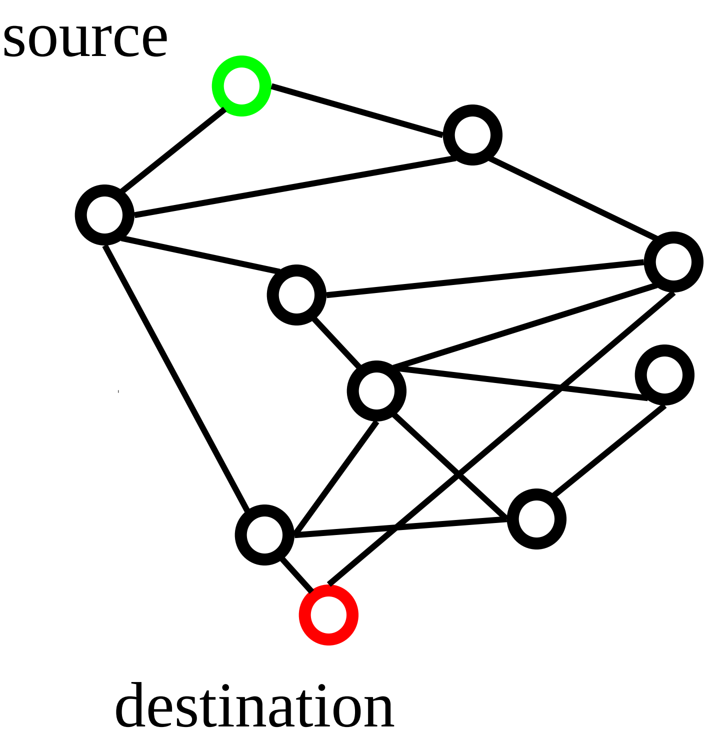
online convex optimization
(with partial information)

Elad Hazan @ IBM Almaden

Joint work with
Jacob Abernethy and Alexander Rakhlin
@ UC Berkeley

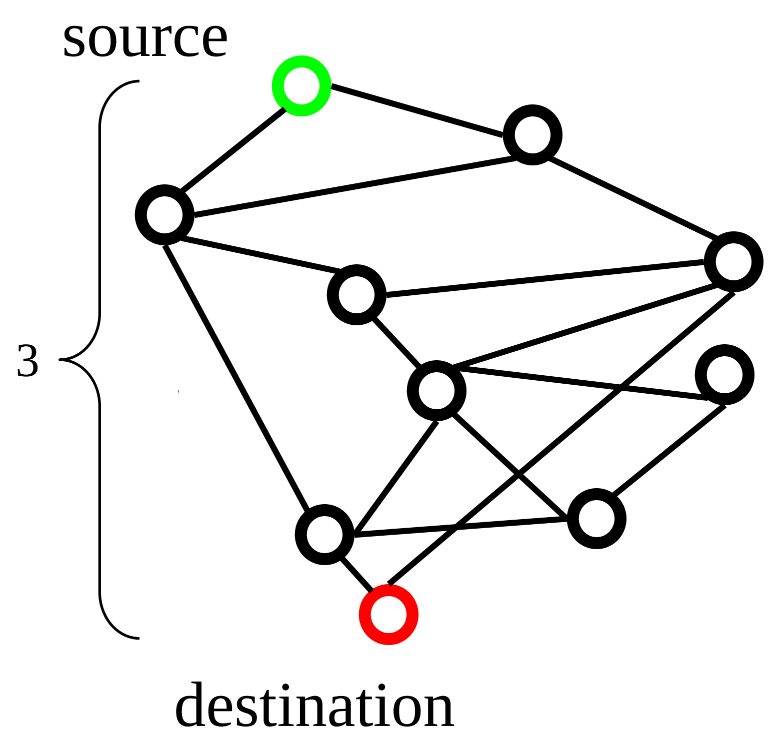
Online routing

Edge weights (congestion) – not known in advance.
Can change arbitrarily between 0-10 (say)



