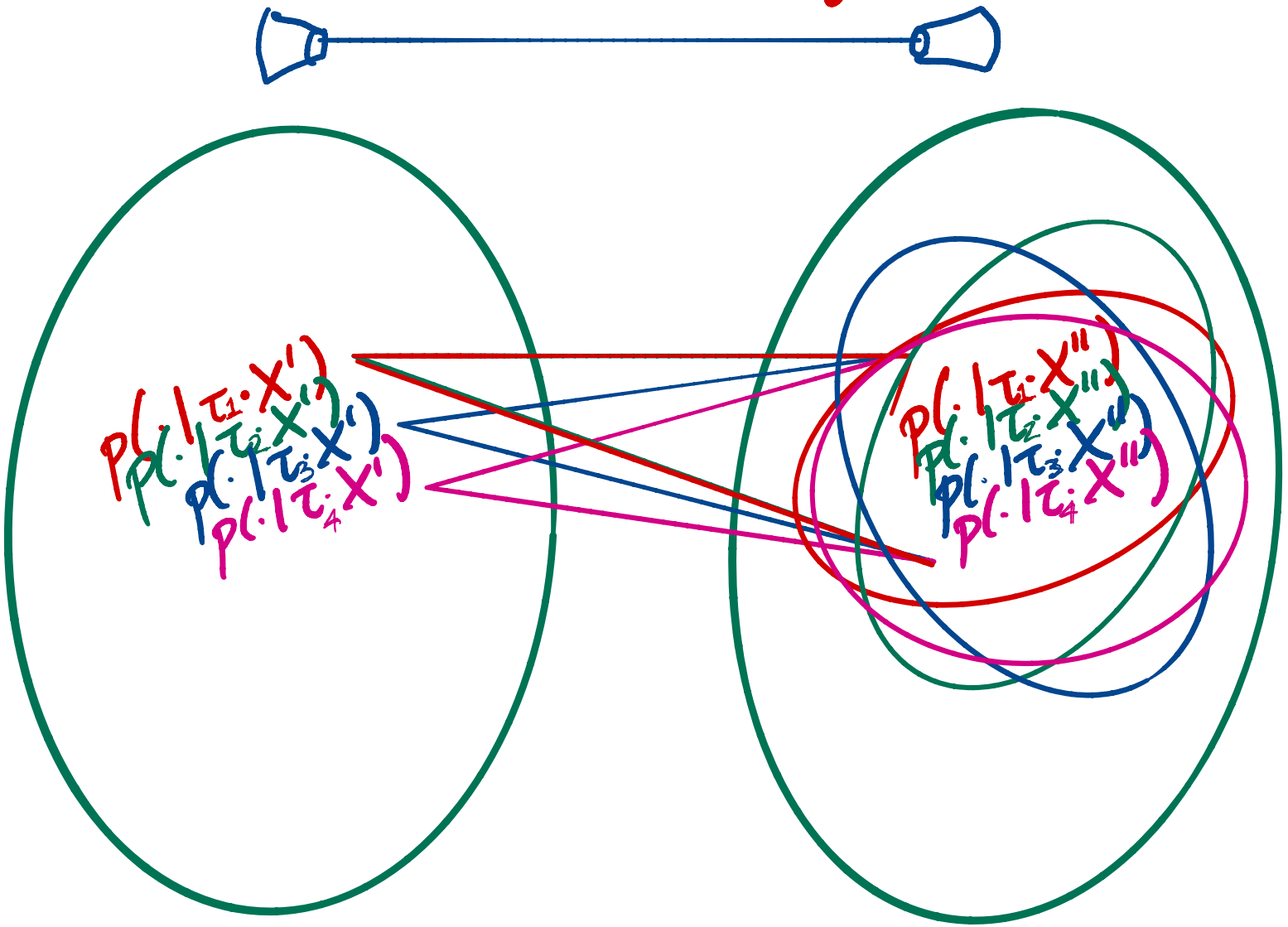



τ_3
$\tau_3 \cdot X''$
$\mathcal{A}$
?
$p(\cdot|\tau_3 \cdot X')$
$p(\cdot|\tau_3 \cdot X'')$
$\tau_3 \cdot X'$
$\tau_3 \cdot X''$
$\tau_2 \cdot X'$
$\tau_2 \cdot X''$
$\cdot X''$
$\cdot X'$
$\tau_3 \cdot p_X$
$\tau_2 \cdot p_X$
p_X
Instance space $\mathcal{X}$

