

Mathematical Optimization Models and Applications

Yinyu Ye

Department of Management Science and Engineering

Stanford University

Stanford, CA 94305, U.S.A.

<http://www.stanford.edu/~yyye>

Chapters 1, 2.1-2, 6.1-2, 7.2, 11.3, 11.6

What you learn in CME307/MS&E311?

- Present a core element, **mathematical optimization theories and algorithms**, for the ICME/MS&E disciplines.
- Provide **mathematical proofs and in-depth theoretical analyses** of optimization/game models/algorithms discussed in MS&E211
- Introduce additional **conic and nonlinear/nonconvex optimization/game** models/problems comparing to MS&E310.
- Describe new/recent effective optimization/game **models/methods/algorithms** in Data Science, Machine Learning and AI.
- Emphasis is on nonlinear, nonconvex and stochastic/sample-based **optimization theories and practices** together with convex analyses.

Mathematical Optimization

The field of optimization is concerned with the study of **maximization and minimization of mathematical functions**. Very often the arguments of (i.e., **variables** or **unknowns**) in these functions are subject to side conditions or **constraints**. By virtue of its great utility in such diverse areas as applied science, engineering, economics, finance, medicine, and statistics, optimization holds an important place in the practical world and the scientific world. Indeed, as far back as the Eighteenth Century, the famous Swiss mathematician and physicist Leonhard Euler (1707-1783) proclaimed^a that . . . **nothing at all takes place in the Universe in which some rule of maximum or minimum does not appear.**

^aSee Leonhardo Eulero, *Methodus Inviendi Lineas Curvas Maximi Minimive Proprietate Gaudentes*, Lausanne & Geneva, 1744, p. 245.

Mathematical Optimization/Programming (MP)

The class of mathematical optimization/programming problems considered in this course can all be expressed in the form

$$\begin{aligned} (\text{P}) \quad & \text{minimize} \quad f(\mathbf{x}) \\ & \text{subject to} \quad \mathbf{x} \in \mathcal{X} \end{aligned}$$

where \mathcal{X} usually specified by constraints:

$$\begin{aligned} c_i(\mathbf{x}) &= 0 \quad i \in \mathcal{E} \\ c_i(\mathbf{x}) &\leq 0 \quad i \in \mathcal{I}. \end{aligned}$$

If the constraint functions are linear/affine type, then \mathcal{X} is a convex polyhedral set/region.

Model Classifications

Optimization problems are generally divided into Unconstrained, Linear and Nonlinear Programming based upon the objective and constraints of the problem

- **Unconstrained Optimization:** Ω is the entire space R^n
- **Linear Optimization:** If both the objective and the constraint functions are linear/affine
- **Nonlinear Optimization:** If the objective/constraints contain general nonlinear functions
- **Convex Optimization:** If the objective is a convex function and the constraint region is a convex set
- **Conic Linear Optimization:** If both the objective and the constraint functions are linear/affine, and variables are in a convex cone.
- **(Mixed) Integer Optimization:** If some variables are restricted to be integral
- **Stochastic Optimization:** Optimize the expected objective function with random parameters
- **Fixed-Point or Min-Max Optimization:** Optimization of multiple agents with zero-sum objectives

We present a few optimization examples in this lecture that we would cover throughout this course.

Structured Optimization: Conic Linear Programming (CLP)

$$\begin{aligned} & \text{minimize} && \mathbf{c}^T \mathbf{x} \\ & \text{subject to} && A\mathbf{x} = \mathbf{b}, \\ & && \mathbf{x} \in K. \end{aligned}$$

Linear Programming (LP): when K is the nonnegative orthant cone

Second-Order Cone Programming (SOCP): when K is the second-order cone

Semidefinite Cone Programming (SDP): when K is the semidefinite matrix cone

$$\begin{array}{ll} & \min_{x_1, x_2, x_3} \\ & 2x_1 + x_2 + x_3 \\ \text{s.t.} & x_1 + x_2 + x_3 = 1, \\ & (x_1; x_2; x_3) \geq \mathbf{0}; \end{array}$$

$$\begin{array}{ll} & \min_{x_1, x_2, x_3} \\ & 2x_1 + x_2 + x_3 \\ \text{s.t.} & x_1 + x_2 + x_3 = 1, \\ & \sqrt{x_2^2 + x_3^2} \leq x_1. \end{array}$$

$$\begin{array}{ll} & \min_{x_1, x_2, x_3} \\ & 2x_1 + x_2 + x_3 \\ \text{s.t.} & x_1 + x_2 + x_3 = 1, \\ & \begin{pmatrix} x_1 & x_2 \\ x_2 & x_3 \end{pmatrix} \succeq \mathbf{0}, \end{array}$$

Facility Location Problem

Let \mathbf{c}_j be the location of client $j = 1, 2, \dots, m$, and \mathbf{y} be the location decision of a facility to be built. Then we solve

$$\underset{\mathbf{y}}{\text{minimize}} \quad \sum_j \|\mathbf{y} - \mathbf{c}_j\|_p.$$

Or equivalently (?)

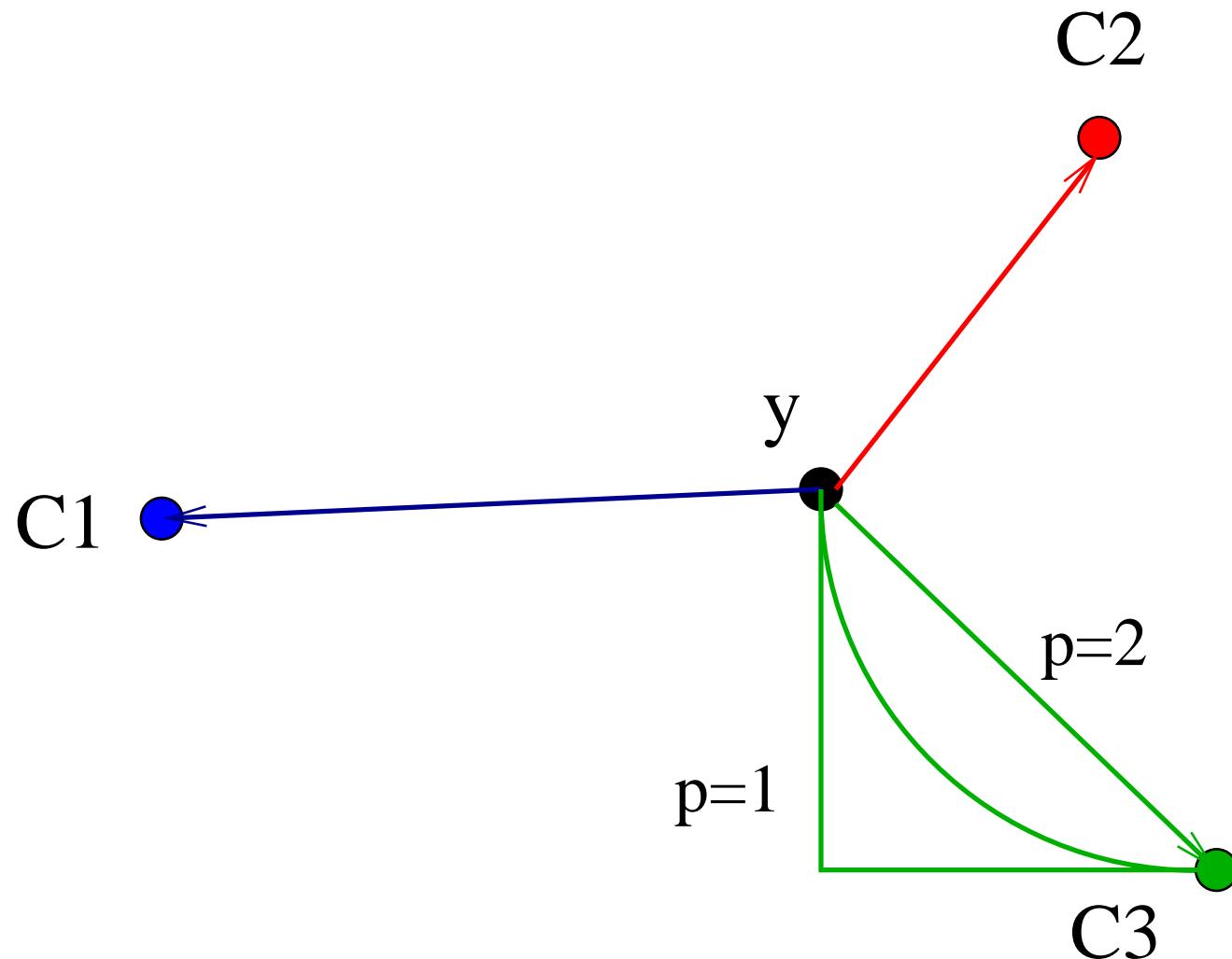
$$\begin{aligned} & \underset{\mathbf{y}}{\text{minimize}} \quad \sum_j \delta_j \\ & \text{subject to} \quad \mathbf{y} + \mathbf{x}_j = \mathbf{c}_j, \quad \|\mathbf{x}_j\|_p \leq \delta_j, \quad \forall j. \end{aligned}$$