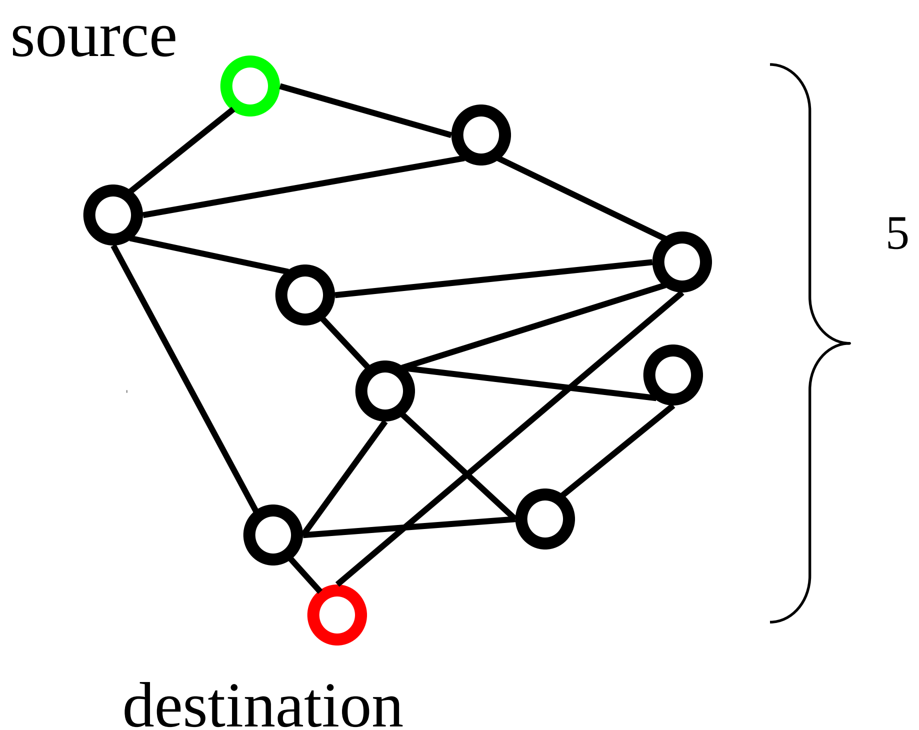
Online routing

Iteratively:
Pick path (not knowing network congestion),
then see length of path.



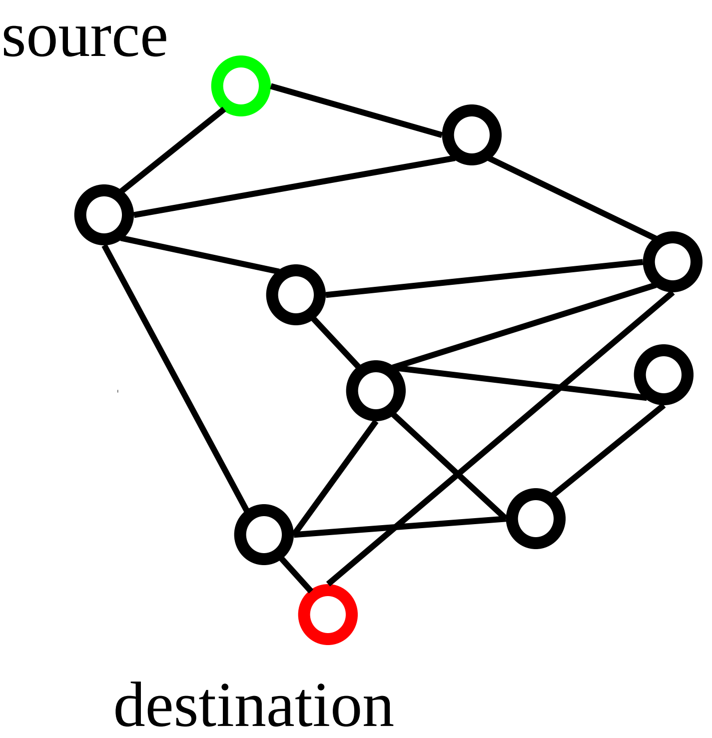
Online routing

Iteratively:
Pick path (not knowing network congestion), then see length of path.



Online routing

Iteratively:
Pick path (not knowing network congestion), then see length of path.



This talk: efficient and optimal algorithm for online routing

Why is the problem hard ?

- Three sources of difficulty:

1. Prediction problem – future unknown, unrestricted
2. Partial information – performance of other decisions unknown
3. Efficiency: exponentially large decision set