high T: too close codewords and narrow typical sets

---

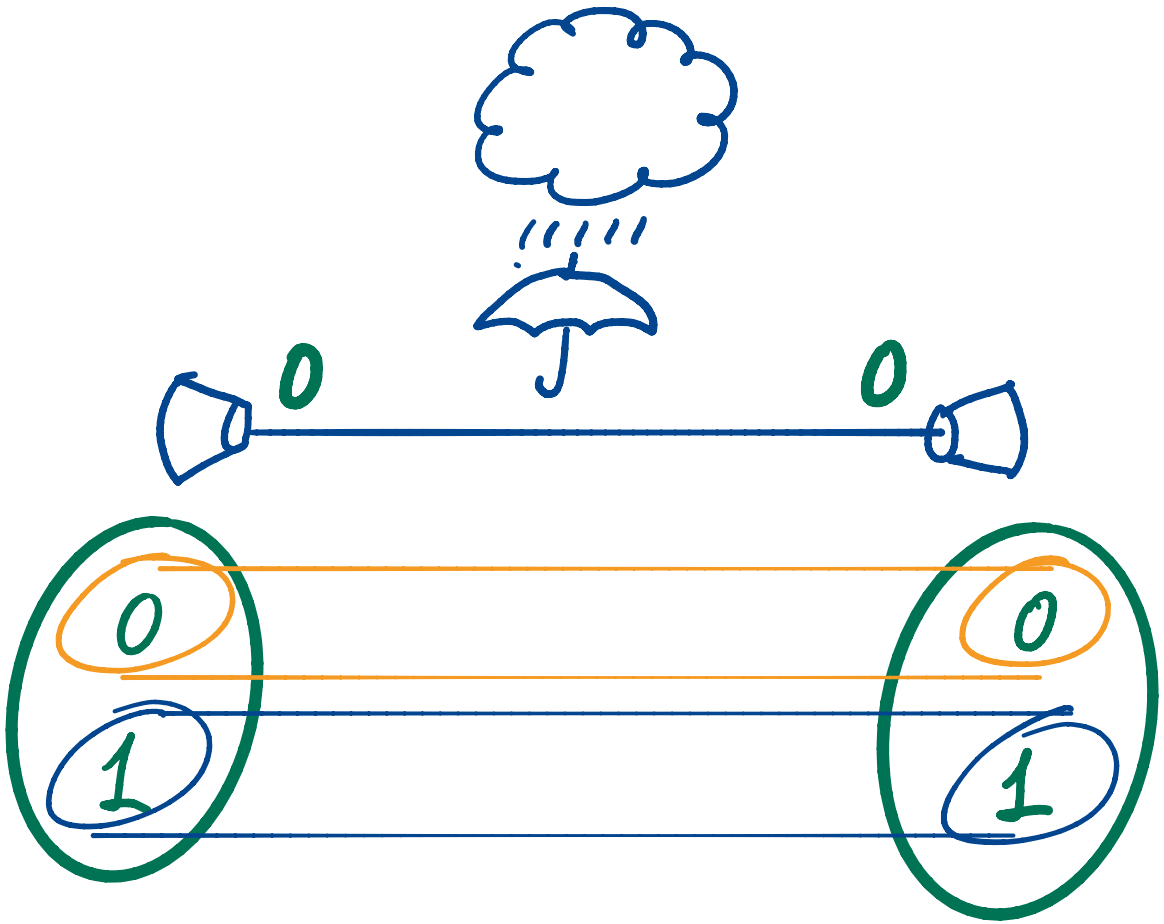
few messages



high T: few codewords and narrow typical sets

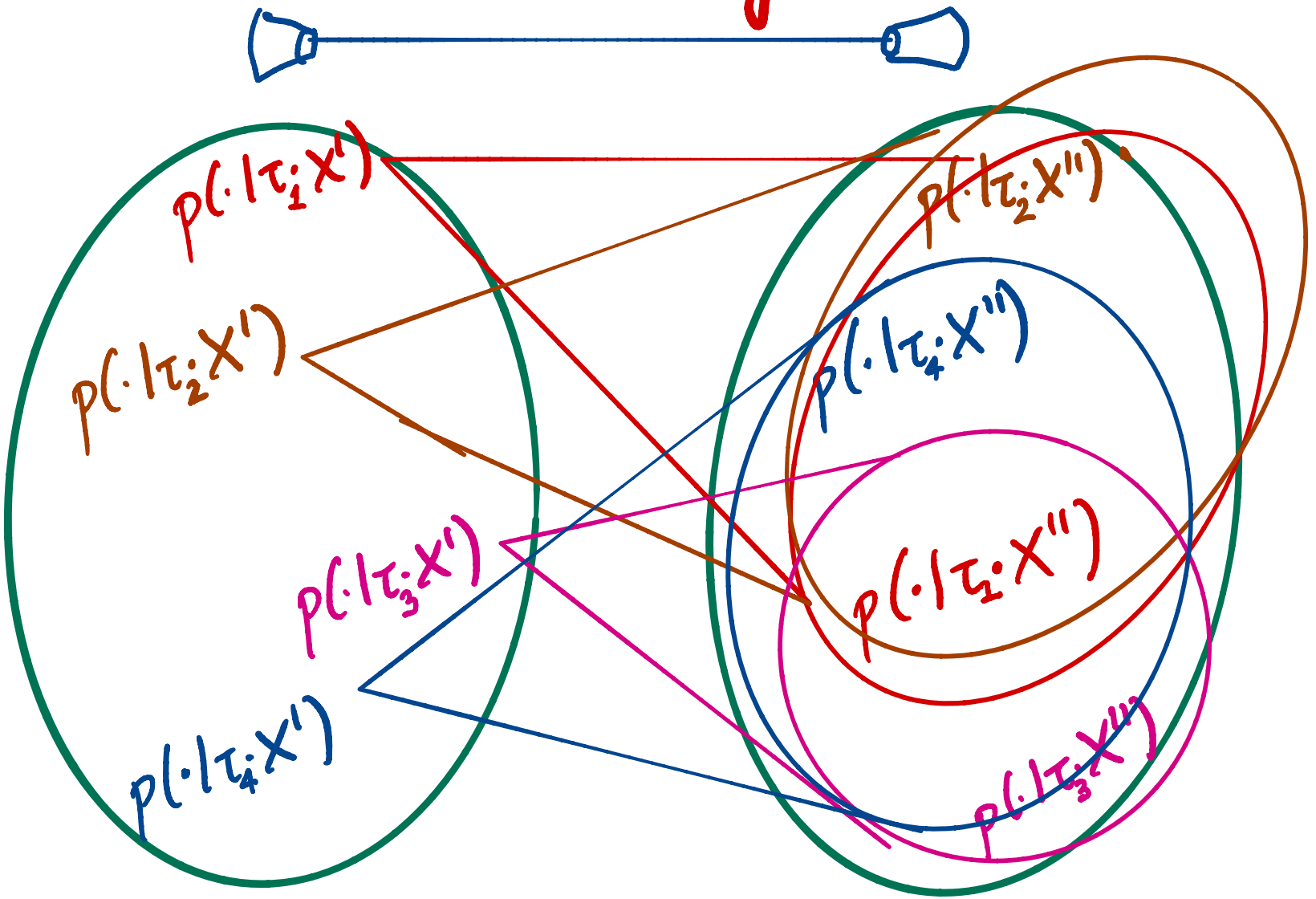
---

few messages



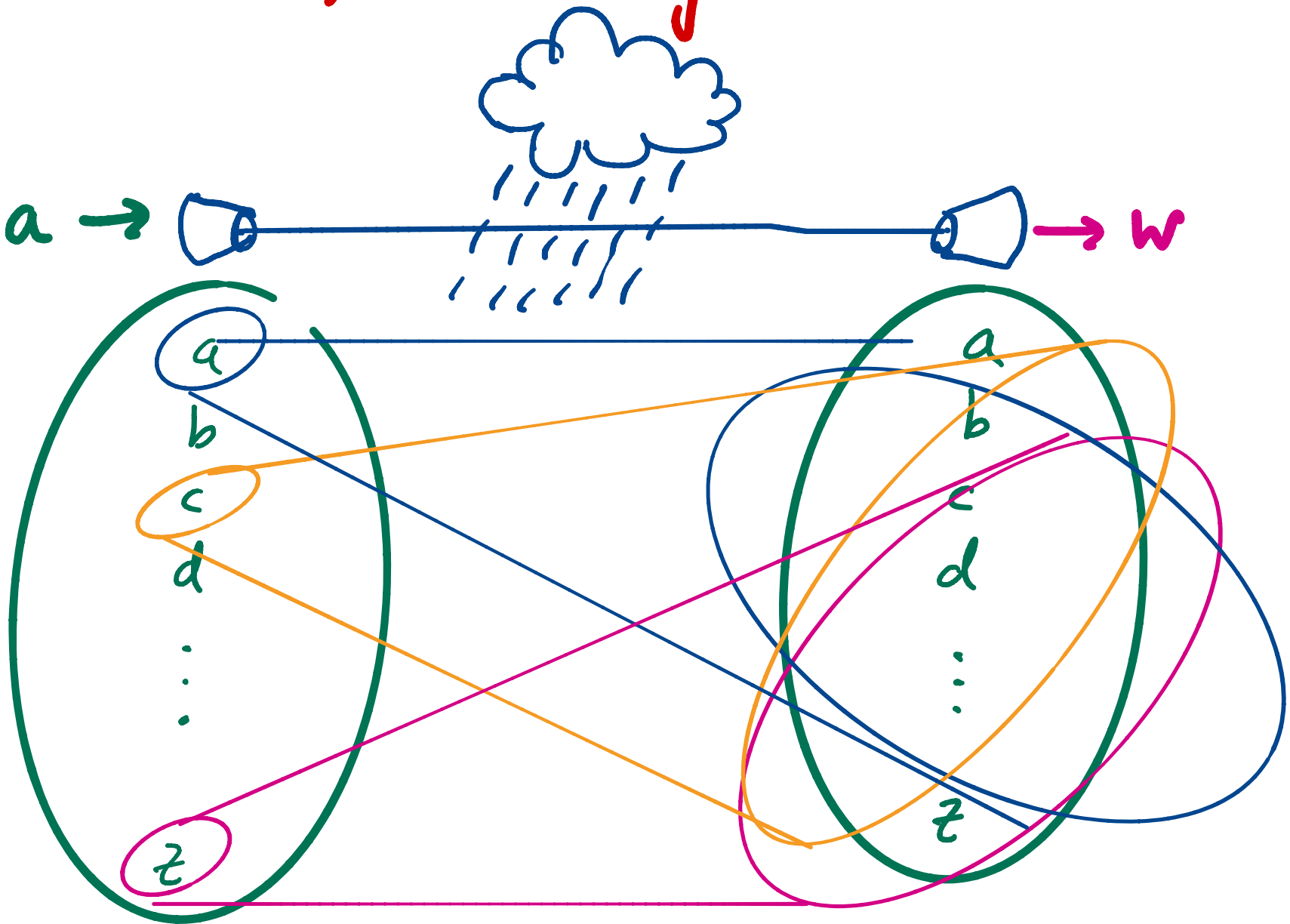
low T: distant posteriors and too wide typical sets

few messages



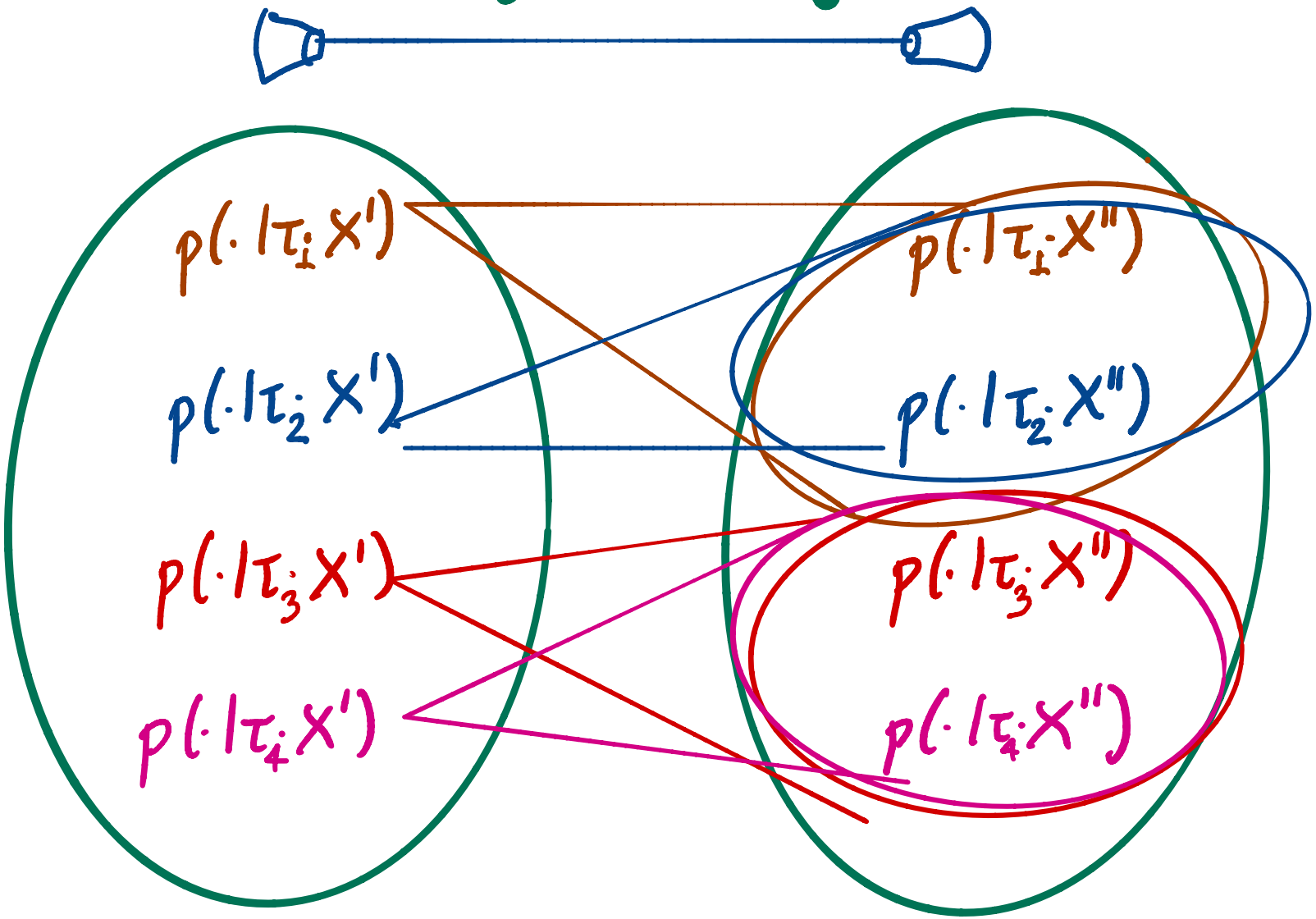
low T: many codewords and too wide typical sets

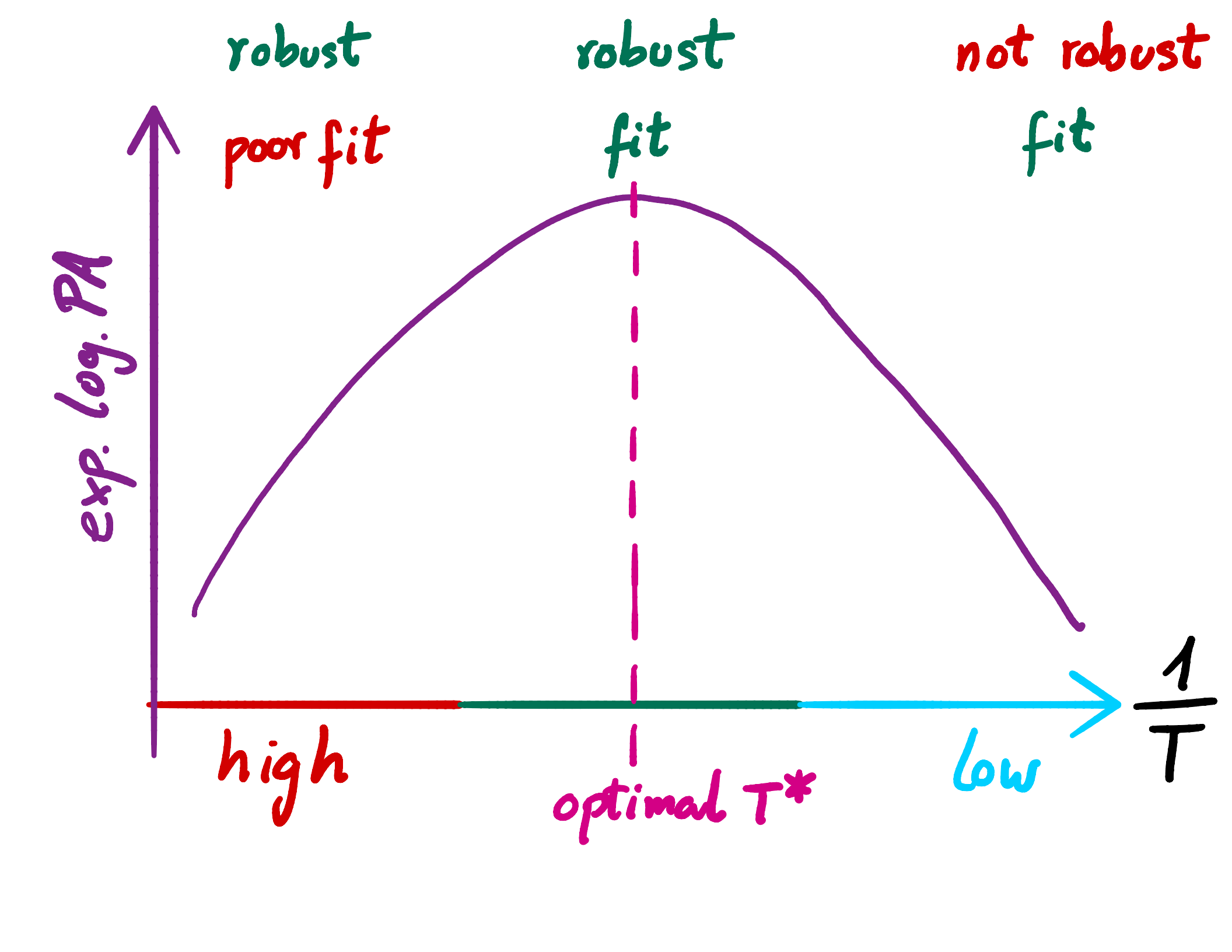
few messages



optimal T: distant posteriors and narrow typical sets

many messages!




robust
poor fit
robust
fit
not robust
fit
exp. log. PA
high
optimal T*
low
1
T

Posterior agreement recommends

$$T^* = \arg\max_T \text{ exp. log. PA}$$

$$\uparrow \; \mathbb{E}_{x',x''}[\log(|C| k(x',x''))]$$

$$\approx \arg\max_T \text{ emp. log. PA}$$

$$= \arg\max_T \log(|C|\, k(x',x''))$$

$$= \arg\max_T k_T(x',x'')$$

with $k_T(x',x'') = \langle p_T(\cdot|x'), p_T(\cdot|x'') \rangle$

Max entropy + PA

Problem: find $\min_c \mathbb{E}_X[R(c, X)]$, given X', X''.

Approach:

1. Compute $p_{T^*}(\cdot|X') \propto \exp\left(-\frac{1}{T^*} R(\cdot, X')\right)$

   $p_{T^*}(\cdot|X'') \propto \exp\left(-\frac{1}{T^*} R(\cdot, X'')\right)$

2. $\hat{c} \overset{\$}{\leftarrow} \frac{1}{Z} p_{T^*}(\cdot|X') \odot p_{T^*}(\cdot|X'')$

$$T^* = \arg\max_T K_T(X', X'')$$

Experiments comparing ME vs ERM

$$C = \{1, \ldots, s, s+1, \ldots, s+u\}$$

$$\vec{X} = (\underbrace{X_1, \ldots, X_s}_{\text{stable}}, \underbrace{\tilde{X}_1, \ldots, \tilde{X}_u}_{\text{unstable}})$$