This is a p -order conic linear program (POCP) for $p \geq 1$.

In particular, when $p = 2$, it is an SOCP problem.

why we need the p norm, because sometimes the signal's distance is between 1 norm and 2 norm (such as 1.5 norm)

For simplicity, consider $m = 3$.

Figure 1: Facility Location at Point y .

Sparse Linear Regression Problems

We want to find a sparsest solution to fit exact data measurements, that is, to minimize the number of non-zero entries in \mathbf{x} such that $A\mathbf{x} = \mathbf{b}$:

$$\begin{aligned} &\text{minimize} && \|\mathbf{x}\|_0 = |\{j : x_j \neq 0\}| \\ &\text{subject to} && A\mathbf{x} = \mathbf{b}. \end{aligned}$$

Sometimes this objective can be accomplished by LASSO:

$$\begin{aligned} &\text{minimize} && \|\mathbf{x}\|_1 = \sum_{j=1}^n |x_j| \\ &\text{subject to} && A\mathbf{x} = \mathbf{b}. \end{aligned}$$

It can be equivalently represented by (?)

$$\begin{aligned} &\text{minimize} && \sum_{j=1}^n y_j \\ &\text{subject to} && A\mathbf{x} = \mathbf{b}, -\mathbf{y} \leq \mathbf{x} \leq \mathbf{y}; \end{aligned}$$

or

$$\begin{aligned} &\text{minimize} && \sum_{j=1}^n (x'_j + x''_j) \\ &\text{subject to} && A(\mathbf{x}' - \mathbf{x}'') = \mathbf{b}, \mathbf{x}' \geq \mathbf{0}, \mathbf{x}'' \geq \mathbf{0}. \end{aligned}$$

Both are linear programs!

Sparsest Data Fitting continued

A better approximation of the objective can be accomplished by

$$\begin{array}{ll} \text{minimize} & \| \mathbf{x} \|_p := \left(\sum_{j=1}^n |x_j|^p \right)^{1/p} \\ \text{subject to} & A\mathbf{x} = \mathbf{b}; \end{array} \quad \text{or} \quad \text{minimize} \quad \|A\mathbf{x} - \mathbf{b}\|^2 + \mu \left(\sum_{j=1}^n |x_j|^p \right)^{1/p}$$

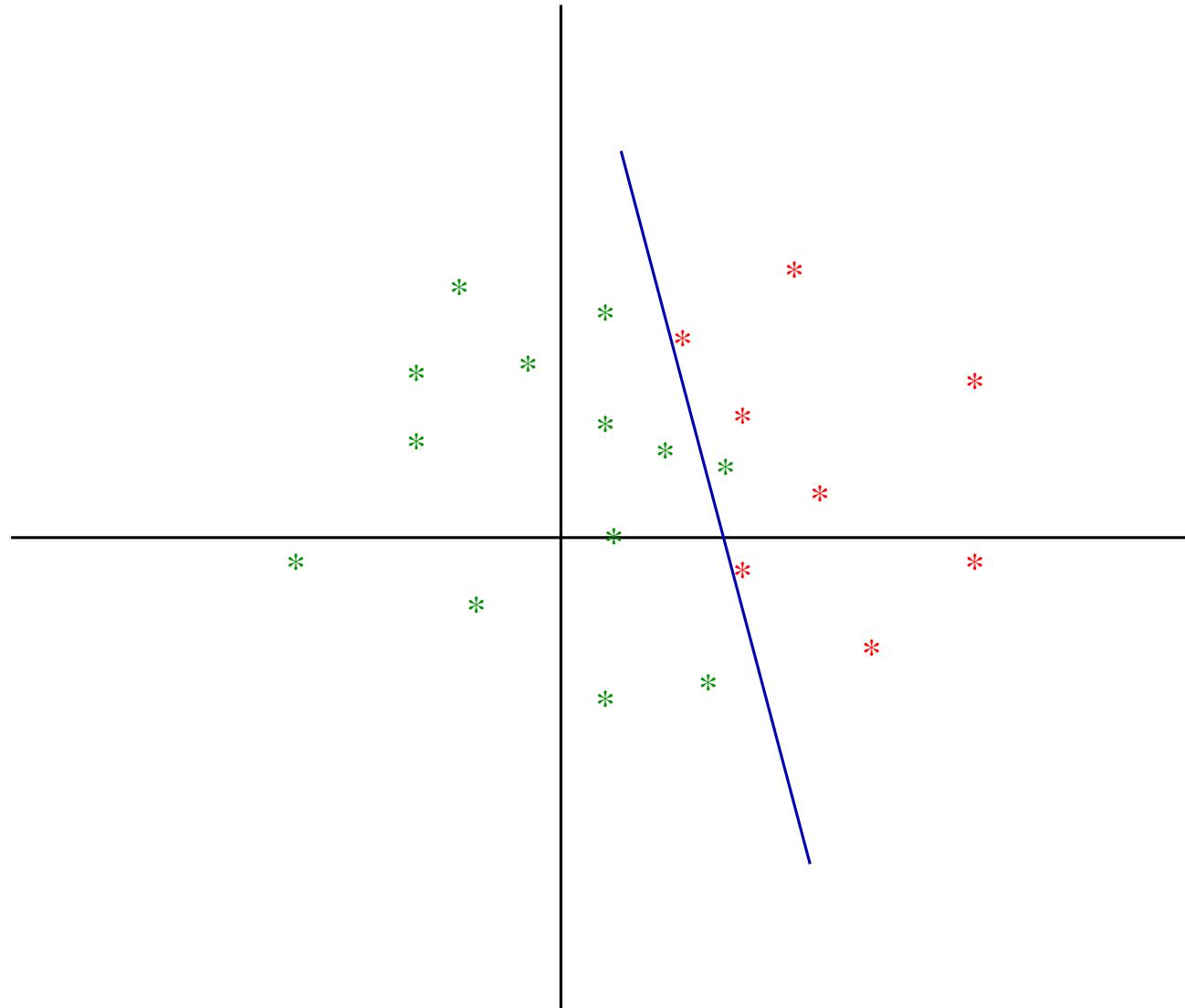
for some $0 < p < 1$, where $\mu > 0$ is a regularization parameter.