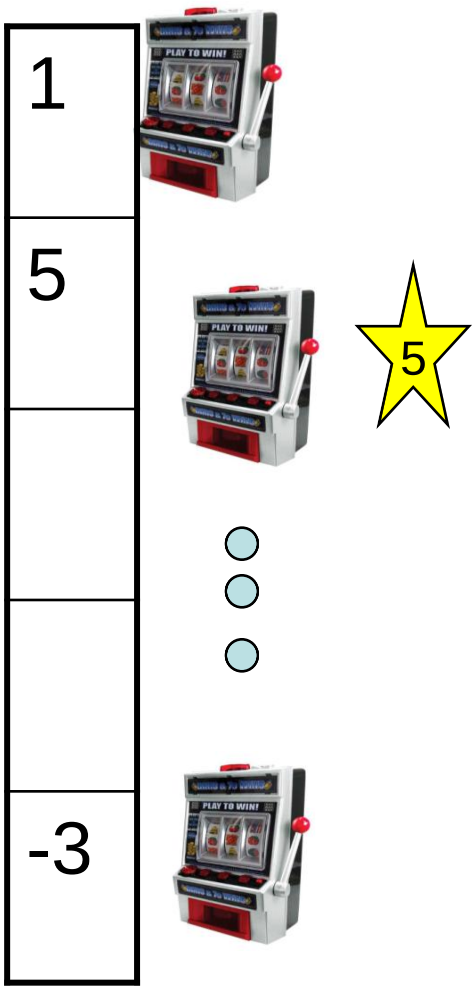
- Similar situations arise in

Production, web retail, Advertising, Online routing…

Technical Agenda

1. Describe classical general framework
2. Where classical approaches come short
3. Modern, Online convex optimization framework
4. Our new algorithm, and a few technical details

Multi-armed bandit problem



Iteratively, for time t=1,2,… :

1. Adversary chooses payoffs on the machines r_t
2. Player picks a machine i_t
3. Loss is revealed $r_t(i_t)$

Process is repeated T times ! (with different, unknown, arbitrary losses every round)

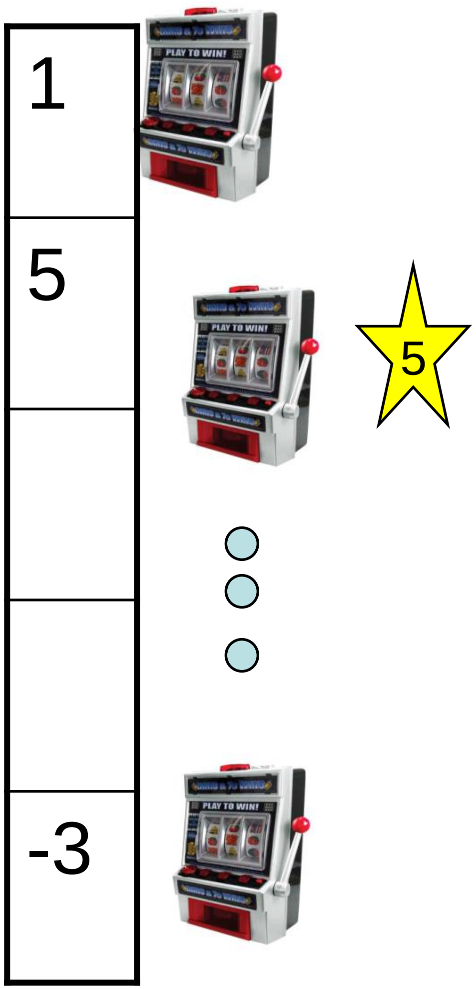
Goal:
Minimize the regret

Regret = cost(ALG) – cost(smallest-cost fixed machine)

$$= \sum_t r_t(i_t) - \min_j \sum_t r_t(j)$$

Online learning problem - design efficient algorithms with small regret

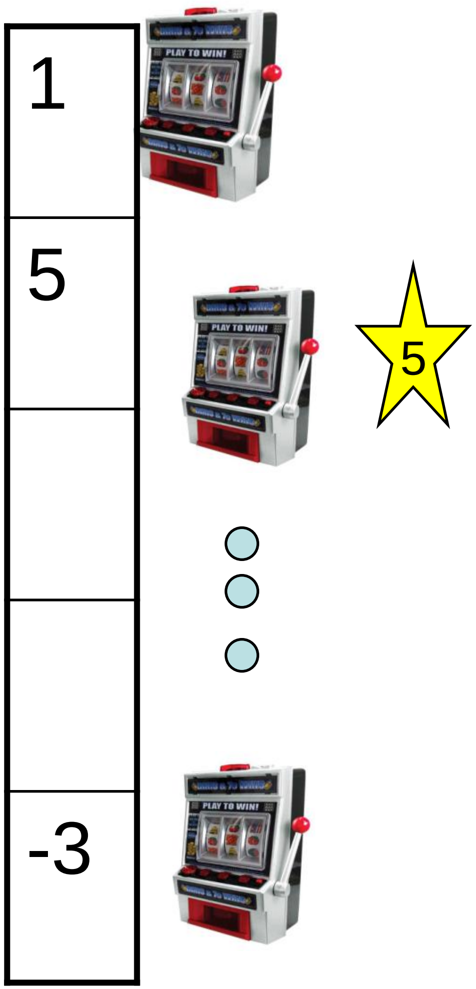
Multi-armed bandit problem



Studied in game theory, statistics (as early as 1950's), more recently machine learning.

