Everything I ever needed to know about life I learned by making crepes
Valid Crepe Recipes

• **Convex hull**: Finds a convex space that contains all recipe data points

• Connections to many other fundamental geometric problems (Voronoi D. & Delaunay Tr.)

• Can reduce to $O(1/\varepsilon)$ points in the hull if can tolerate distance errors of $\varepsilon$

• Can compute dynamically; $O(n \log n)$ time to add $n$ points
Neighbors & Clusters

• Each known recipe is a point in $\mathbb{R}^n$.

• **Voronoi Diagram:** Enables quickly finding nearest neighbor in old recipes of a new recipe.

• **Delaunay Triangulation:** Enables clustering of crepe recipes. Clustering = maximize minimum distance between clusters.
Duality

- Convex Hull and Voronoi Diagram/Delaunay Triangulation problems are connected via **duality**

- **Duality** is a transformation between points and lines

- Duality can also help solve other problems quickly: **finding a linear classifier**; finding a line that passes through 3 points, etc.
Crepes on the Cheap

• **Linear Programming:** Finds a “valid” (i.e. in the feasible convex space) crepe recipe with minimum cost

• For n constraints and O(1) variables, can solve LP in O(n) expected time.

• Can solve general LP to within $\varepsilon$ factor using MWU. Works even when constraints are concave instead of linear!
Learning crepes

- Crepe recipes are either good or bad
- **Winnow**: Learns a “perfect” linear classifier
- **SVM**: Learns a linear classifier with some misclassifications
- **Adaboost**: Non-linear learning via ensembles of “weak” linear classifiers

Winnow and Adaboost use MWU; SVM uses gradient descent
Rock, Scissors, Crepe

- Assume two competitors (on Iron chef?) can prepare one of $x$ different recipes; and there is a known zero-sum payoff matrix

- MWU and **fictitious play** will converge to the Nash equilibrium for this game
Gradient Descent
Offline Crepes

• Convex search space of valid crepe recipes

• **One** convex function to minimize (or concave function to maximize)

• Gradient descent converges to the minimum

• Convergence time depends on diameter of search space and max norm of gradient.
Online Convex Optimization

- Say that each day, you prepare a recipe for a new friend
- Convex functions are: - crepe rating
- In day i, there is a new convex function $f_i$
Online Crepes

- Convex search space of valid crepe recipes

- Many convex functions to minimize, one in each round

- Online gradient descent finds points with cost “close” to the best single offline point

- Regret (our cost - best offline cost) depends on diameter of search space and max norm of gradient
Stochastic Crepes

• Say that each day, you prepare a recipe for a new crowd

• Convex functions are: - average crepe rating

• Suffices to sample gradient just based on the rating of a single person in the crowd; Improves efficiency of gradient descent; This is called stochastic gradient descent
Rounding Crepes

• Say your convex space is given by an LP that assigns \textit{probabilities} to certain variables

• Each day, there is a convex cost as a function of these probabilities

• Online gradient descent can minimize expected regret of the randomized rounding of these variables

• Examples: shortest paths, set cover, SAT, etc.
Projections onto low dimensional subspaces
Compressing Crepes

- Goal: Project recipes from high (n) dimensional space to low

- Johnson-Lindenstrauss
  - Preserves pair-wise distances (and angles) of polynomial points up to ε multiplicative error; $O(\log \frac{n}{\varepsilon^2})$ dimensions
  - Can compute online

- Singular Value Decomposition:
  - Minimizes average distance with original points
  - Must compute offline
Compressing Crepes

- **Johnson-Lindenstrauss**: 1) Learning; 2) Reducing state (data streaming)

- **Singular Value Decomposition**
  - Data: Compression; Reducing noise
  - Functions: Best linear fit (smallest eigenvector)
  - Graphs: Finding “dense” subgraphs (between recipes and items?)
Questions?

RICKY BOBBY JUST SAY...

THAT YOU LOVE CREPES