Or simply

$$\begin{array}{ll} \text{minimize} & \| \mathbf{x} \|_p^p := \left(\sum_{j=1}^n |x_j|^p \right) \\ \text{subject to} & A\mathbf{x} = \mathbf{b}; \end{array} \quad \text{or} \quad \text{minimize} \quad \|A\mathbf{x} - \mathbf{b}\|^2 + \beta \left(\sum_{j=1}^n |x_j|^p \right);$$

where the former is a **linearly constrained (nonconvex) optimization problem** and the latter is an **unconstrained (nonconvex) optimization problem**

Linear Classifier: Logistic Regression and Support Vector Machine



Data Classification: Supporting Vector Machine I

A powerful **binary-classification method** is the **Supporting Vector Machine (SVM)**.

Let the first class, say in **Red**, data points i be denoted by $\mathbf{a}_i \in R^d$, $i = 1, \dots, n_1$ and the second class data points j be denoted by $\mathbf{b}_j \in R^d$, $j = 1, \dots, n_2$. We like to find a hyperplane, slope vector \mathbf{x} and intersect scalar x_0 , to separate the two data classes:

$$\begin{aligned} \text{subject to } & \mathbf{a}_i^T \mathbf{x} + x_0 \geq 1, \quad \forall i, \\ & \mathbf{b}_j^T \mathbf{x} + x_0 \leq -1, \quad \forall j. \end{aligned}$$

This is a **linear program** with the null objective!

Data Classification: Supporting Vector Machine II

If strict separation is impossible, we then minimize error variable β

$$\begin{aligned} & \text{minimize} && \beta \\ & \text{subject to} && \mathbf{a}_i^T \mathbf{x} + x_0 + \beta \geq 1, \quad \forall i, \\ & && \mathbf{b}_j^T \mathbf{x} + x_0 - \beta \leq -1, \quad \forall j, \\ & && \beta \geq 0. \end{aligned}$$

Frequently we add the regularization term on the slope vector

$$\begin{aligned} & \text{minimize} && \beta + \mu \|\mathbf{x}\|^2 \\ & \text{subject to} && \mathbf{a}_i^T \mathbf{x} + x_0 + \beta \geq 1, \quad \forall i, \\ & && \mathbf{b}_j^T \mathbf{x} + x_0 - \beta \leq -1, \quad \forall j, \\ & && \beta \geq 0, \end{aligned}$$

where μ is a fixed positive regularization parameter.

This becomes a constrained quadratic program (QP). If $\mu = 0$, then it is a linear program (LP)!

Supporting Vector Machine: Ellipsoidal Separation?

$$\begin{aligned}
 & \text{minimize} && \text{trace}(X) + \|\mathbf{x}\|^2 \\
 & \text{subject to} && \mathbf{a}_i^T X \mathbf{a}_i + \mathbf{a}_i^T \mathbf{x} + x_0 \geq 1, \quad \forall i, \\
 & && \mathbf{b}_j^T X \mathbf{b}_j + \mathbf{b}_j^T \mathbf{x} + x_0 \leq -1, \quad \forall j, \\
 & && X \succeq 0.
 \end{aligned}$$

This type of problems is semidefinite programming (SDP). When the problem is not separable:

$$\begin{aligned}
 & \text{minimize} && \beta + \mu(\text{trace}(X) + \|\mathbf{x}\|^2) \\
 & \text{subject to} && \mathbf{a}_i^T X \mathbf{a}_i + \mathbf{a}_i^T \mathbf{x} + x_0 + \beta \geq 1, \quad \forall i, \\
 & && \mathbf{b}_j^T X \mathbf{b}_j + \mathbf{b}_j^T \mathbf{x} + x_0 - \beta \leq -1, \quad \forall j, \\
 & && \beta \geq 0, \\
 & && X \succeq 0.
 \end{aligned}$$

This is a mixed linear and SDP program.

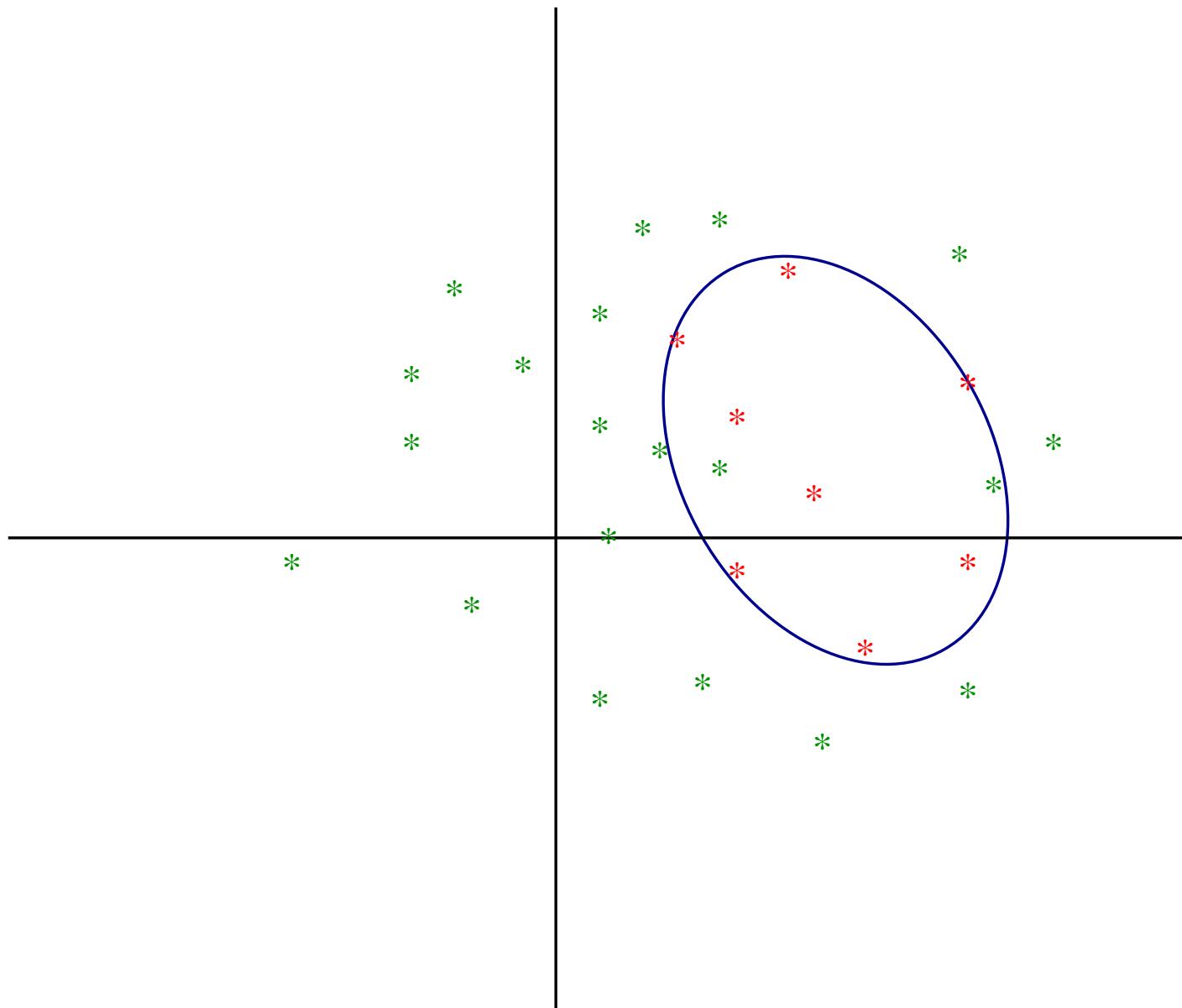


Figure 2: Quadratic Support Vector Machine

Unconstrained Optimization: Logistic Regression I

Similar to SVM, given the two-class **discrimination training data** points $\mathbf{a}_i \in R^n$, according to the logistic model, the probability that it's in a class C , say in **Red**, is represented by a linear/affine function with slope-vector \mathbf{x} and intersect scalar x_0 :

$$\frac{e^{\mathbf{a}_i^T \mathbf{x} + x_0}}{1 + e^{\mathbf{a}_i^T \mathbf{x} + x_0}}.$$