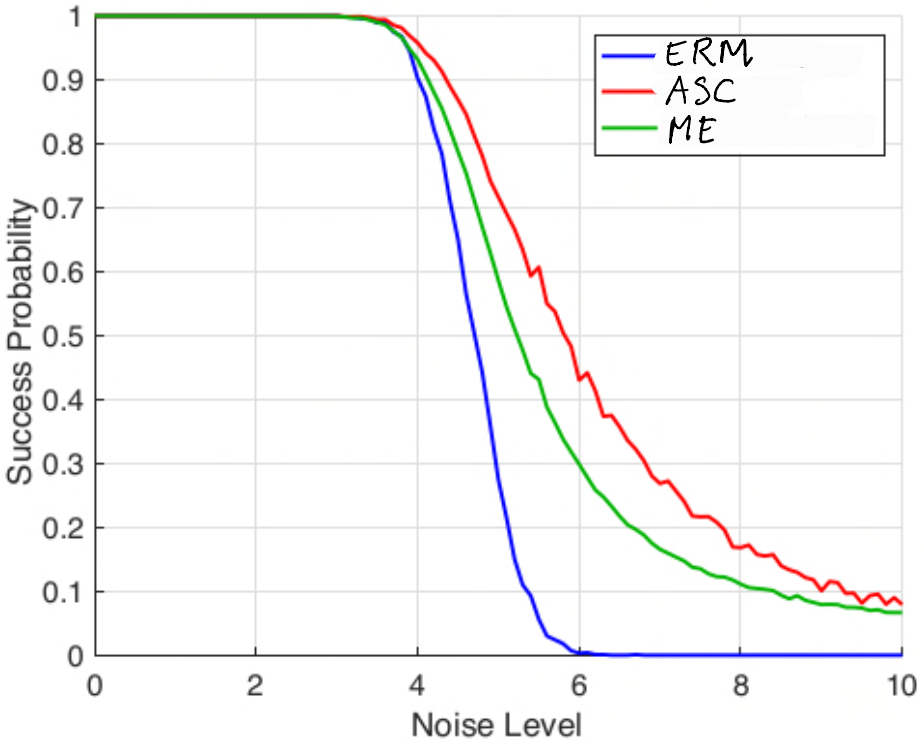
noise parameter

$$X_i \sim \mathcal{N}(1, 1) \qquad \tilde{X}_i \sim \mathcal{N}(10, \sigma^2)$$

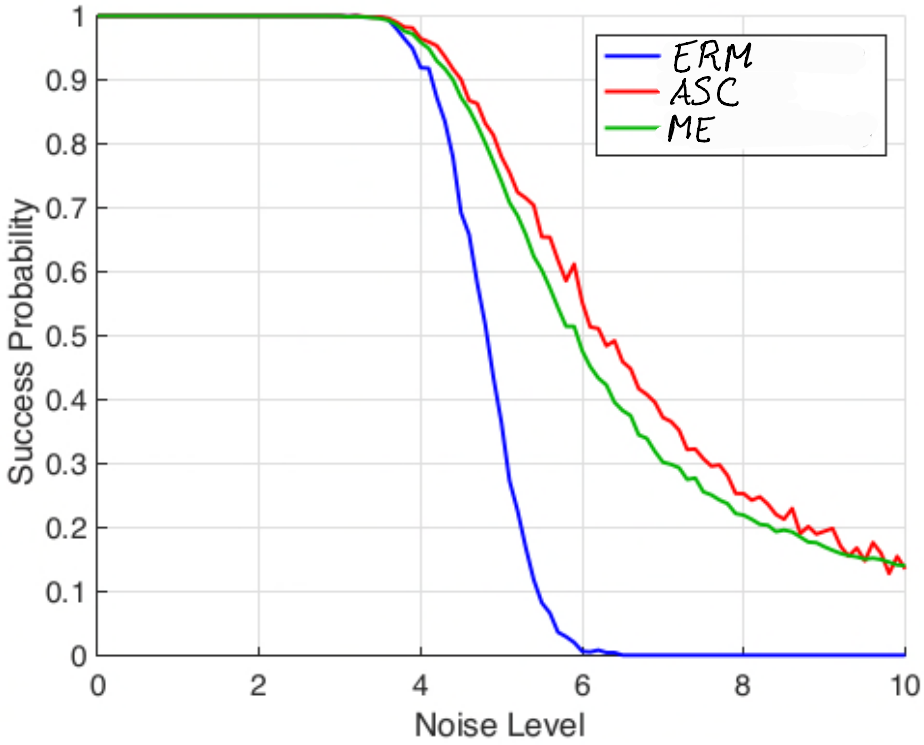
Observe that if $c \leq s$ then $\mathbb{E}_{\vec{X}}[R(c, \vec{X})] = \mathbb{E}[X_c] = 1$

if $s < c \leq s+u$ then $\mathbb{E}_{\vec{X}}[R(c, \vec{X})] = \mathbb{E}[X_c] = 10$

Alex's experiments on ME vs ERM using only two instances X' and X''.



(a) 5% of solutions are stable.

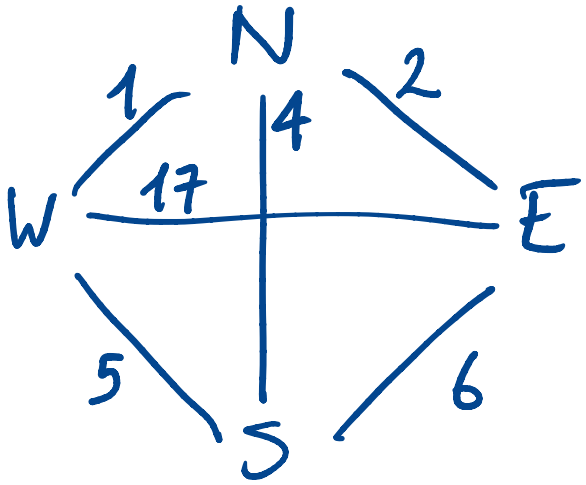


(b) 10% of solutions are stable.

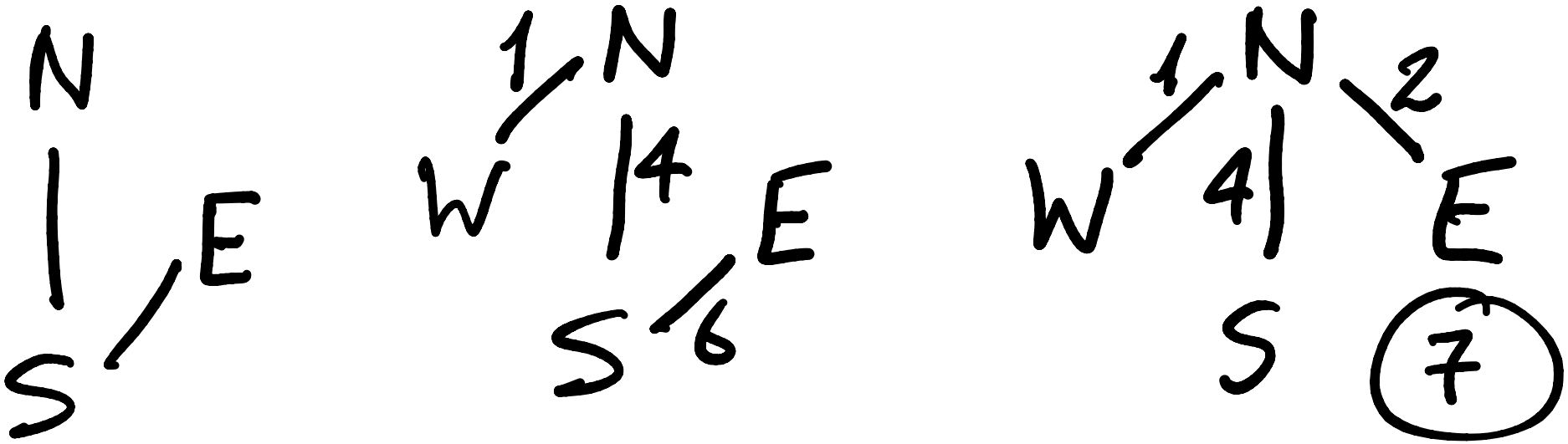
Minimum spanning tree algorithms: Regularization by stopping

A. Gronskiy and J. Buhmann

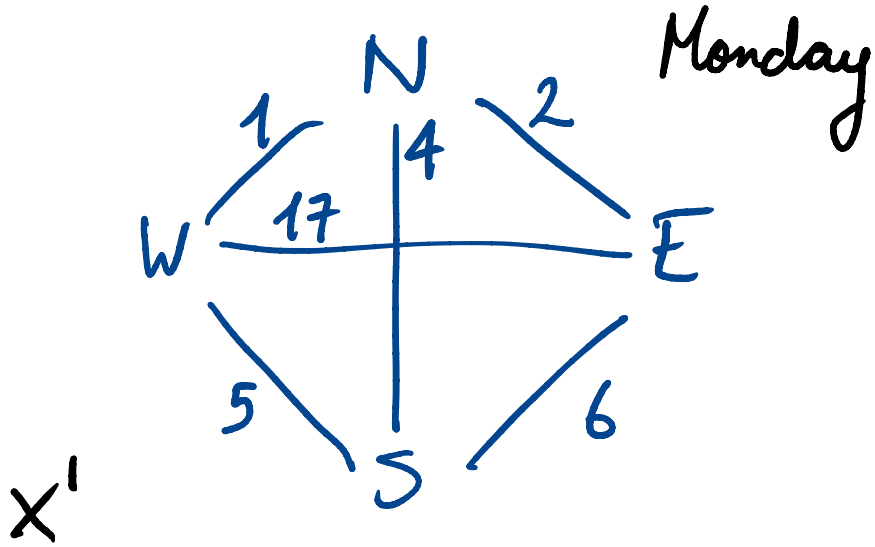
Motivation



What's the minimum spanning tree?



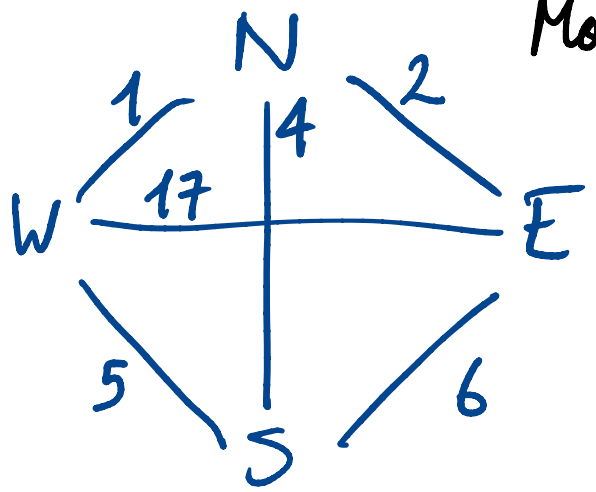
Motivation



What's the minimum spanning tree?

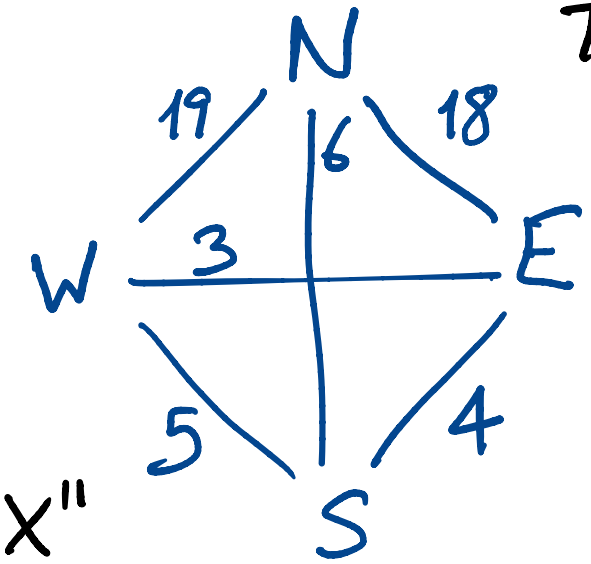
Motivation

Monday



X'

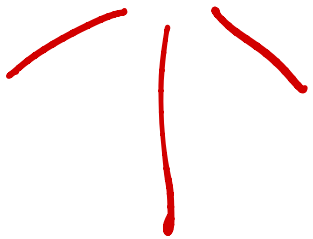
Tuesday



X''

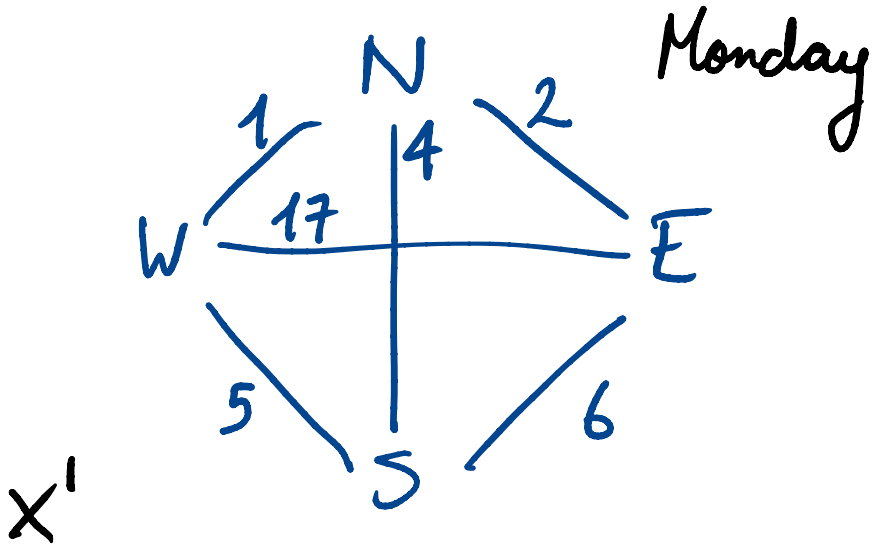
What's the minimum spanning tree?

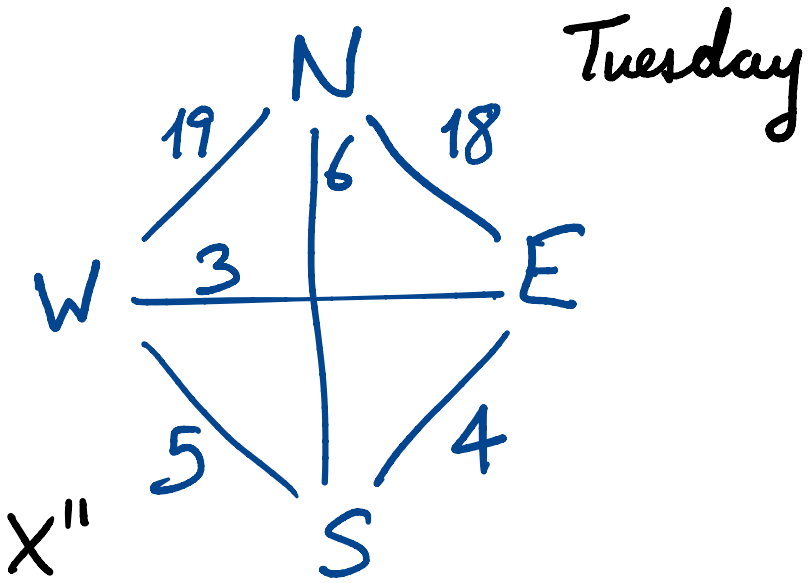
Monday



Tuesday

Motivation





What's the minimum spanning tree?