- Robbins (1952) (statistical setting),
- Hannan (1957) – game theoretic setting, full information.
- [ACFS '98] – adversarial bandits:

Multi-armed bandit problem



[ACFS ’98] – adversarial bandits:

Regret = $O(\sqrt{T}\sqrt{K})$ for K machines, T game iterations.

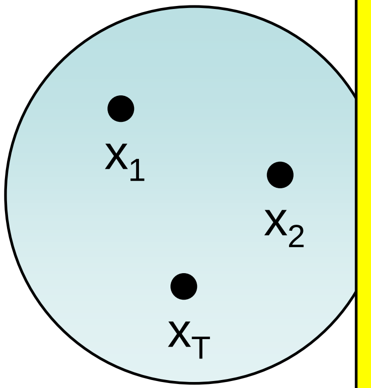
Optimal up to the constant.

What’s left to do ?

Exponential regret & run time for routing
(K = # of paths in the graph)

Our result: Efficient algorithm with $\sqrt{T}\, n$ Regret (n = # of edges in graph)

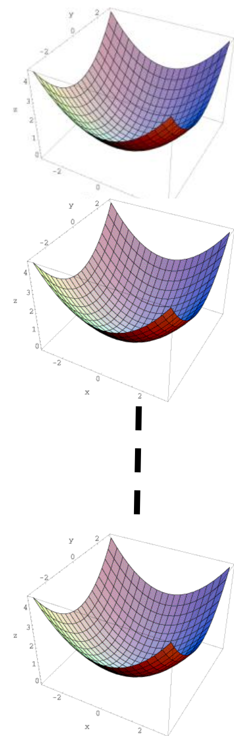
Online convex optimization



Distribution over paths = flow in the graph

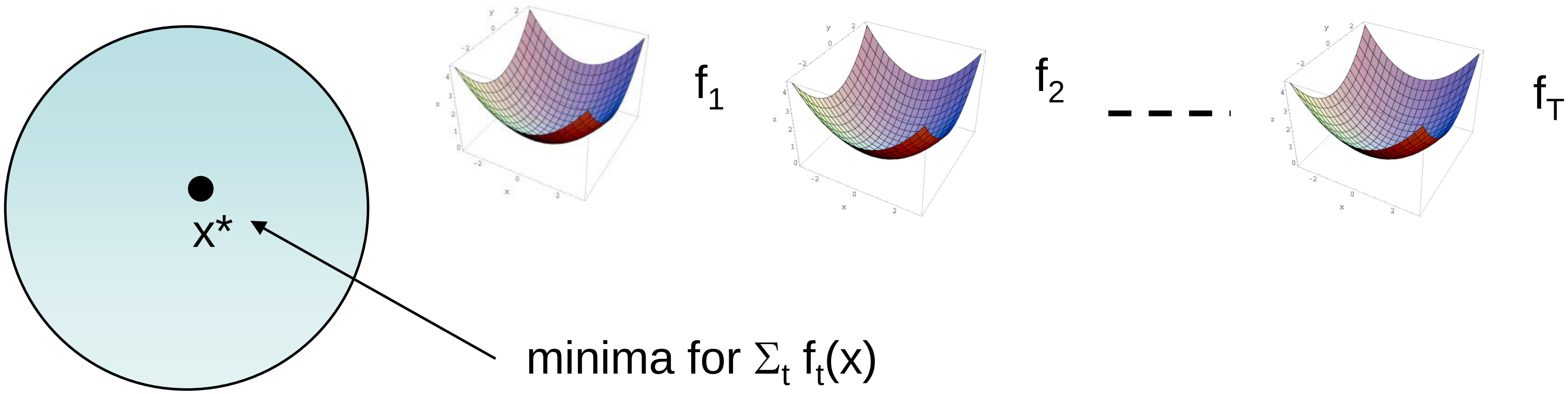
The set of all flows = $P \subseteq R^n$ = polytope of flows

This polytope has a compact representation, as the number of facets = number of constraints for flows in graphs is small.
(The flow constraints, flow conservation and edge capacity)



- Convex bounded functions (for this talk, linear functions)
  - Total loss = $\sum_t f_t(x_t)$
- Can express online routing as OCO over polytope in R^n with O(n) constraints (despite exp' large decision set)

Online performance metric - regret



minima for $\Sigma_t f_t(x)$
(fixed optimum in hindsight)

- Loss of our algorithm = $\Sigma_t f_t(x_t)$
- Regret = Loss(ALG) – Loss(OPT point) = $\Sigma_t f_t(x_t) - \Sigma_t f_t(x^*)$

Existing techniques

- Adaboost, Online Gradient Descent, Weighted Majority, Online Newton Step, Perceptron, Winnow…
- All can be seen as special case of "Follow the regularized leader"
  - At time t predict:

$$x_t = \arg\min_{x \in K} \left\{ \sum_{\tau=1}^{t-1} f_\tau(x) + R(x) \right\}$$

Requires complete knowledge of cost functions

Our algorithm - template

Two generic parts:

- Compute the "Regularized leader" $x_t = \arg\min_{x \in K} \left\{ \sum_{\tau=1}^{t-1} g_\tau(x) + R(x) \right\}$

- In order to make the above work, estimate functions g_t such that $E[g_t] = f_t$.