Thus, for some training data points, we like to determine intercept x_0 and slope vector $\mathbf{x} \in R^n$ such that

$$\frac{e^{\mathbf{a}_i^T \mathbf{x} + x_0}}{1 + e^{\mathbf{a}_i^T \mathbf{x} + x_0}} = \begin{cases} 1, & \text{if } \mathbf{a}_i \in C \\ 0, & \text{otherwise} \end{cases}.$$

Then the probability to give a “right classification answer” for all training data points is

$$\left(\prod_{\mathbf{a}_i \in C} \frac{e^{\mathbf{a}_i^T \mathbf{x} + x_0}}{1 + e^{\mathbf{a}_i^T \mathbf{x} + x_0}} \right) \left(\prod_{\mathbf{a}_i \notin C} \frac{1}{1 + e^{\mathbf{a}_i^T \mathbf{x} + x_0}} \right)$$

Logistic Regression II

Therefore, we like to maximize the probability when deciding intercept x_0 and slope vector $\mathbf{x} \in R^n$

$$\left(\prod_{\mathbf{a}_i \in C} \frac{e^{\mathbf{a}_i^T \mathbf{x} + x_0}}{1 + e^{\mathbf{a}_i^T \mathbf{x} + x_0}} \right) \left(\prod_{\mathbf{a}_i \notin C} \frac{1}{1 + e^{\mathbf{a}_i^T \mathbf{x} + x_0}} \right) = \left(\prod_{\mathbf{a}_i \in C} \frac{1}{1 + e^{-\mathbf{a}_i^T \mathbf{x} - x_0}} \right) \left(\prod_{\mathbf{a}_i \notin C} \frac{1}{1 + e^{\mathbf{a}_i^T \mathbf{x} + x_0}} \right),$$

which is equivalently to maximize

$$-\left(\sum_{\mathbf{a}_i \in C} \ln(1 + e^{-\mathbf{a}_i^T \mathbf{x} - x_0}) \right) - \left(\sum_{\mathbf{a}_i \notin C} \ln(1 + e^{\mathbf{a}_i^T \mathbf{x} + x_0}) \right).$$

Or

$$\min_{x_0, \mathbf{x}} \left(\sum_{\mathbf{a}_i \in C} \ln(1 + e^{-\mathbf{a}_i^T \mathbf{x} - x_0}) \right) + \left(\sum_{\mathbf{a}_i \notin C} \ln(1 + e^{\mathbf{a}_i^T \mathbf{x} + x_0}) \right).$$

This is an **unconstrained optimization** problem, where the objective is a convex function of decision variables: intercept x_0 and slope vector $\mathbf{x} \in R^n$.

More QP Examples: Portfolio Management

For expected return vector \mathbf{r} and co-variance matrix V of an investment portfolio, one management model is:

$$\begin{aligned} &\text{minimize} \quad \mathbf{x}^T V \mathbf{x} \\ &\text{subject to} \quad \mathbf{r}^T \mathbf{x} \geq \mu, \\ &\quad \mathbf{e}^T \mathbf{x} = 1, \quad \mathbf{x} \geq \mathbf{0}, \end{aligned}$$

$$\begin{aligned} &\text{minimize} \quad \mathbf{x}^T V \mathbf{x} \\ &\text{or simply} \quad \text{subject to} \quad \mathbf{r}^T \mathbf{x} \geq \mu, \\ &\quad \mathbf{e}^T \mathbf{x} = 1, \end{aligned}$$

where \mathbf{e} is the vector of all ones.

This is a (convex) quadratic program.

More CLP Examples: Robust Portfolio Management

In applications, \mathbf{r} and V may be estimated under various scenarios, say \mathbf{r}_i and V_i for $i = 1, \dots, m$. Then, we like

$$\begin{array}{ll} \text{minimize} & \max_i \mathbf{x}^T V_i \mathbf{x} \\ \text{subject to} & \min_i \mathbf{r}_i^T \mathbf{x} \geq \mu, \\ & \mathbf{e}^T \mathbf{x} = 1, \mathbf{x} \geq \mathbf{0}. \end{array} \Rightarrow \begin{array}{ll} \text{minimize} & \alpha \\ \text{subject to} & \mathbf{r}_i^T \mathbf{x} \geq \mu, \forall i \\ & \sqrt{\mathbf{x}^T V_i \mathbf{x}} \leq \alpha, \forall i \\ & \mathbf{e}^T \mathbf{x} = 1, \mathbf{x} \geq \mathbf{0}. \end{array}$$

This is a quadratically constrained quadratic program (QCQP). If factorize $V_i = R_i^T R_i$ and let $\mathbf{y}_i = R_i \mathbf{x}$, we can rewrite the problem as

$$\begin{array}{ll} \text{minimize} & \alpha \\ \text{subject to} & \mathbf{r}_i^T \mathbf{x} \geq \mu, \mathbf{y}_i - R_i \mathbf{x} = \mathbf{0}, \forall i \\ & \|\mathbf{y}_i\| \leq \alpha, \forall i, \mathbf{e}^T \mathbf{x} = 1, \mathbf{x} \geq \mathbf{0}, \end{array}$$

which is an SOCP with additional benefits.

Portfolio Selection Problem

If no more than k stocks can be selected into your portfolio as a policy constraint?