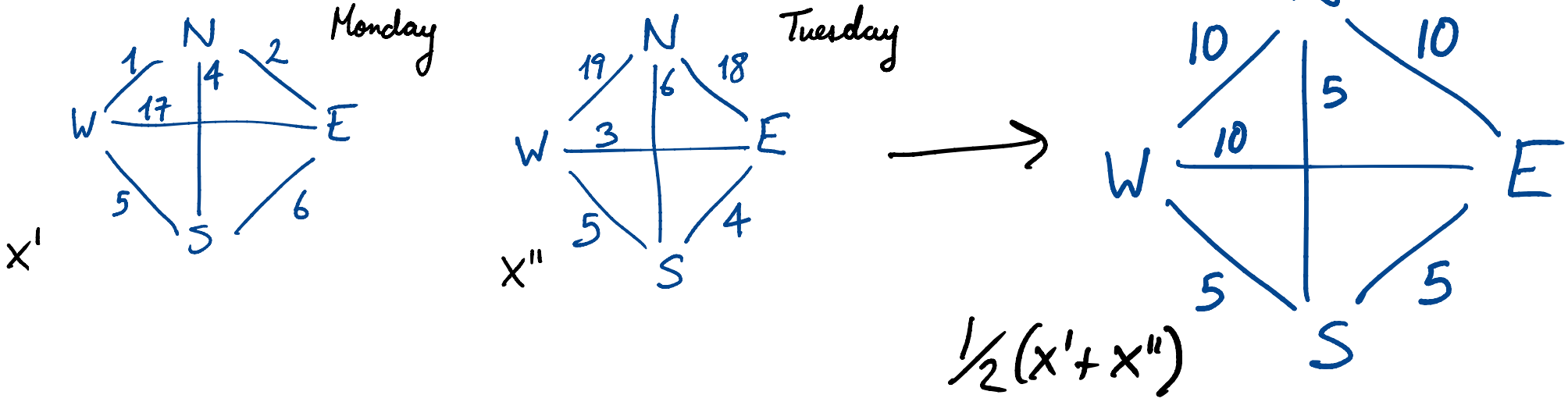
$$\min_{c \in G} \mathbb{E}_X[R(c, X)]$$

- G: all spanning trees
- X: city graph
- $R(c, X)$: c's weight in X

Natural choice (ERM):

$$\mathbb{E}_X[R(c,X)] \approx \frac{1}{2}\Big(R(c,X') + R(c,X'')\Big).$$

$$= R(c, \tfrac{1}{2}(X'+X''))$$



Alternative : ME+PA

Let $p(\cdot|x') \propto \exp\left(-\frac{1}{T^*} R(c, x')\right)$

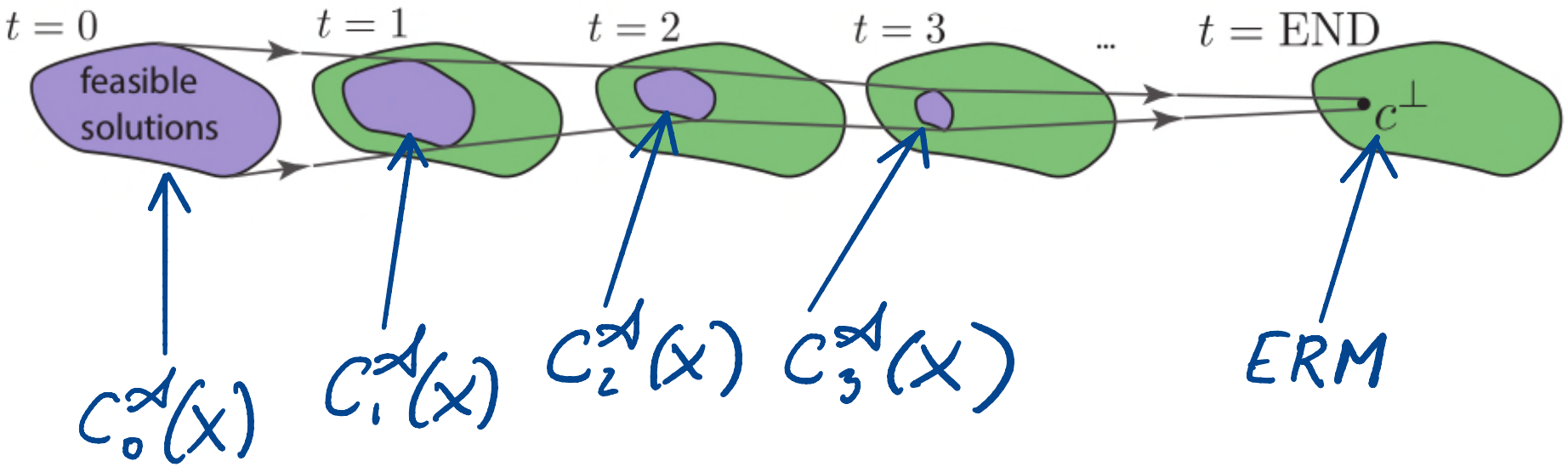
$p(\cdot|x'') \propto \exp\left(-\frac{1}{T^*} R(c, x'')\right)$

$T^* = \arg\max_T K_T(x', x'')$

Problem: computing $p(\cdot|x')$ is intractable as there are n^{n-2} trees!

Robust solving via early stopping + PA

Many MST algos are "contractive" and efficient.



Def: Algo $\mathcal{A}$ is **contractive** if, given X, its execution can be expressed as a sequence

$$C = C_0^{\mathcal{A}}(X) \supseteq C_1^{\mathcal{A}}(X) \supseteq \dots \supseteq C_T^{\mathcal{A}}(X) = \{c^\perp\}$$

$$C = C_0^{\mathcal{A}}(X) \supseteq C_1^{\mathcal{A}}(X) \supseteq \dots \supseteq C_T^{\mathcal{A}}(X) = \{c^{\perp}\}$$

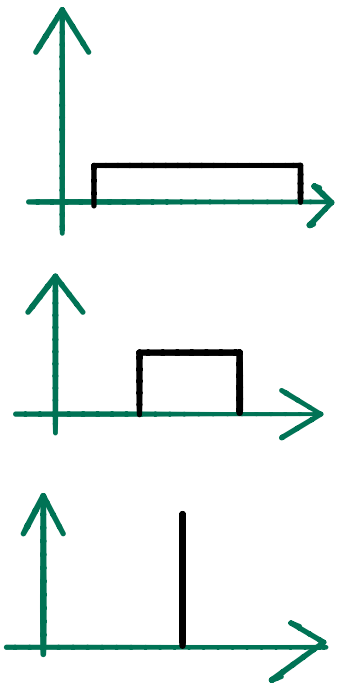
$$p_0^{\mathcal{A}}(\cdot \mid X),\ p_1^{\mathcal{A}}(\cdot \mid X),\ \dots,\ p_T^{\mathcal{A}}(\cdot \mid X)$$

At each step, a uniform distr over $C_t^{\mathcal{A}}(X)$ is computed

$$p_0^{\mathcal{A}}(\cdot \mid X) \leftarrow \mathrm{unif}(C)$$

$$p_t^{\mathcal{A}}(\cdot \mid X) \leftarrow \mathrm{unif}(C_t^{\mathcal{A}}(X))$$

$$p_T^{\mathcal{A}}(\cdot \mid X) \leftarrow \mathrm{unif}(\{c^{\perp}\})$$



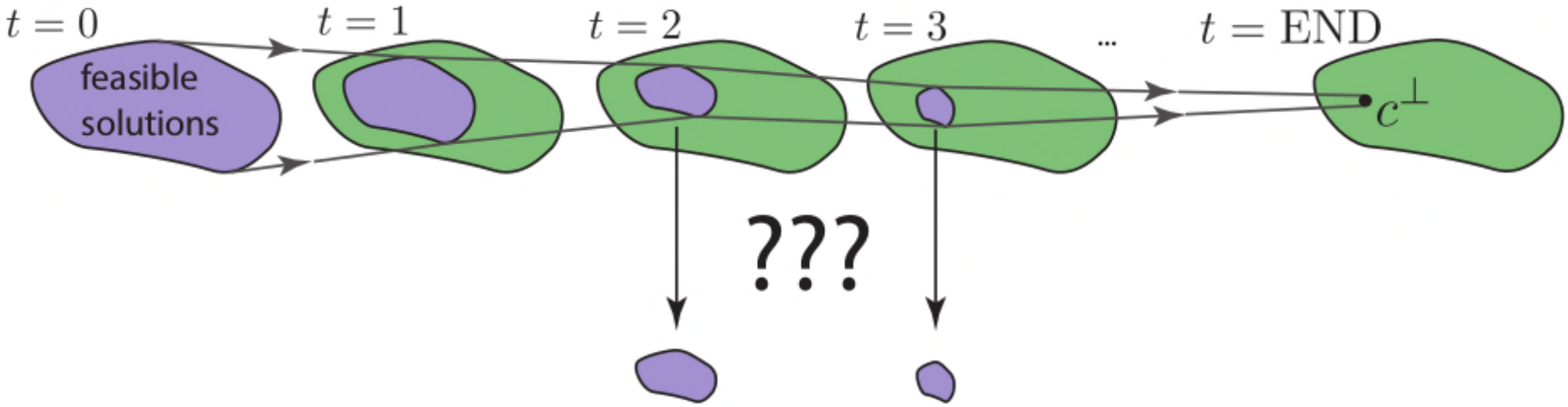
We can apply PA
to build more robust MST
algorithms

$$C = C_0^{\mathcal{A}}(X) \supseteq C_1^{\mathcal{A}}(X) \supseteq \dots \supseteq C_T^{\mathcal{A}}(X) = \{c^{\perp}\}$$

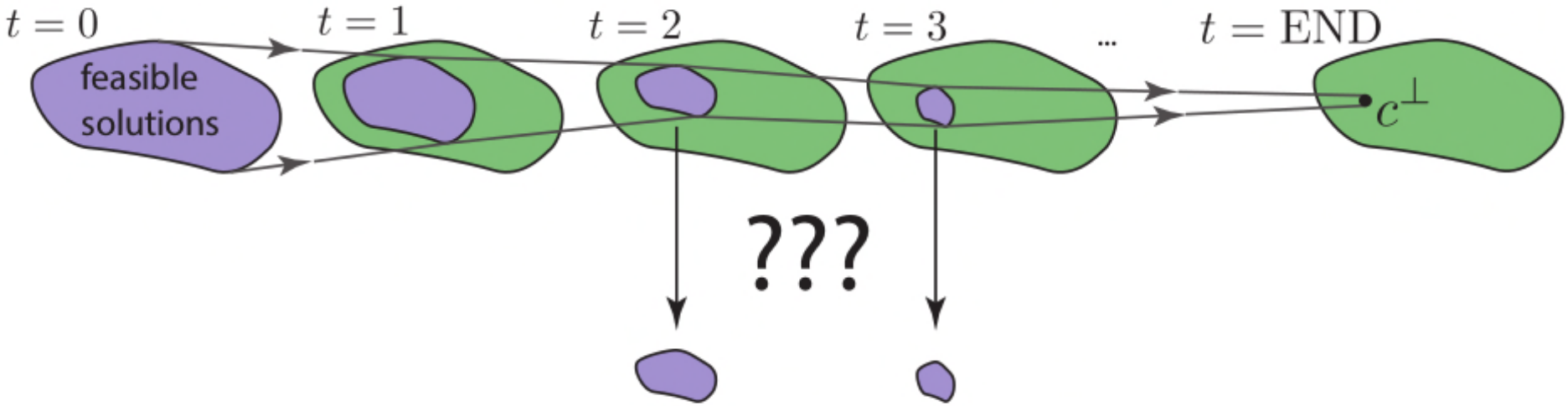
$$p_0^{\mathcal{A}}(\cdot|X),\ p_1^{\mathcal{A}}(\cdot|X),\ \dots,\ p_T^{\mathcal{A}}(\cdot|X)$$

STOP

Idea: Stop the algorithm earlier at $t < T$ and output $p_t^{\mathcal{A}}(\cdot|X)$.



When do we stop?