Note: we would prefer to use $x_t = \arg\min_{x \in K} \left\{ \sum_{\tau=1}^{t-1} f_\tau(x) + R(x) \right\}$

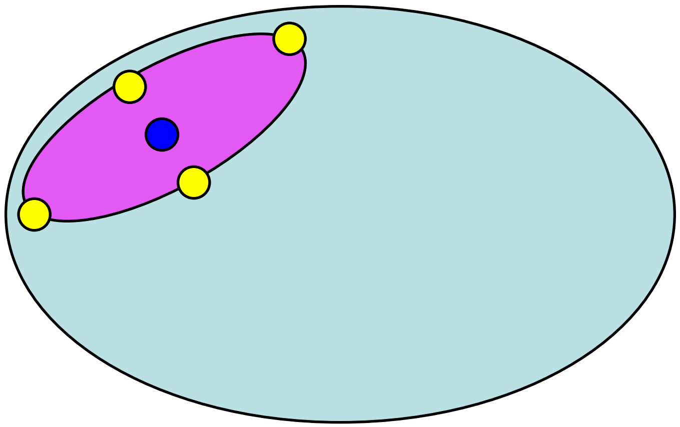
But we do not have access to f_t !

Our algorithm - template

1. Compute the current prediction x_t based on previous observations.

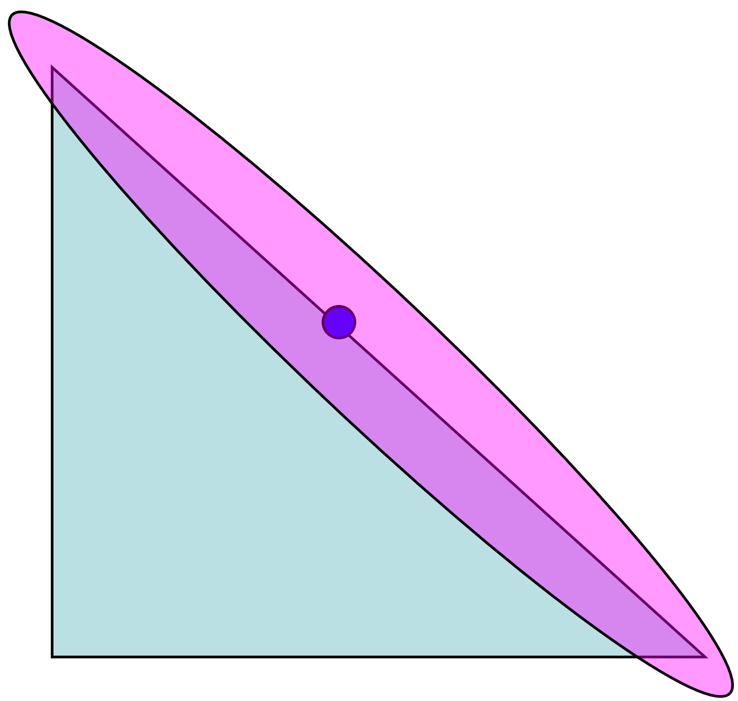
$$x_t = \arg\min_{x \in K} \left\{ \sum_{\tau=1}^{t-1} g_\tau(x) + R(x) \right\}$$

2. Play random variable y_t, taken from a distribution such that
   1. Distribution is centered at prediction: $E[y_t] = x_t$
   2. From the information $f_t(y_t)$, we can create a random variable g_t, such that
      $E[g_t] = f_t$
   3. The variance of the random variable g_t needs to be small