$$\text{minimize} \quad \mathbf{x}^T V \mathbf{x}$$

$$\text{subject to} \quad \mathbf{r}^T \mathbf{x} \geq \mu,$$

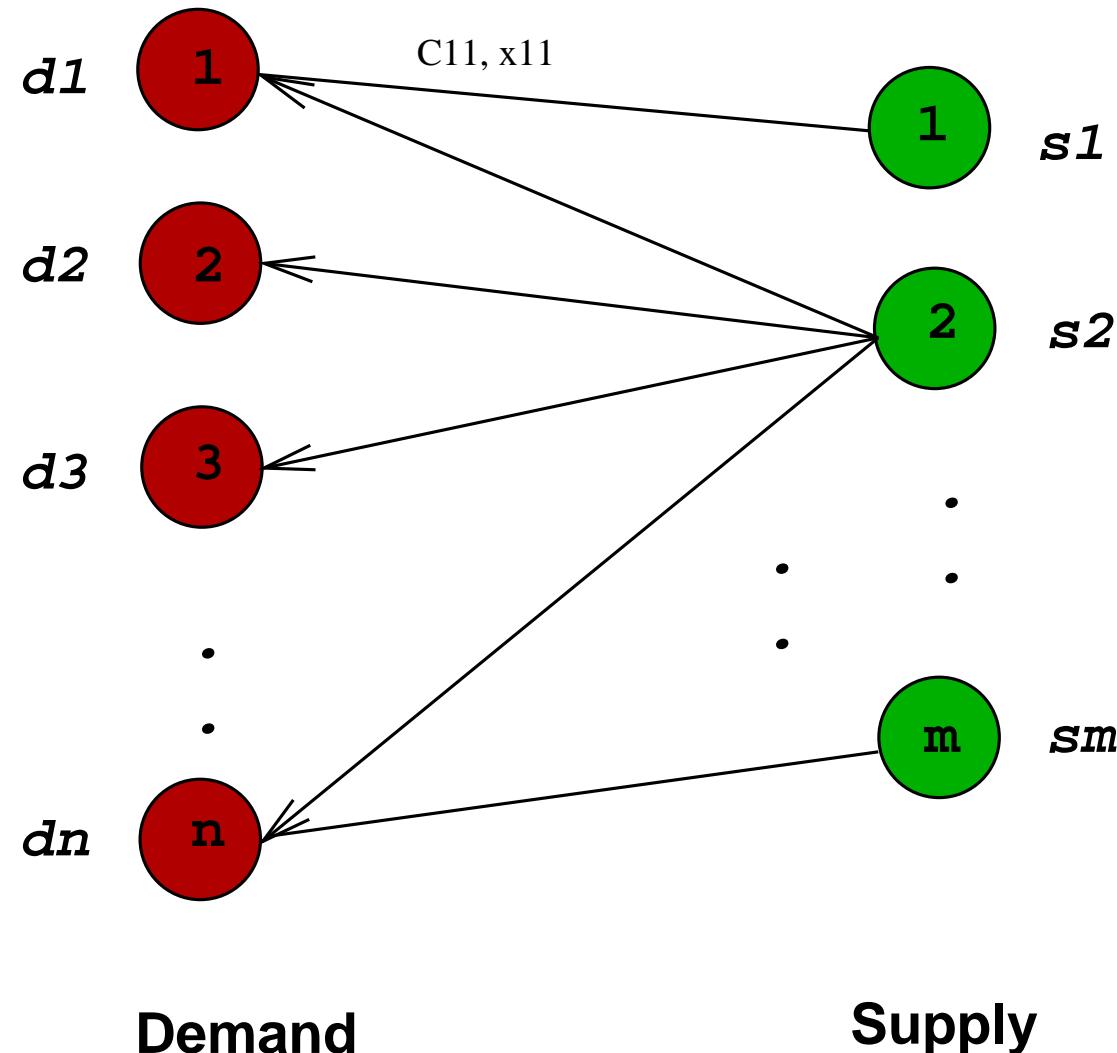
$$\mathbf{e}^T \mathbf{x} = 1,$$

$$\mathbf{0} \leq \mathbf{x} \leq \mathbf{y}, \quad \mathbf{e}^T \mathbf{y} \leq k, \quad \mathbf{y} \in \{0, 1\}^n$$

This is a mixed-integer quadratic program (MIP).

If the integer variables are restricted 0 or 1, it is also names as the binary optimization problem.

The Transportation Problem



Mathematical Optimization Model:

$$\begin{aligned}
 \min \quad & \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \\
 \text{s.t.} \quad & \sum_{j=1}^n x_{ij} = s_i, \quad \forall i = 1, \dots, m \\
 & \sum_{i=1}^m x_{ij} = d_j, \quad \forall j = 1, \dots, n \\
 & x_{ij} \geq 0, \quad \forall i, j.
 \end{aligned}$$

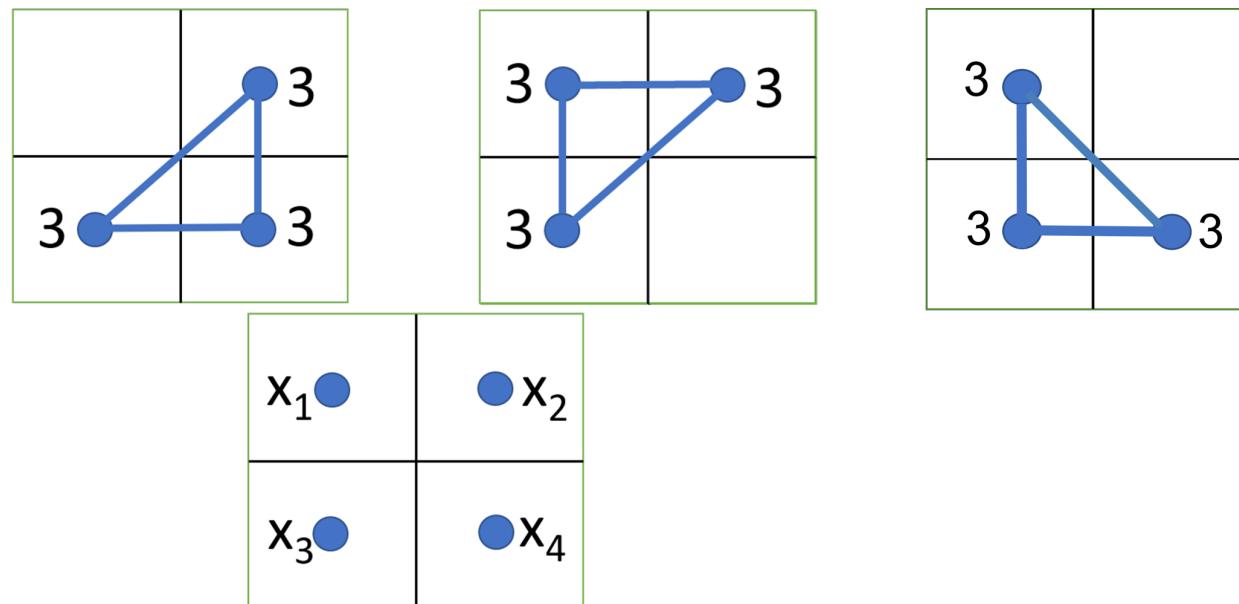
here we can normalize the supply and demand side to 1

The minimal transportation cost is called the **Wasserstein Distance (WD)** between supply distribution **s** and demand distribution **d** (can be interpreted as two probability distributions after normalization). This is a **linear program!**

What happen if supplies **s** are also **decision variables**?

The **Wasserstein Barycenter Problem** is to find a distribution such that the sum of its Wasserstein Distance to each of a set of distributions would be minimized.

A Wasserstein Barycenter Application: Stochastic Optimization



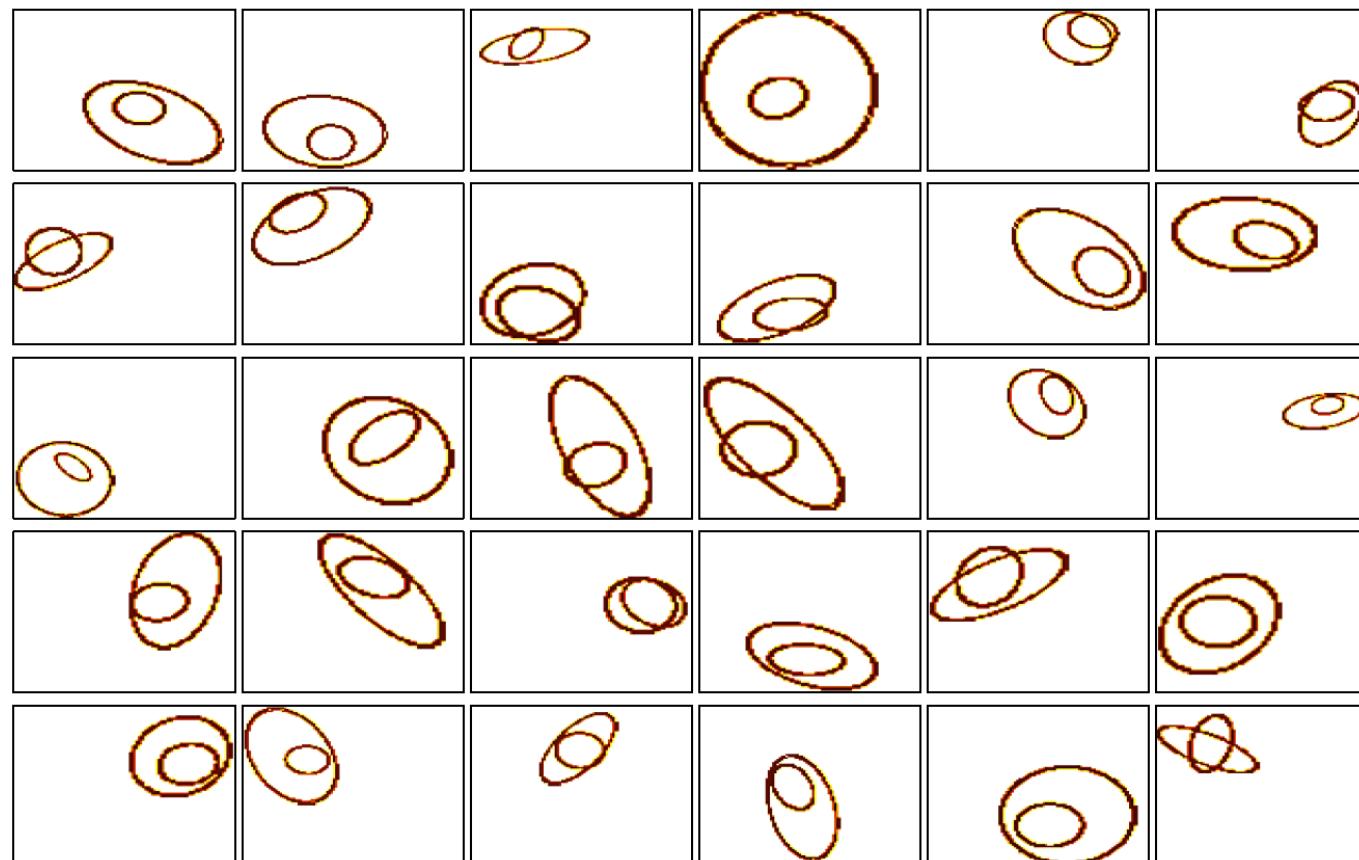
Find distribution of $x_i, i = 1, 2, 3, 4$ to minimize