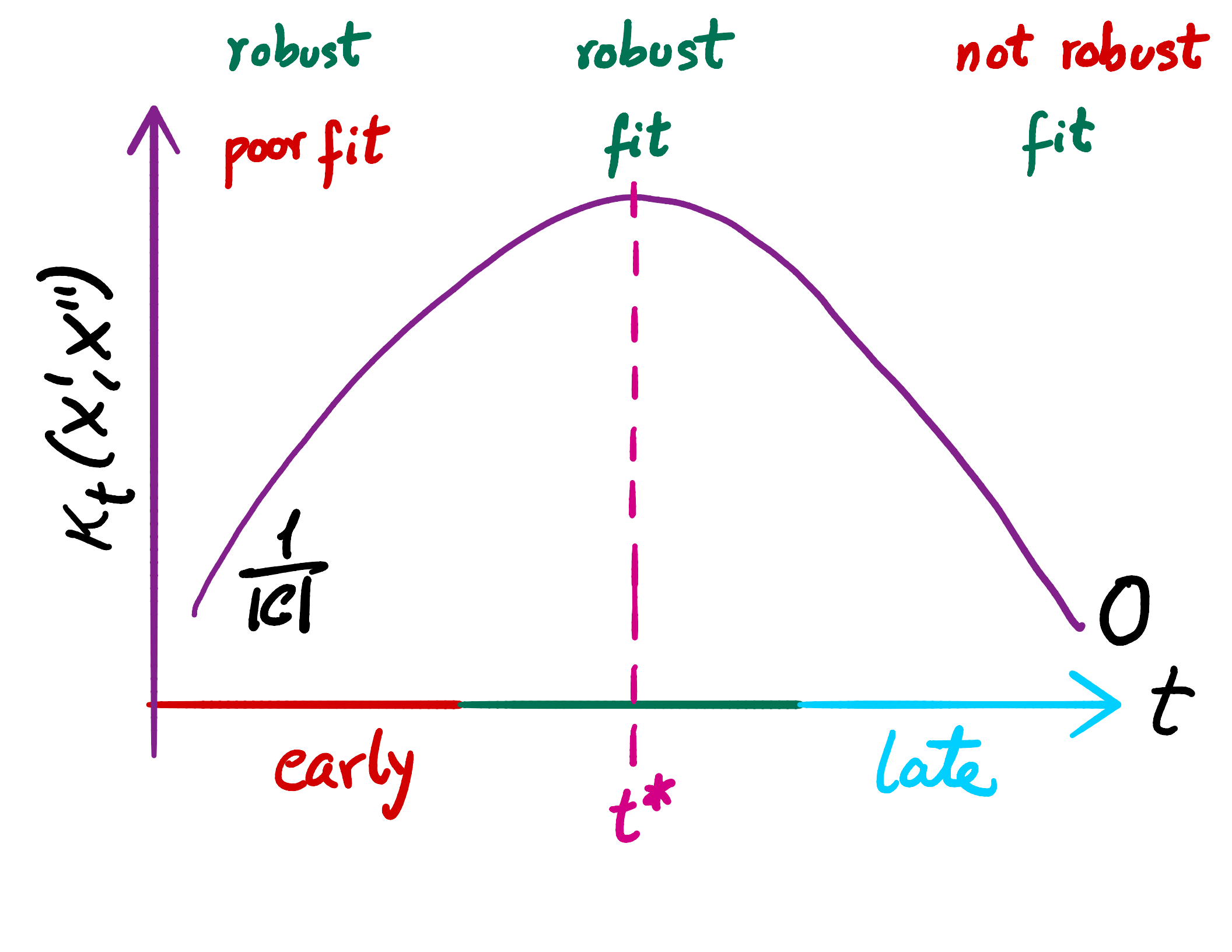


When do we stop?

$$t^* = \arg\max_t \kappa_t^{\mathcal{A}}(x', x'')$$

$$\kappa_t^{\mathcal{A}}(x', x'') = \sum_c p_t^{\mathcal{A}}(c|x') p_t^{\mathcal{A}}(c|x'') = \frac{|C_t^{\mathcal{A}}(x') \cap C_t^{\mathcal{A}}(x'')|}{|C_t^{\mathcal{A}}(x')| |C_t^{\mathcal{A}}(x'')|}$$

$$\frac{|C \cap C|}{|C||C|} = \frac{1}{|C|}$$


robust
poor fit
robust
fit
not robust
fit
$\kappa_t(x', x'')$
$\frac{1}{|c|}$
0
t
early
t^*
late

Regularized MST algo

Input: X', X'', MST algo $\mathcal{A}$

1. Run $\mathcal{A}$ to compute $C_t^{\mathcal{A}}(X')$ and $C_t^{\mathcal{A}}(X'')$, for all t.

2. $t^* \leftarrow \arg\max_t \dfrac{|C_t^{\mathcal{A}}(X') \cap C_t^{\mathcal{A}}(X'')|}{|C_t^{\mathcal{A}}(X')|\,|C_t^{\mathcal{A}}(X'')|}$

3. $c^* \overset{\$}{\leftarrow} \mathrm{Unif}\left(C_{t^*}^{\mathcal{A}}(X') \cap C_{t^*}^{\mathcal{A}}(X'')\right)$

Which MST algo is the best for this?

Which reg MST algo is the best?

MST algos: Prim, Kruskal, reverse-delete.

* Experimental assessment of PA.
* Experimental comparison.

  PA rankings match with performance rankings.

* Analysis

Experiments: PA of the req. MST algos

* For $\sigma \in \{0, \ldots, 80\}$, a "random" complete graph with 60 vertices and the exp. log. PA for each algo was calculated.

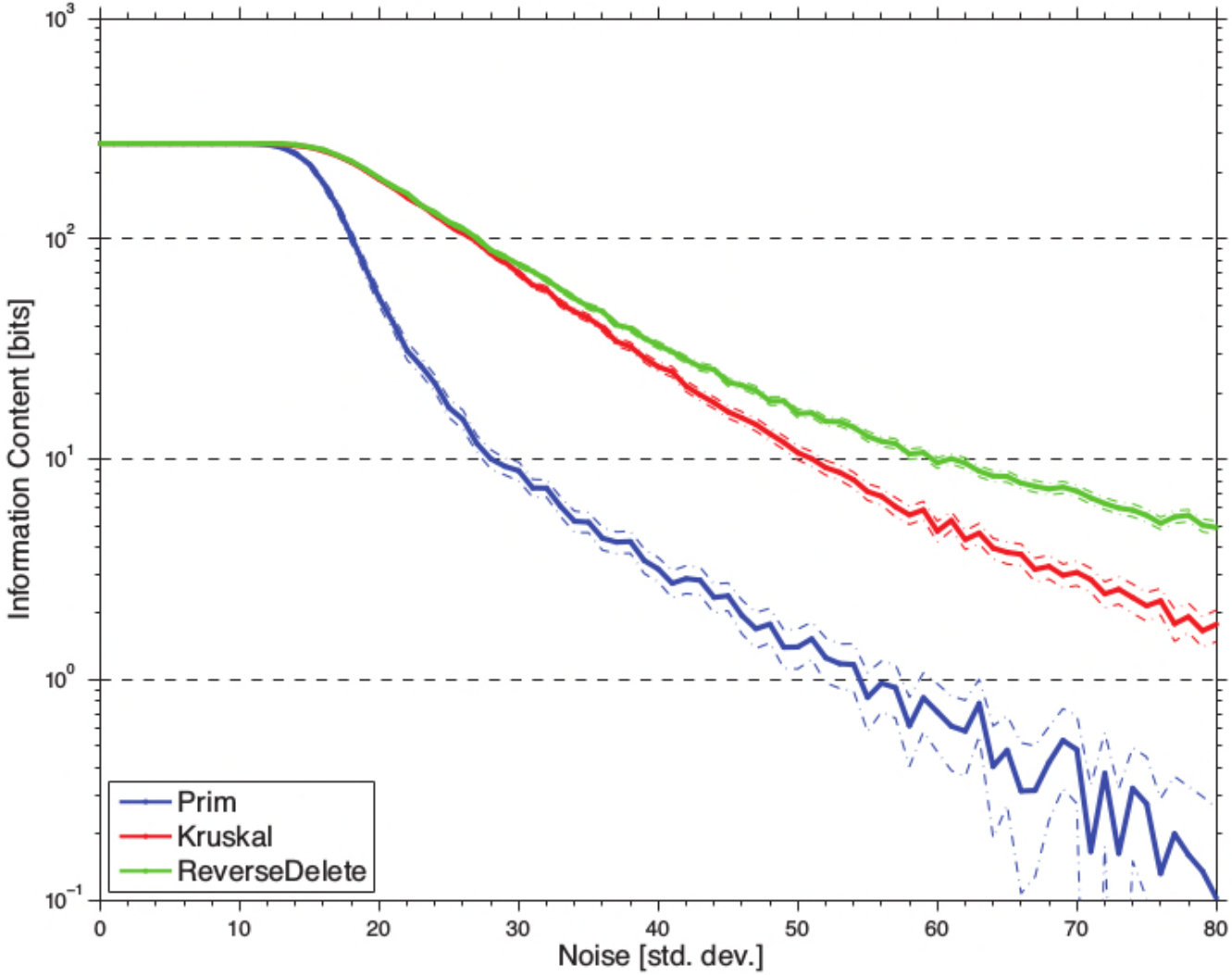
experiment $\rightarrow G = (V, V^2)$ $\quad V = \{1, \ldots, 60\}$

for $v, w \in V$ $\quad$ weight$(v,w) \xleftarrow{\$} \mathcal{N}(100, 100)$.

instance $\rightarrow X = (V, V^2)$

for $v, w \in V$ $\quad$ weight$(v,w) \sim \mathcal{N}($weight$(u,v), \sigma^2)$

Exp. by. PA

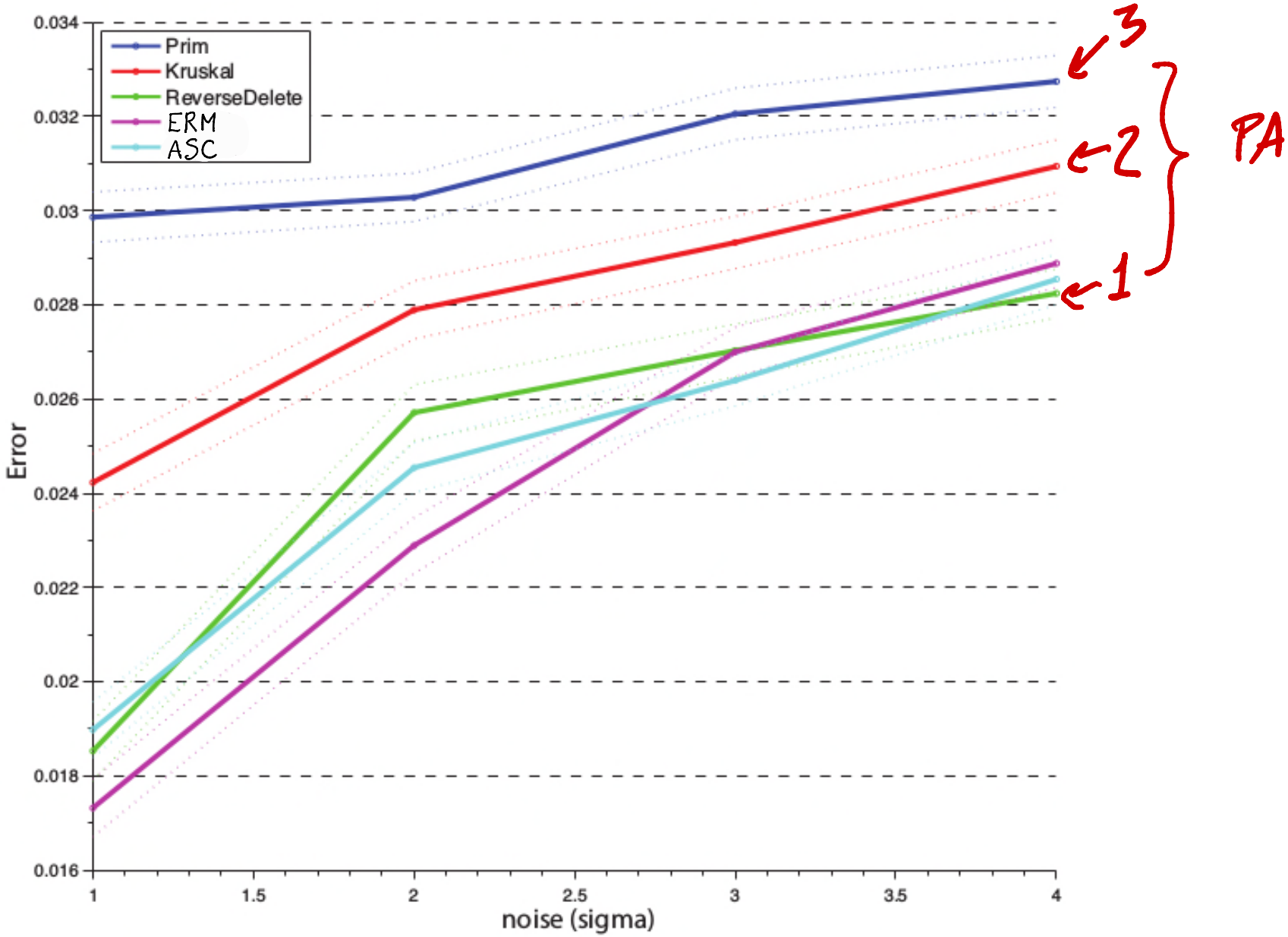

Information Content [bits]
Noise [std. dev.]
Prim
Kruskal
ReverseDelete

Experiments: Performance of the reg. MST algos.