Simple example: interpolating the cost function

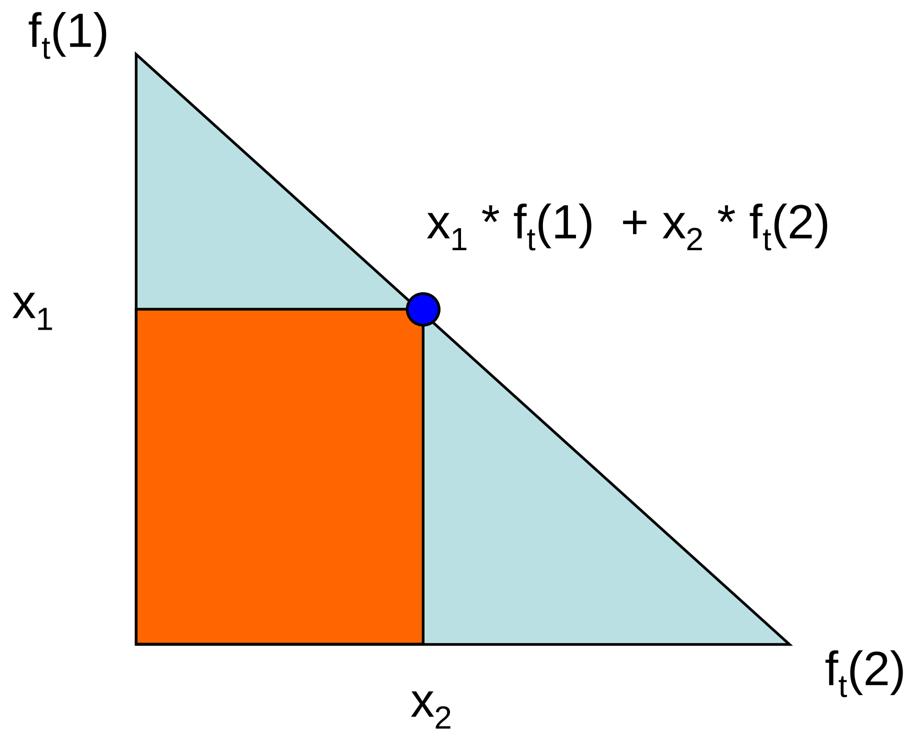
- Want to compute & use:
  what is f_t ? (we only see $f_t(x_t)$)



Convex set =
2-dim simplex
(dist on two
endpoints)

Simple example: interpolating the cost function

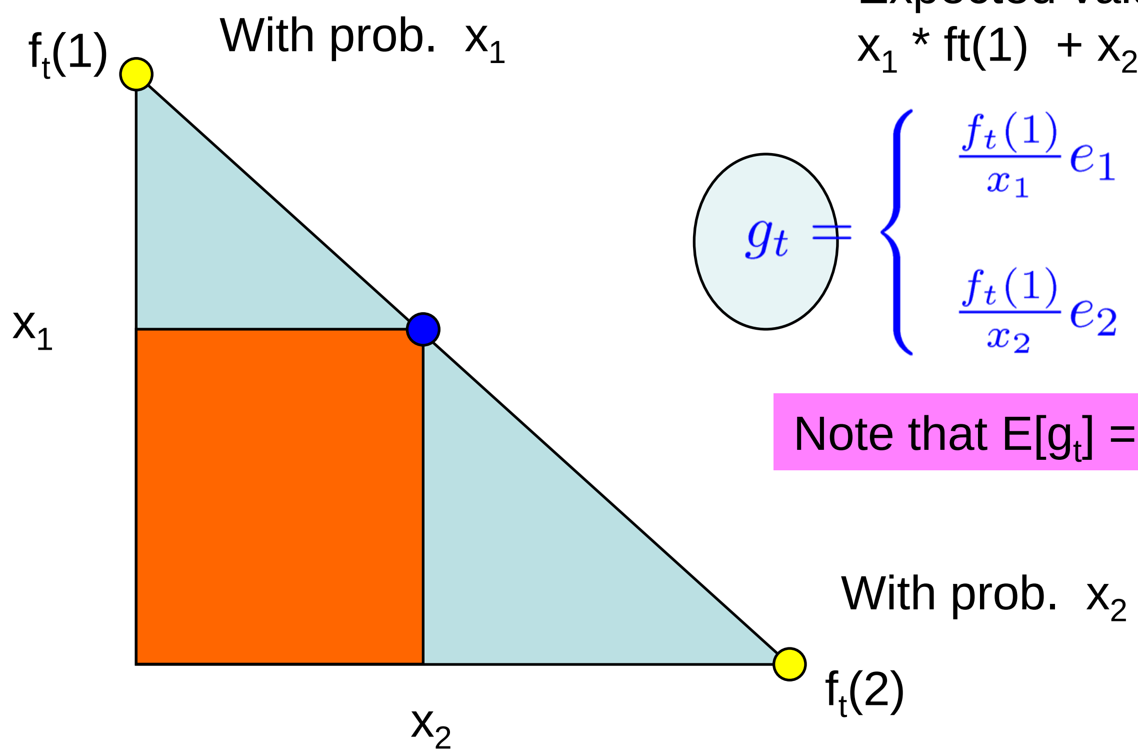
- Want to compute & use:
  what is f_t ? (we only see $f_t(x_t)$)



Convex set =
2-dim simplex
(dist on two
endpoints)