$$\begin{aligned} \min \quad & WD_l(\mathbf{x}) + WD_m(\mathbf{x}) + WD_r(\mathbf{x}) \\ \text{s.t.} \quad & x_1 + x_2 + x_3 + x_4 = 9, \quad x_i \geq 0, \quad i = 1, 2, 3, 4. \end{aligned}$$

The objective is a nonlinear function, but its gradient vector $\nabla WD_l(\mathbf{x})$, $\nabla WD_m(\mathbf{x})$ and $\nabla WD_r(\mathbf{x})$ are shadow prices of the three sub-transportation problems –popularly used in **Hierarchy Optimization**.

The Wasserstein Barycenter (Mean) Problem in Data Science

What is the “mean or consensus” image from a set of images/distributions:



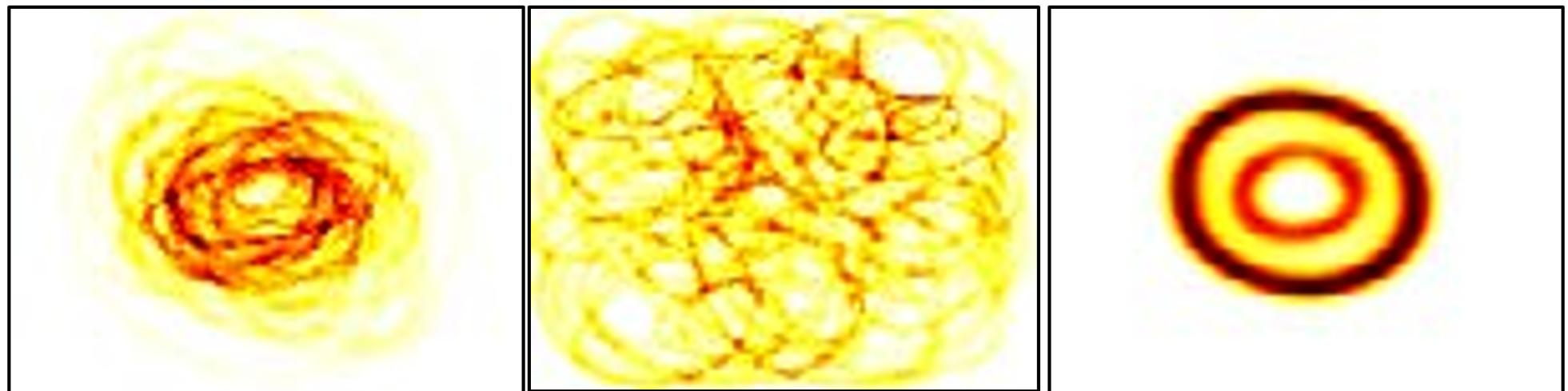


Figure 3: Mean picture constructed from the (a) Euclidean mean after re-centering images (b) Euclidean mean (c) Wasserstein Barycenter (self recenter, resize and rotate)

Euclidean Mean/Center:

$$\mathbf{x} = \frac{1}{n} \sum_{i=1}^n \mathbf{a}_i, \quad \text{or} \quad \min_{\mathbf{x}} \sum_{i=1}^n \|\mathbf{x} - \mathbf{a}_i\|_2^2,$$

which is an unconstrained optimization, or least-squares, problem

Graph Realization and Sensor Network Localization

Given a graph $G = (V, E)$ and sets of non-negative **weights**, say $\{d_{ij} : (i, j) \in E\}$, the goal is to compute a **realization** of G in the **Euclidean space** \mathbf{R}^d for a **given low dimension** d , where the distance information is preserved.

More precisely: given anchors $\mathbf{a}_k \in \mathbf{R}^d$, $d_{ij} \in N_x$, and $\hat{d}_{kj} \in N_a$, find $\mathbf{x}_i \in \mathbf{R}^d$ such that

$$\begin{aligned}\|\mathbf{x}_i - \mathbf{x}_j\|^2 &= d_{ij}^2, \quad \forall (i, j) \in N_x, i < j, \\ \|\mathbf{a}_k - \mathbf{x}_j\|^2 &= \hat{d}_{kj}^2, \quad \forall (k, j) \in N_a.\end{aligned}$$

This is a set of Quadratic Equations, which can be represented as an optimization problem:

$$\min_{\mathbf{x}_i \forall i} \sum_{(i, j) \in N_x} (\|\mathbf{x}_i - \mathbf{x}_j\|^2 - d_{ij}^2)^2 + \sum_{(k, j) \in N_a} (\|\mathbf{a}_k - \mathbf{x}_j\|^2 - \hat{d}_{kj}^2)^2.$$

Does the system have a localization or realization of all \mathbf{x}_j 's? Is the localization **unique**? Is there a **certification** for the solution to make it **reliable or trustworthy**? Is the system **partially** localizable with certification?

It can be relaxed to SOCP (change “=” to “ \leq ”) or SDP.