- 400 complete "ground-truth" graphs with 6 vertices

- Each edge weight $\xleftarrow{\$} \mathcal{N}(\mu, \sigma^2)$

- From each graph, two instances X' and X'' were generated by adding noise $\mathcal{N}(0, \sigma_0^2)$ to each edge.



$$\text{Error} = \frac{1}{400} \sum_{i \le 400} |t_i^* \triangle t_i^{\mathcal{A}}|$$

$\triangle$: symmetric diff.

t_i^*: MST

$t_i^{\mathcal{A}}$: ST output by $\mathcal{A}$

MST algorithms and their regularized versions

Prim's algo (growing tree)

- pick an arbitrary vertex v_0
- $T \leftarrow \{(v_0, \arg\min_{v'} \text{weight}(v_0, v'))\}$
- while T not spanning

  Expand T with

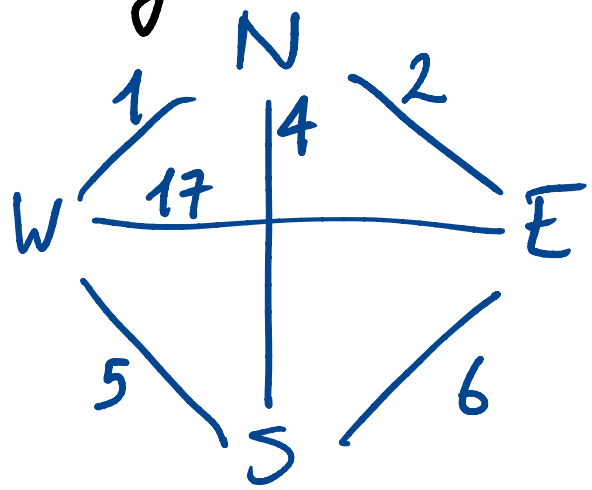
  $$\arg\min_{(u,v)} \text{weight}(u,v)$$

  $$\text{s.t. } u \in T,\ u \notin T$$

  $$T \cup \{(u,v)\} \text{ is a tree.}$$

MST algorithms and their regularized versions

Prim's algo (growing tree)



X'

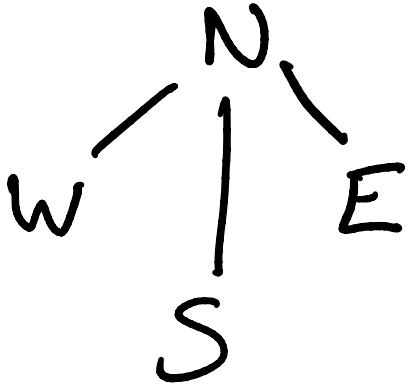
$t=0$

S

$t=1$

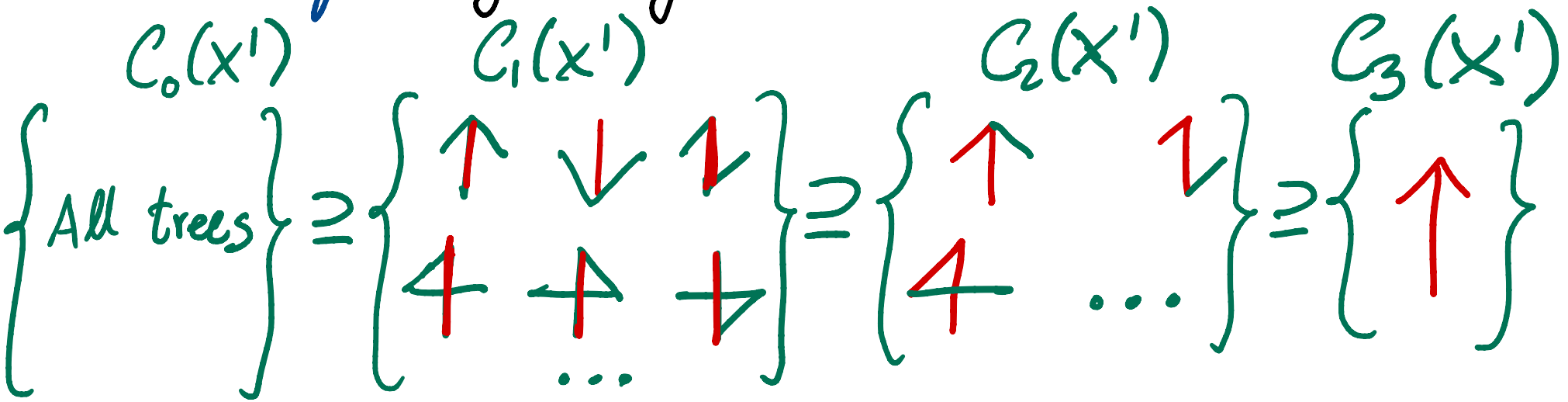
$t=2$

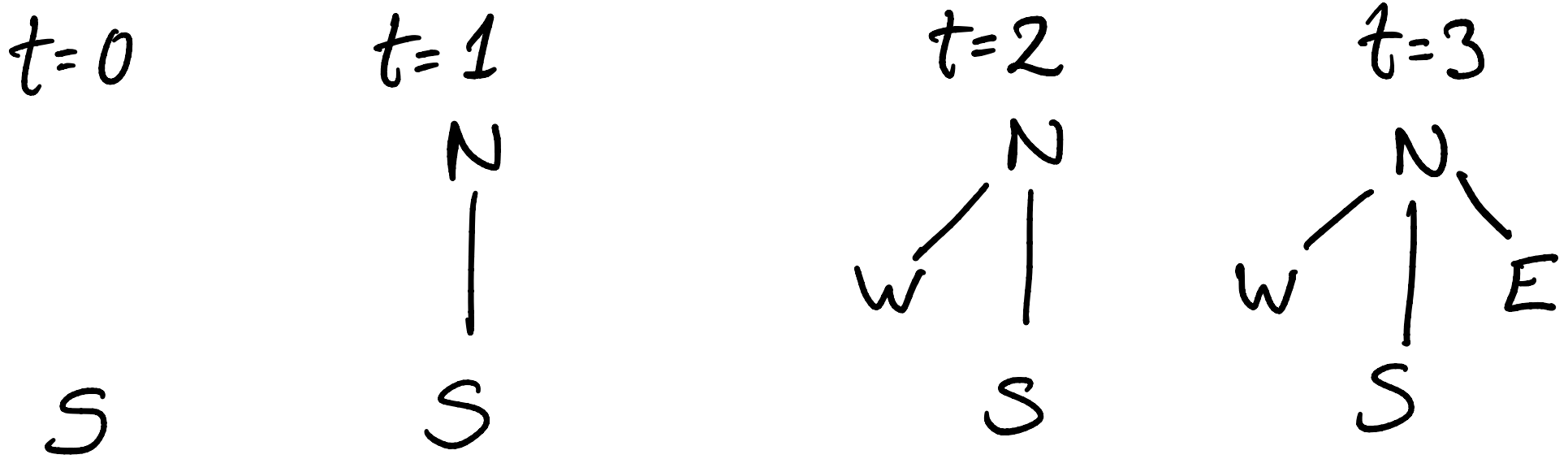
$t=3$



MST algorithms and their regularized versions

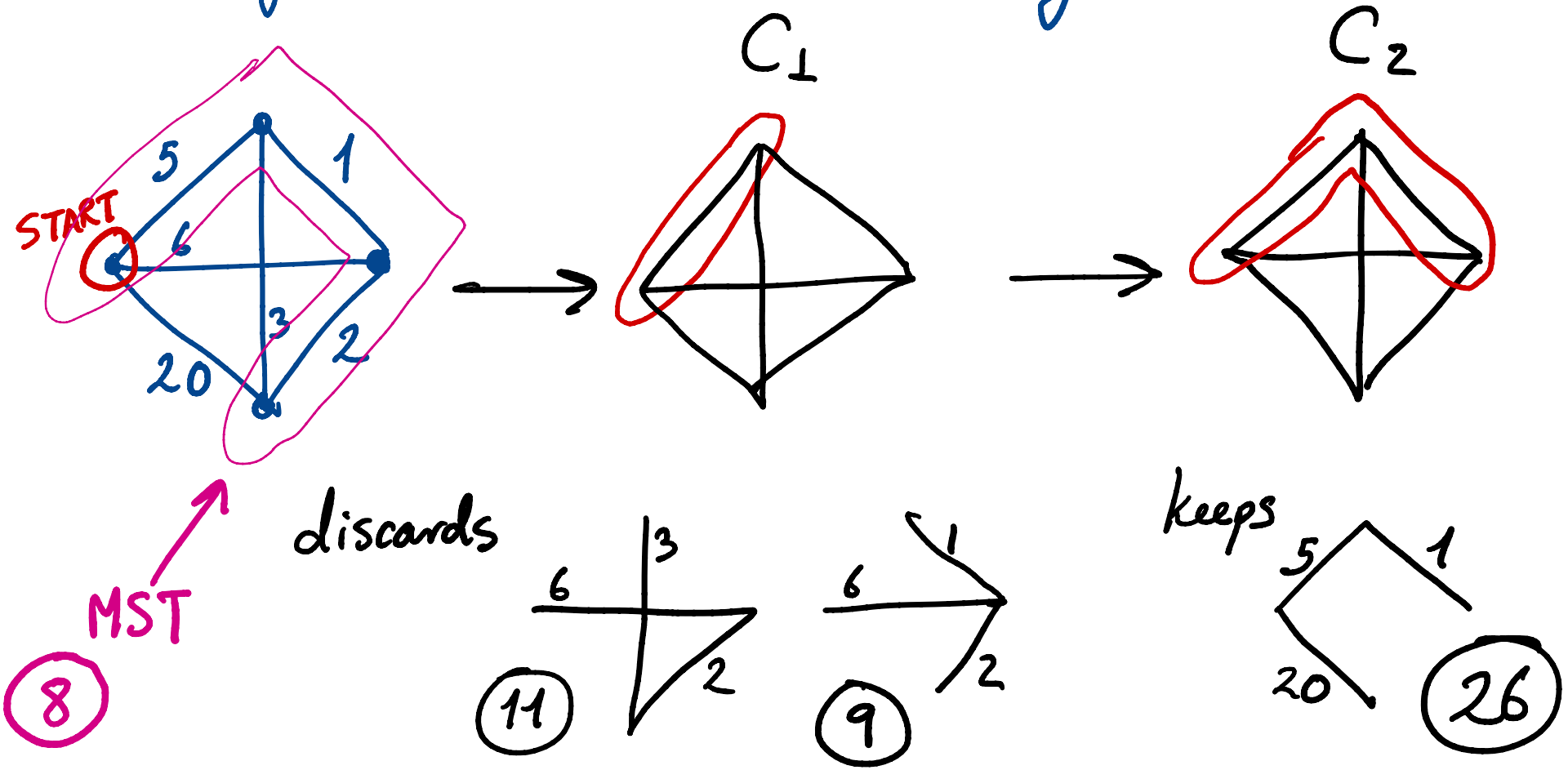
Prim's algo (growing tree)





Prim:

* Only adds edges connected to the current tree
* If you start wrong, then you discard good candidates too early.



MST algorithms and their regularized versions

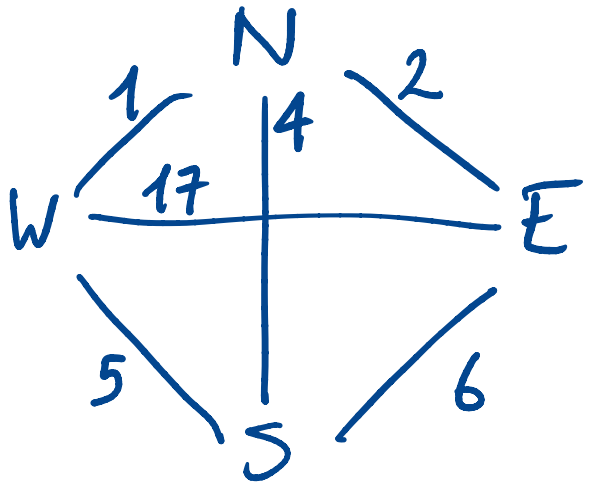
Kruskal's algo (growing forest)

1. $T \leftarrow \phi$
2. While T not spanning

   Expand T with the lightest edge that does not induce a cycle

MST algorithms and their regularized versions

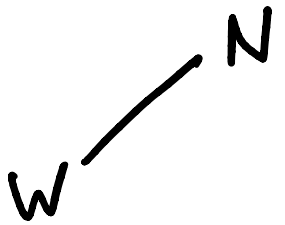
Kruskal's algo (growing forest)



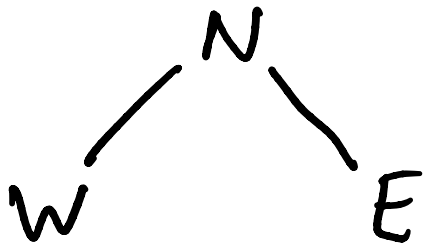
X'