Simple example: interpolating the cost function

Unbiased estimate of f_t is just as good



Expected value remains
x_1 * ft(1) + x_2 * $f_t(2)$

$$g_t = \begin{cases} \frac{f_t(1)}{x_1} e_1 & x_t = e_1 \\ \frac{f_t(1)}{x_2} e_2 & x_t = e_2 \end{cases}$$

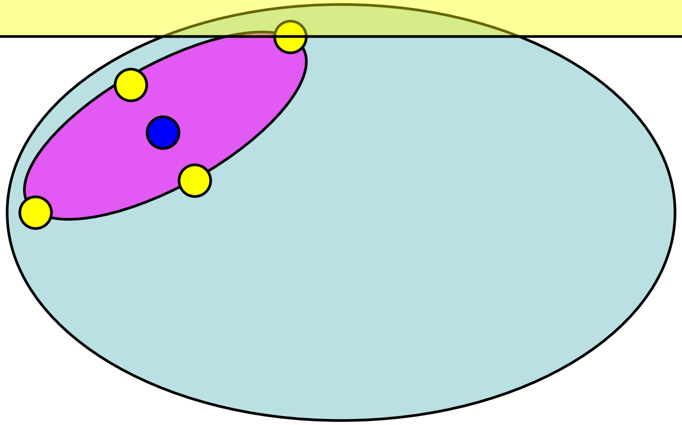
Note that $E[g_t] = f_t$

The algorithm – complete definition

- Let R(x) be a self concordant barrier for the convex set
- Compute current prediction center:

$$x_t = \arg\min_{x \in K} \left\{ \sum_{\tau=1}^{t-1} g_\tau(x) + R(x) \right\}$$

- Compute eigenvectors of Hessian at current point. Consider the intersection of eigenvectors and the Dikin ellipsoid
- Sample uniformly from eigendirections, to obtain unbiased estimates $E[g_t] = f_t$



Remember that we need the random variables y_t, g_t to satisfy:

1. $E[y_t] = x_t$
2. $E[g_t] = f_t$
3. The variance of the random variable g_t needs to be small

This is where self-concordance is crucial

An extremely short survey: IP methods

- Probably most widely used algorithms today
  "IP polynomial algorithms in convex programming" [NN '94]
- To solve a general optimization problem: minimize convex function over a convex set (specified by constraints), reduce to unconstrained optimization via a self concordant barrier function

$$
\begin{aligned}
&\min f(x) \\
&A_1 \cdot x - b_1 \le 0 \\
&\ldots \\
&A_m \cdot x - b_m \le 0 \\
&x \in R^n
\end{aligned}
$$

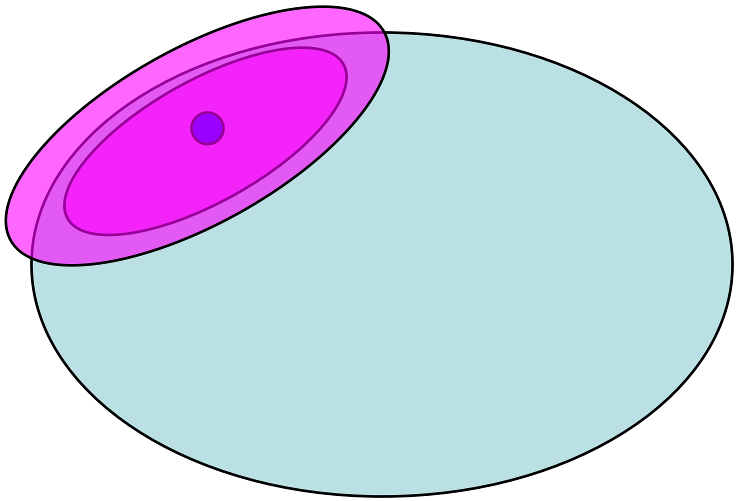
$$
\begin{aligned}
&\min f(x) - \alpha \times \underbrace{\textstyle\sum_i \log(b_i - A_i x)}_{R(x)} \\
&x \in R^n
\end{aligned}
$$