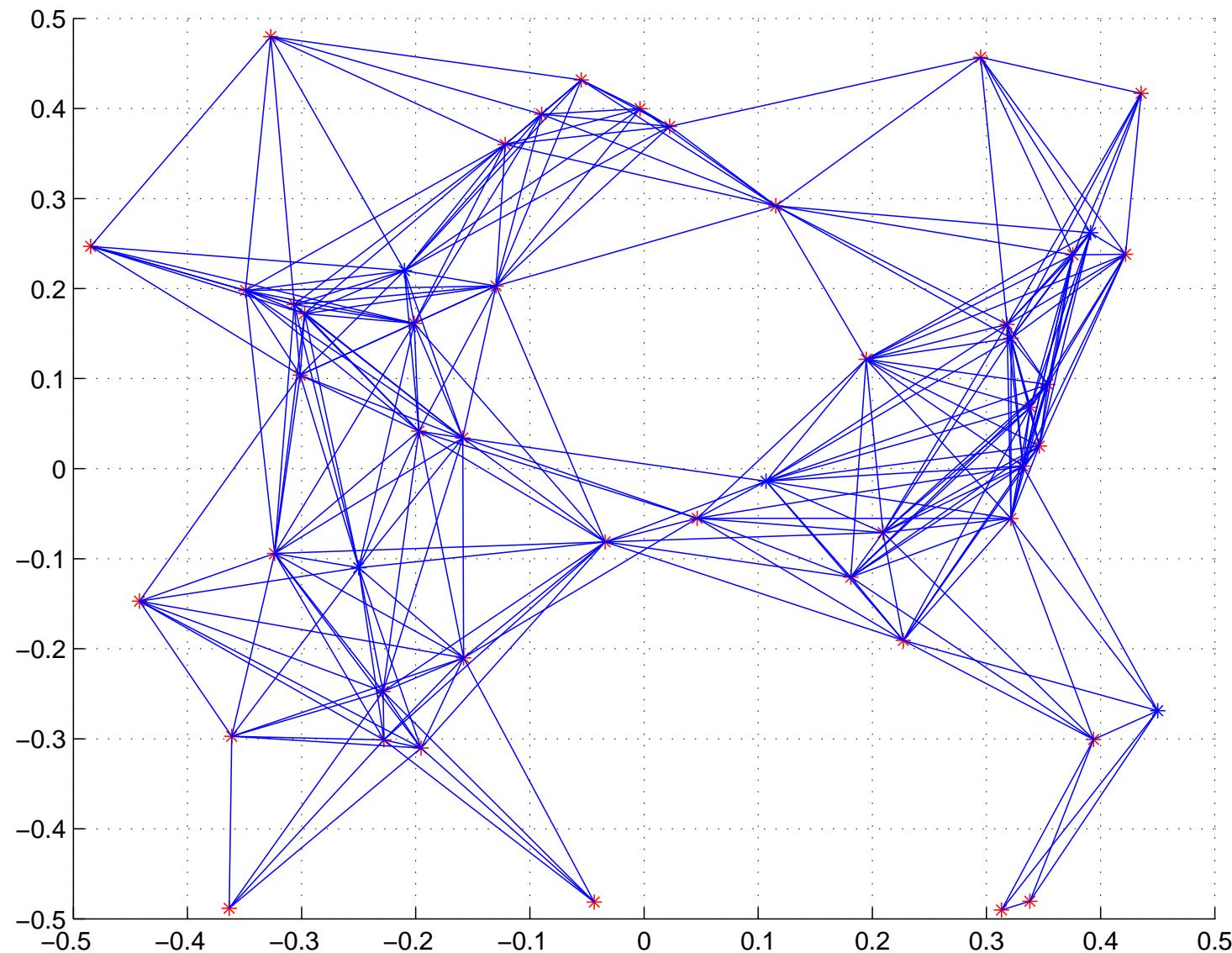


Figure 4: 50-node 2-D Sensor Localization.

Matrix Representation of SNL and SDP Relaxation

Let $X = [\mathbf{x}_1 \ \mathbf{x}_2 \ \dots \ \mathbf{x}_n]$ be the $d \times n$ matrix that needs to be determined and \mathbf{e}_j be the vector of all zero except 1 at the j th position. Then

$$\mathbf{x}_i - \mathbf{x}_j = X(\mathbf{e}_i - \mathbf{e}_j) \quad \text{and} \quad \mathbf{a}_k - \mathbf{x}_j = [I \ X](\mathbf{a}_k; -\mathbf{e}_j)$$

so that

$$\|\mathbf{x}_i - \mathbf{x}_j\|^2 = (\mathbf{e}_i - \mathbf{e}_j)^T X^T X (\mathbf{e}_i - \mathbf{e}_j)$$

$$\|\mathbf{a}_k - \mathbf{x}_j\|^2 = (\mathbf{a}_k; -\mathbf{e}_j)^T [I \ X]^T [I \ X] (\mathbf{a}_k; -\mathbf{e}_j) =$$

$$(\mathbf{a}_k; -\mathbf{e}_j)^T \begin{pmatrix} I & X \\ X^T & X^T X \end{pmatrix} (\mathbf{a}_k; -\mathbf{e}_j).$$

Or, equivalently,

$$\begin{aligned}
 (\mathbf{e}_i - \mathbf{e}_j)^T Y (\mathbf{e}_i - \mathbf{e}_j) &= d_{ij}^2, \quad \forall i, j \in N_x, \quad i < j, \\
 (\mathbf{a}_k; -\mathbf{e}_j)^T \begin{pmatrix} I & X \\ X^T & Y \end{pmatrix} (\mathbf{a}_k; -\mathbf{e}_j) &= \hat{d}_{kj}^2, \quad \forall k, j \in N_a, \\
 Y &= X^T X.
 \end{aligned}$$

Relax $Y = X^T X$ to $Y \succeq X^T X$, which is equivalent to matrix inequality:

$$\begin{pmatrix} I & X \\ X^T & Y \end{pmatrix} \succeq \mathbf{0}.$$

This matrix has rank at least d ; if it's d , then $Y = X^T X$, and the converse is also true.

The problem is now an SDP problem.

Convex relaxation first and steepest-descent-search second strategy?

Stochastic Optimization and Learning

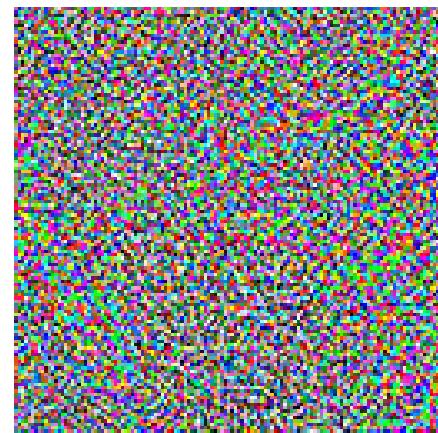
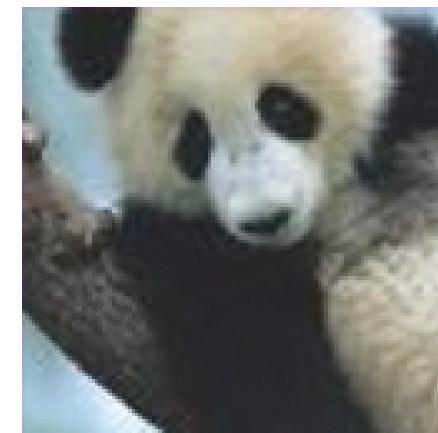
In real world, we most often do

$$\text{minimize}_{\mathbf{x} \in X} \quad \mathbf{E}_{F_\xi} [h(\mathbf{x}, \xi)] \quad (1)$$

where ξ represents random variables with the joint distribution F_ξ .

- Pros: In many cases, the expected value is a good measure of performance
- Cons: One has to know the exact distribution of ξ to perform the stochastic optimization so that we most frequently use sample distribution. Then, deviant from the assumed distribution may result in sub-optimal solutions. Even know the distribution, the solution/decision is generically risky.