$t=0$

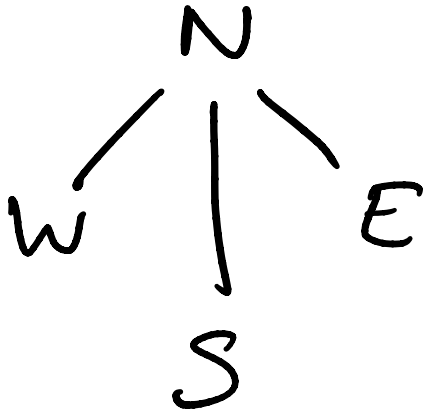
$t=1$



$t=2$

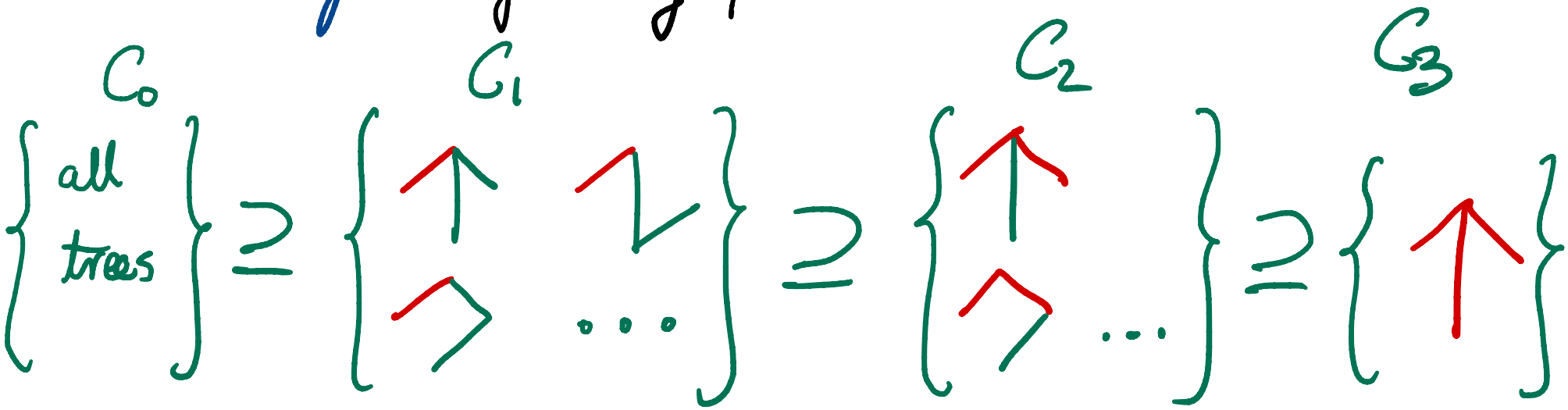


$t=3$

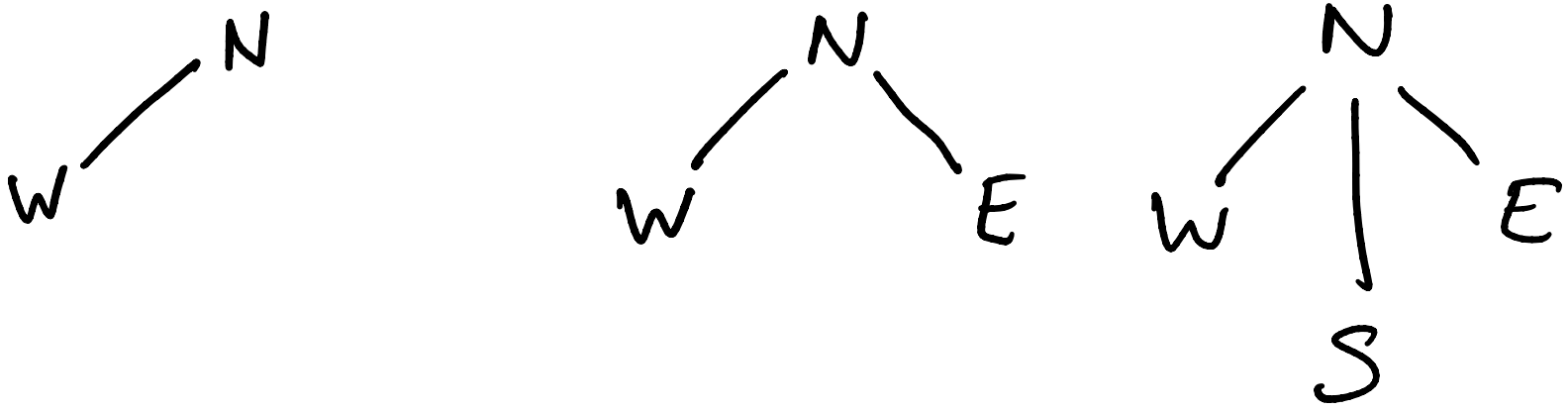


MST algorithms and their regularized versions

Kruskal's algo (growing forest)

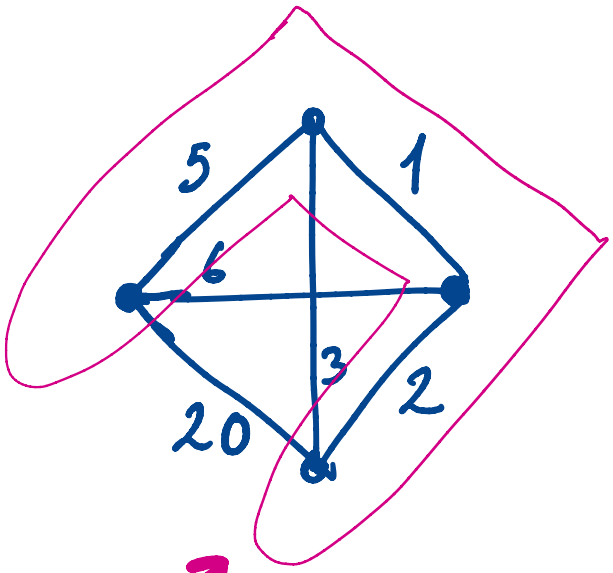


$t=0$ $t=1$ $t=2$ $t=3$



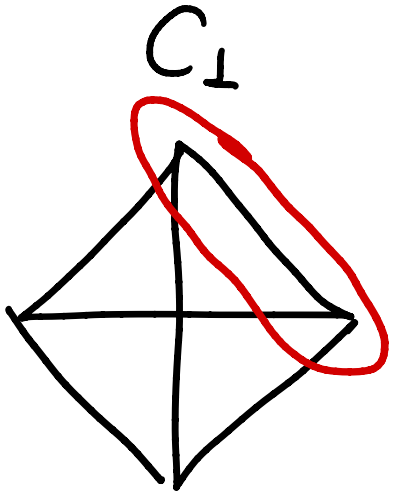
Kruskal:

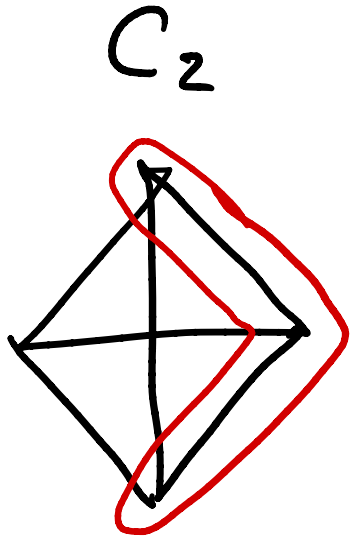
* Does not require selected edges to be connected.
* If you start wrong, you discard good candidates.



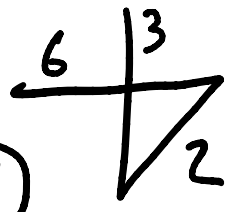
MST

8

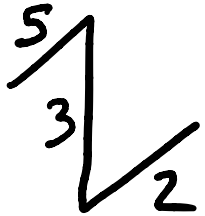




discards:

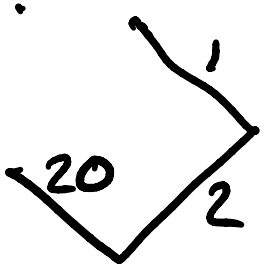


11



10

keeps:



23

MST algorithms and their regularized versions

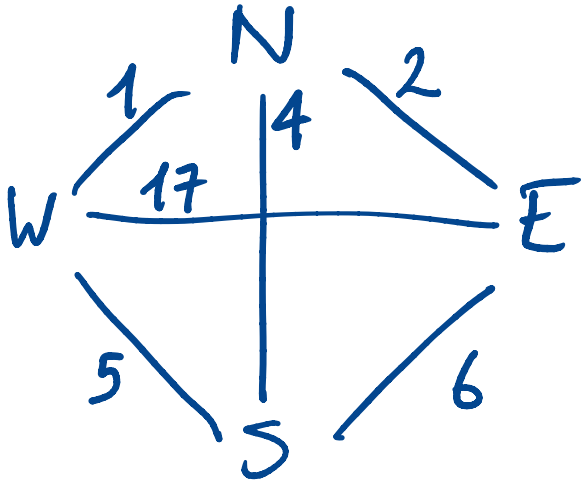
Reverse-delete (reducing graph)

1. $T \leftarrow$ all edges
2. While T is not a tree:

   remove the heaviest edge that does not disconnect the graph

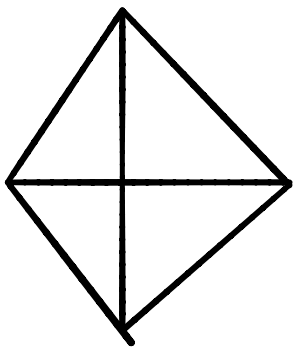
MST algorithms and their regularized versions

Reverse-delete (reducing graph)

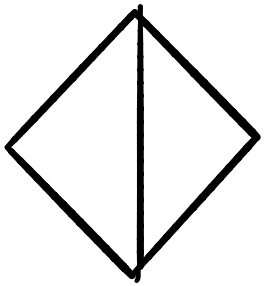


X'

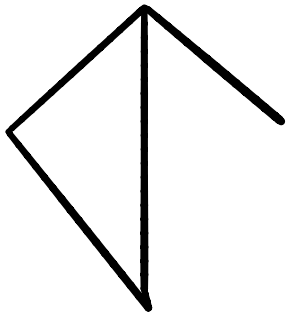
$t=0$



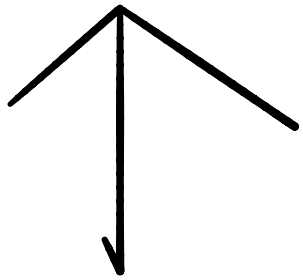
$t=1$



$t=2$



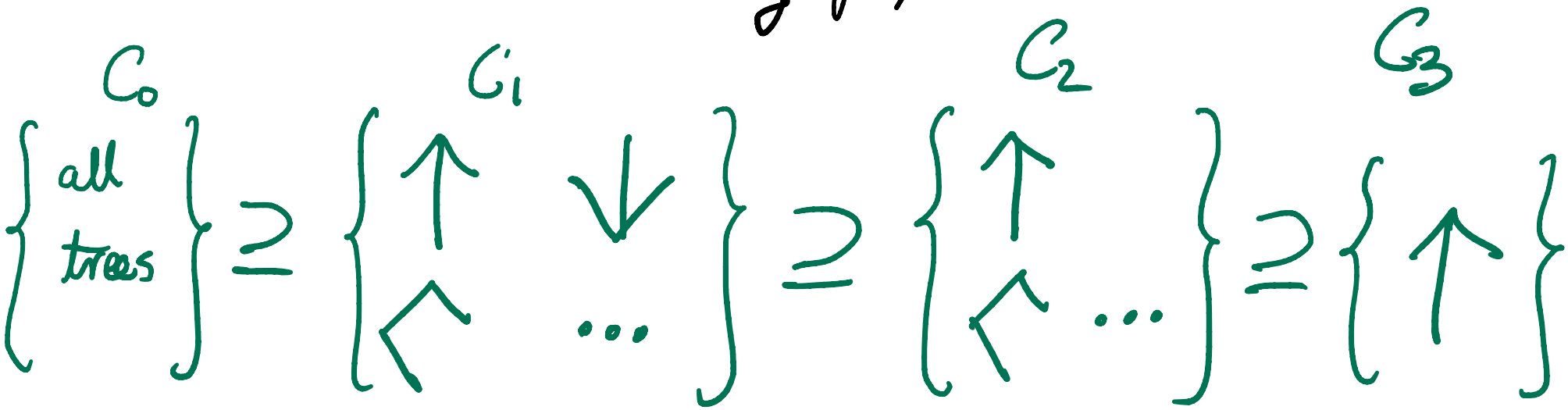
$t=3$



MST algorithms and their regularized versions

Reverse-delete (reducing graph)

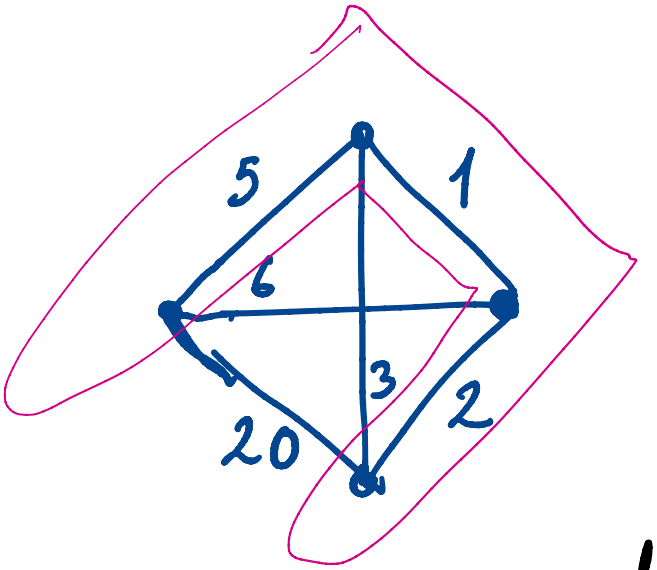
C_0 C_1 C_2 C_3



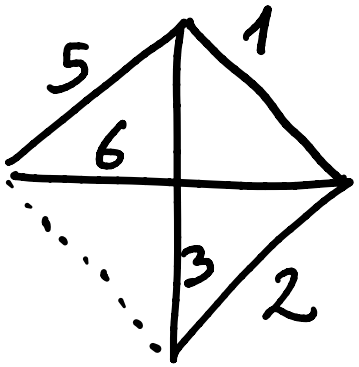
$t=0$ $t=1$ $t=2$ $t=3$

Reverse-delete

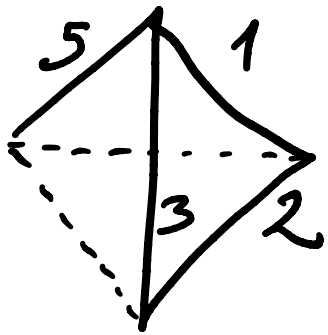
* Discards the heaviest edge, while keeping the graph connected
* Discards bad candidates early



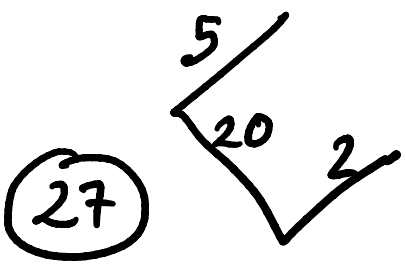
C_1

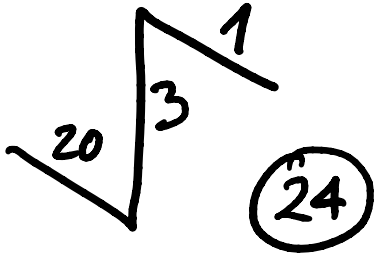


C_2



discards





Conclusion

* PA correctly ranked the performance of these algorithms.