Barrier function

The geometric properties of self concordant functions

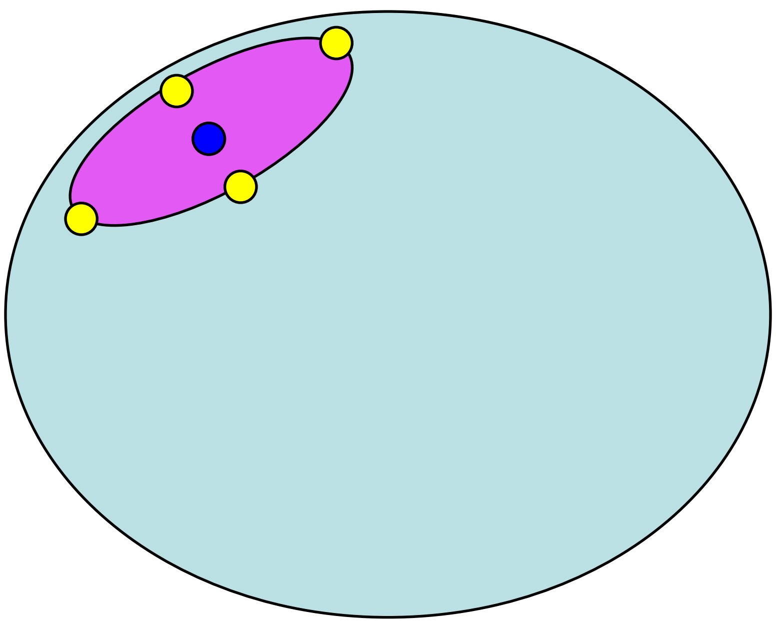
- Let R be self concordant for convex set K, then at each point x, the hessian of R at x defines a local norm.
- The Dikin ellipsoid $$D_1(x) = \{y \text{ such that } \|y - x\|_x \leq 1\}$$
- Fact: for all x in the set, $D_R(x) \subseteq K$
- The Dikin ellipsoid characterizes "space to the boundary" in every direction
- It is tight:

$$D_2(x) \not\subseteq K$$

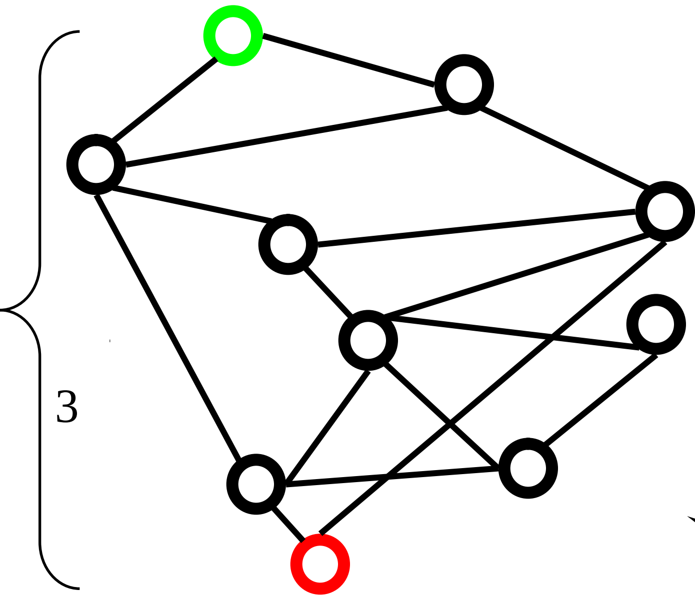


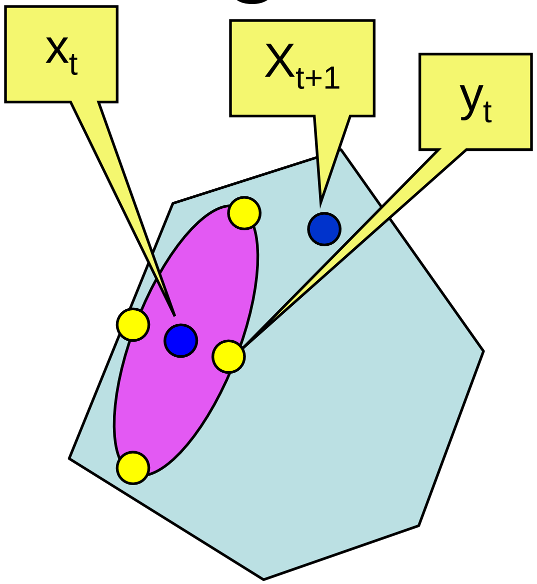
Unbiased estimators from self-concordant functions

- Eigendirections of Hessian: create unbiased gradient estimator in each direction
- Orthogonal basis – complete estimator
- Self concordance is important for:
  - Capture the geometry of the set (Dikin ellipsoid)
  - Control the variance of the estimator



Online routing – the final algorithm





Flow decomposition for y_t
Obtain path p_t

Guess path,
Observe length

Summary

1. Online optimization algorithm with limited feedback
   - Optimal regret – $O(\sqrt{T}\, n)$
   - Efficient (n^3 time per iteration)
   - More efficient implementation based on IP theory

Open Questions:

- Dependence on dimension is not known to be optimal
- Algorithm has large variance – $T^{2/3}$, reduce the variance to $\sqrt{T}$?
- Adaptive adversaries

The End