* PA can guide the choice of hyperparameter values, cost functions, and algorithms.

Approximate sorting

L. Busse, M. Haghir Chehreghani, J. Buhmann

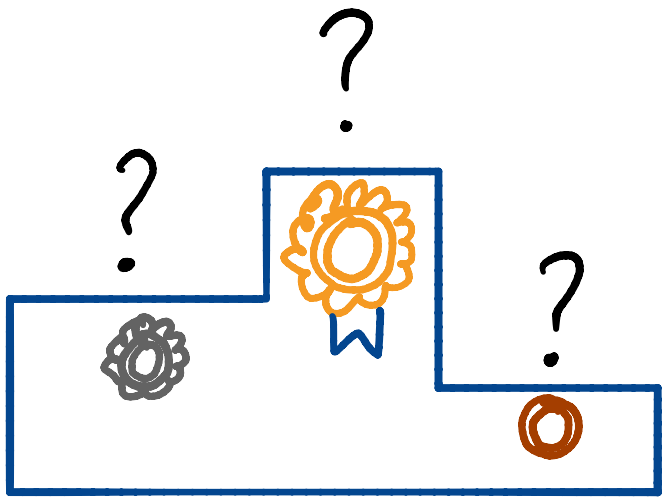
Problem intro

	A	B	C
Alice		W	W
Bob	L		W
Charlie	L	L	

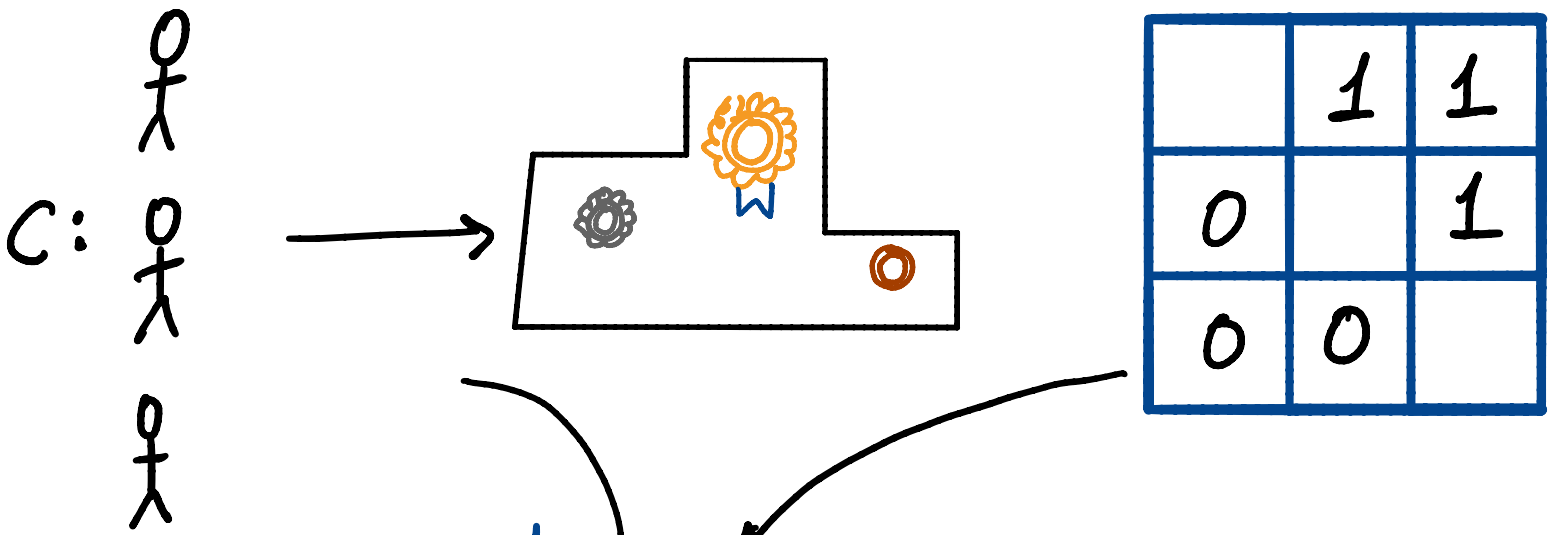
Fall 2019

	A	B	C
Alice		W	W
Bob	L		L
Charlie	L	W	

Spring 2020



Problem formalization



$$R^{sort}(c, X) = \sum_{i,j} X_{ij} \mathbb{1}\{c(i) > c(j)\}$$

$$X = (X_{ij})_{i,j} \in \{0,1\}^{n \times n}$$

$$X_{ij} \begin{cases} 1 & i \text{ is "larger than" } j. \\ 0 & \text{otherwise.} \end{cases}$$

$$c : \{1, \dots, n\} \longleftrightarrow \{1, \dots, n\}$$

ME+PA

$$p^{sort}(c|X) \propto \exp\left(-\frac{1}{T^*} R^{sort}(c, X)\right).$$

$$T^* \leftarrow \arg\max_T \; K_T(X', X'')$$

Problem: $p^{sort}(c|X)$'s normalization constant is intractable.
(n! different solutions)

Approx. sorting

$$R^{sort}(c, X) = \sum_{i,j} X_{ij} \mathbb{1}\{c(i) > c(j)\}.$$

$$R^{mf}(M, \mathcal{E}) = \sum_i \sum_k M_{ik} \mathcal{E}_{ik}.$$

$$R^{mfs}(M, \mathcal{E}) = \sum_i \sum_k M_{ik} \mathcal{E}_{ik} + \sum_k \mu_k \left(\sum_i M_{ik} - 1 \right)$$

$$= \sum_{i,k} M_{ik} \mathcal{E}_{ik} + \sum_{i,k} M_{ik} \mu_k - \sum_{i,k} \frac{\mu_k}{n}$$

$$= \text{const} + \sum_{i,k} M_{ik} (\mathcal{E}_{ik} + \mu_k)$$

What's the optimal T?

$$T^* = \arg\max_T K_T(X', X'')$$

$$\vdots$$

Tractable! ⇓

$$= 1 + \frac{1}{\log|C|}\sum_i \log\sum_k \exp\left(-\frac{1}{T}\left(\varepsilon'_{ik} + \mu'_k + \varepsilon''_{ik} + \mu''_k\right)\right)$$

$$- \frac{1}{\log|C|}\sum_i \log\left(\sum_k \exp\left(-\frac{1}{T}\left(\varepsilon'_{ik} + \mu'_k\right)\right)\sum_k \exp\left(-\frac{1}{T}\left(\varepsilon''_{ik} + \mu''_k\right)\right)\right).$$

Experiments

Synthetic experiments

A noisy ranking $\tilde{\pi}$ was generated

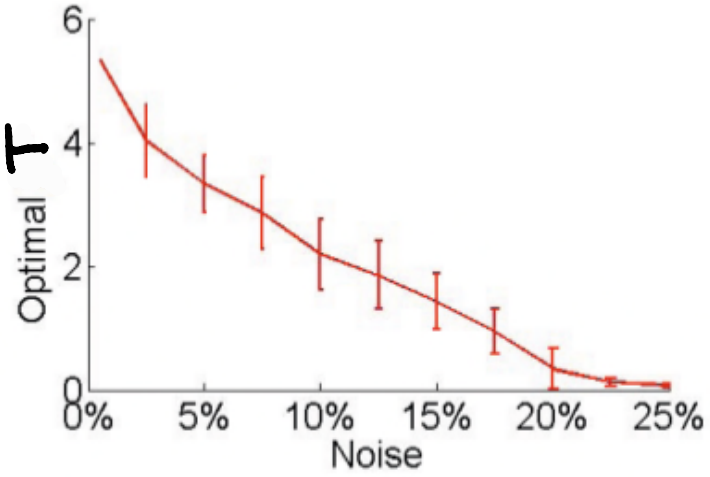
$$\tilde{\pi}_i \sim \mathcal{N}(i, \sigma^2)$$

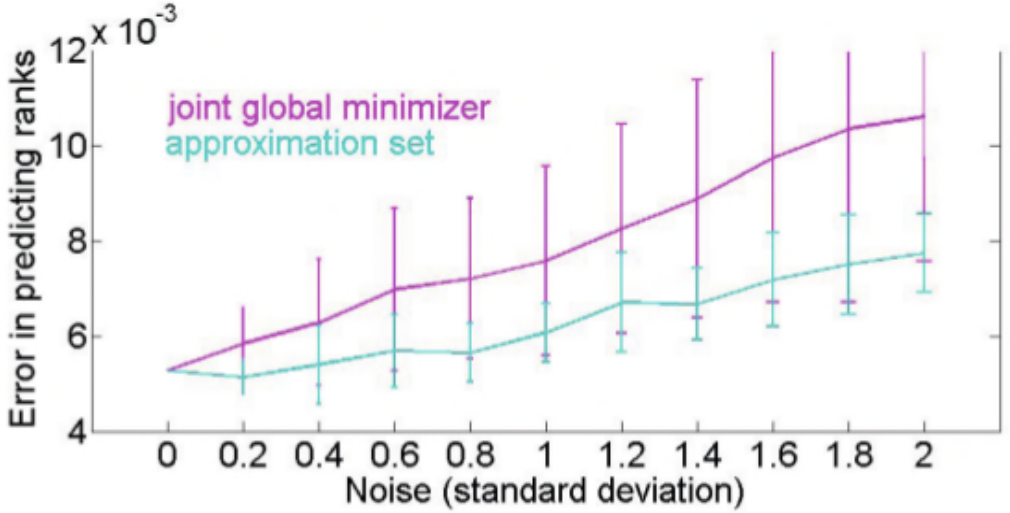
σ^2: noise parameter

Then X was defined by

$$X_{ij} := \mathbb{1}\{\tilde{\pi}_i < \tilde{\pi}_j\}.$$

Experimental results





Experiments with real-world data

Tournaments from the German chess league competitions ($n = 16$ teams)

records from wins and losses

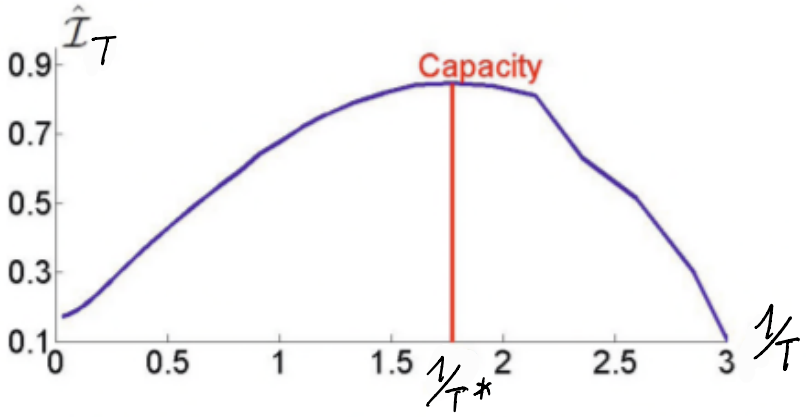
$X^{I} \leftarrow$ 2009-2010

$X^{II} \leftarrow$ 2010-2011

Approach: $c \overset{\$}{\leftarrow} \propto \langle p_{T*}(\cdot \mid X^{I}), p_{T*}(\cdot \mid X^{II}) \rangle$

Baseline: $\arg\min_{c} R^{sort}(c, X^{I}) + R^{sort}(c, X^{II})$

Results



(a) Sorting structure in chess data

Method	Prediction error per rank		
	abs.	rel.	gain
Baseline	0.29	14%	
ME + PA	0.14	6.8%	51%

(b) Prediction of chess league results

What we learned

* PA as a method for validating algorithms.
* Algorithms as communication channels.
* Asymptotic equipartition property and typicality as tools for simplifying analysis of complex systems.
* How to use information theory to develop effective algorithms for robust stochastic optimization.