Learning with Noises/Distortions

 $+ \epsilon$  $=$ 

“panda”

57.7% confidence

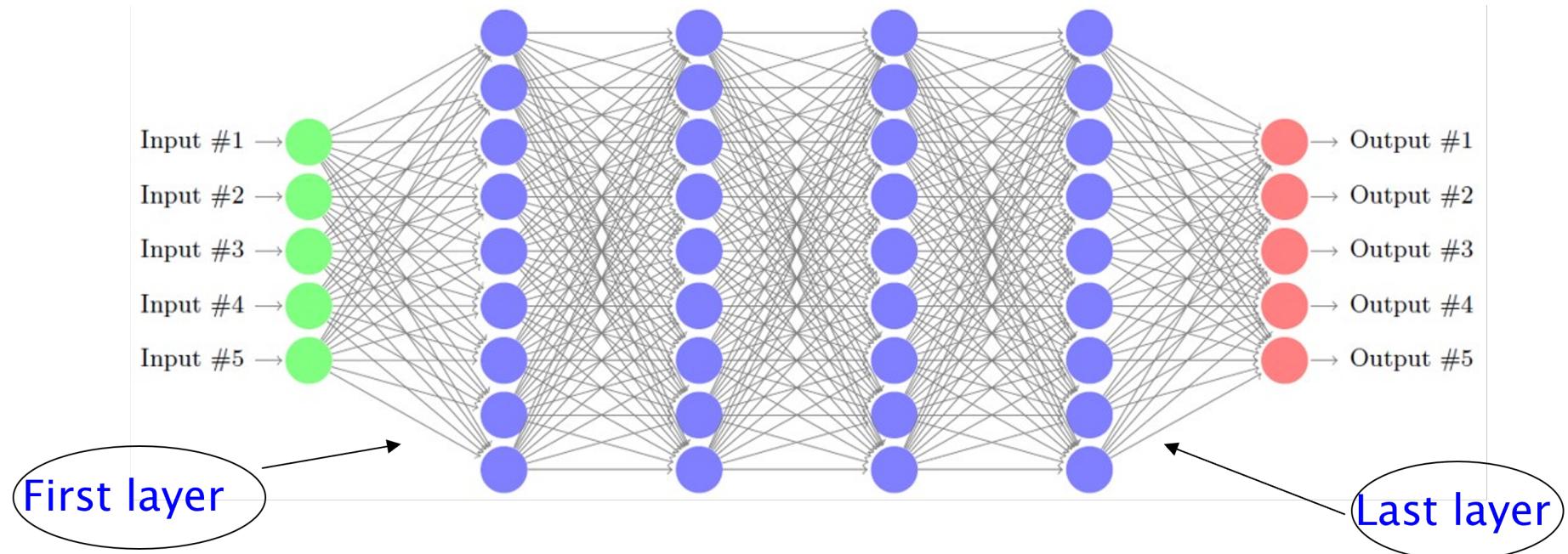
“gibbon”

99.3% confidence



Goodfellow et al. [2014]

Deep-Learning on Neural-Network I



The **input vector** is denoted by \mathbf{x} and the **output vector** of layer l is denoted by \mathbf{y}^l . The **edge-weights** of layer l are denoted by $w_{i,j}^l$ where the relation of input-output is

$$y_j^l = \max\{0, w_{0,j}^l + \sum_i w_{i,j}^l y_j^{l-1}\}, \forall j, l = 1, \dots, L.$$

where the formula is called **ReLU operator/function** and $\mathbf{y}^0 = \mathbf{x}$.

Deep-Learning on Neural-Network II

The Deep-Learning is to use massive sample images/inputs \mathbf{x} to optimize/train (or learn edge-weights $w_{i,j}^l$) such that a (classification) sample-average error function is minimized. In other words, for this example, the outputs of images/inputs of Panda and Gibbon are distinguishable/separable, or they belong to different regions in the output space.

When all weights are determined, then the last-layer output vector of the neural-network, denoted by $\mathbf{y}^L(\mathbf{x})$, is a vector function/mapping of an input vector \mathbf{x} .

The neural network verification, for this example, is to find the smallest distortion of a given typical Panda image such that its output is in the output-region of normal Gibbon images, that is,

$$\text{minimize}_{\mathbf{x}} \quad \|\mathbf{x} - \hat{\mathbf{x}}\|^2$$

$$\text{subject to} \quad \mathbf{y}^L(\mathbf{x}) \in \text{a (convex) region outside of } \mathbf{y}^L(\hat{\mathbf{x}}).$$

A Neural-Network Verification Optimization Problem

The problem can be represented as a constrained problem:

$$\begin{aligned}
 & \text{minimize}_{\{\mathbf{x}, \mathbf{y}^l\}} \quad \|\mathbf{x} - \hat{\mathbf{x}}\|^2 \\
 & \text{subject to} \quad \mathbf{y}^L \in \text{a (convex) region outside of } \mathbf{y}^L(\hat{\mathbf{x}}), \\
 & \qquad \qquad \qquad y_j^l = \max\{0, w_{0,j}^l + \sum_i w_{i,j}^l y_j^{l-1}\}, \forall j, l = 1, \dots, L \\
 & \qquad \qquad \qquad \mathbf{y}^0 = \mathbf{x}.
 \end{aligned}$$

Convex Relaxation(?):

$$\begin{aligned}
 & \text{minimize}_{\{\mathbf{x}, \mathbf{y}^l\}} \quad \|\mathbf{x} - \hat{\mathbf{x}}\|^2 \\
 & \text{subject to} \quad \mathbf{y}^L \in \text{a (convex) region outside of } \mathbf{y}(\hat{\mathbf{x}}), \\
 & \qquad \qquad \qquad y_j^l \geq w_{0,j}^l + \sum_i w_{i,j}^l y_j^{l-1}, \forall j, l = 1, \dots, L \\
 & \qquad \qquad \qquad y_j^l \geq 0, \forall j, l = 1, \dots, L \\
 & \qquad \qquad \qquad \mathbf{y}^0 = \mathbf{x}.
 \end{aligned}$$

A Neural-Network Verification Decision Problem

A related problem would be a regularized problem where $\mathbf{y}^L(\mathbf{x})$ is in a (convex) polyhedral region outside of $\mathbf{y}^L(\hat{\mathbf{x}})$?

Linearly-Constrained Quadratic Minimization Problem:

$$\begin{aligned}
 & \text{minimize}_{\{\mathbf{x}, \mathbf{y}^l\}} \quad \|\mathbf{x} - \hat{\mathbf{x}}\|^2 + \mu \sum_l \sum_j y_j^l (y_j^l - w_{0,j}^l - \sum_i w_{i,j}^l y_j^{l-1}) \\
 & \text{subject to} \quad \mathbf{y}^L \in \text{a (convex) polyhedral region outside of } \mathbf{y}(\hat{\mathbf{x}}), \\
 & \quad y_j^l \geq w_{0,j}^l + \sum_i w_{i,j}^l y_j^{l-1}, \forall j, l = 1, \dots, L \\
 & \quad y_j^l \geq 0, \forall j, l = 1, \dots, L \\
 & \quad \mathbf{y}^0 = \mathbf{x}.
 \end{aligned}$$

The is a linearly constrained quadratic **optimization problem**.

Is the objective function **convex**?

Distributionally Robust Optimization and Learning

On the other hand: Why does error occur? Believing that the sample distribution is the true distribution...

In practice, although the exact distribution of the random variables may not be known, people usually know certain observed samples or training data and other statistical information. Thus, we can consider an enlarged distribution set \mathcal{D} that confidently containing the sample distribution, and do

$$\text{minimize}_{\mathbf{x} \in X} \quad \max_{F_\xi \in \mathcal{D}} \mathbf{E}_{F_\xi} [h(\mathbf{x}, \xi)] \quad (2)$$

In DRO, we consider a set of distributions \mathcal{D} and choose one to minimize the expected value for the worst distribution in \mathcal{D} . When choosing \mathcal{D} , we need to consider the following:

- Tractability
- Practical (Statistical) Meanings
- Performance (the potential loss comparing to the benchmark cases)

This is a nonlinear Saddle-Point Min-Max optimization/zero-sum-game problem

Reinforcement Learning: Markov Decision/Game Process

- RL/MDPs provide a mathematical framework for modeling **sequential** decision-making in situations where outcomes are partly **random** and partly under the control of a **decision maker**.
- Markov game processes (MGPs) provide a mathematical framework for modeling **sequential** decision-making of two-person turn-based zero-sum game.
- MDGPs are useful for studying a wide range of optimization/game problems solved via **dynamic programming**, where it was known at least as early as the 1950s (cf. Shapley 1953, Bellman 1957).
- Modern applications include dynamic planning under uncertainty, reinforcement learning, social networking, and almost all other stochastic **dynamic/sequential** decision/game problems in Mathematical, Physical, Management and Social Sciences.

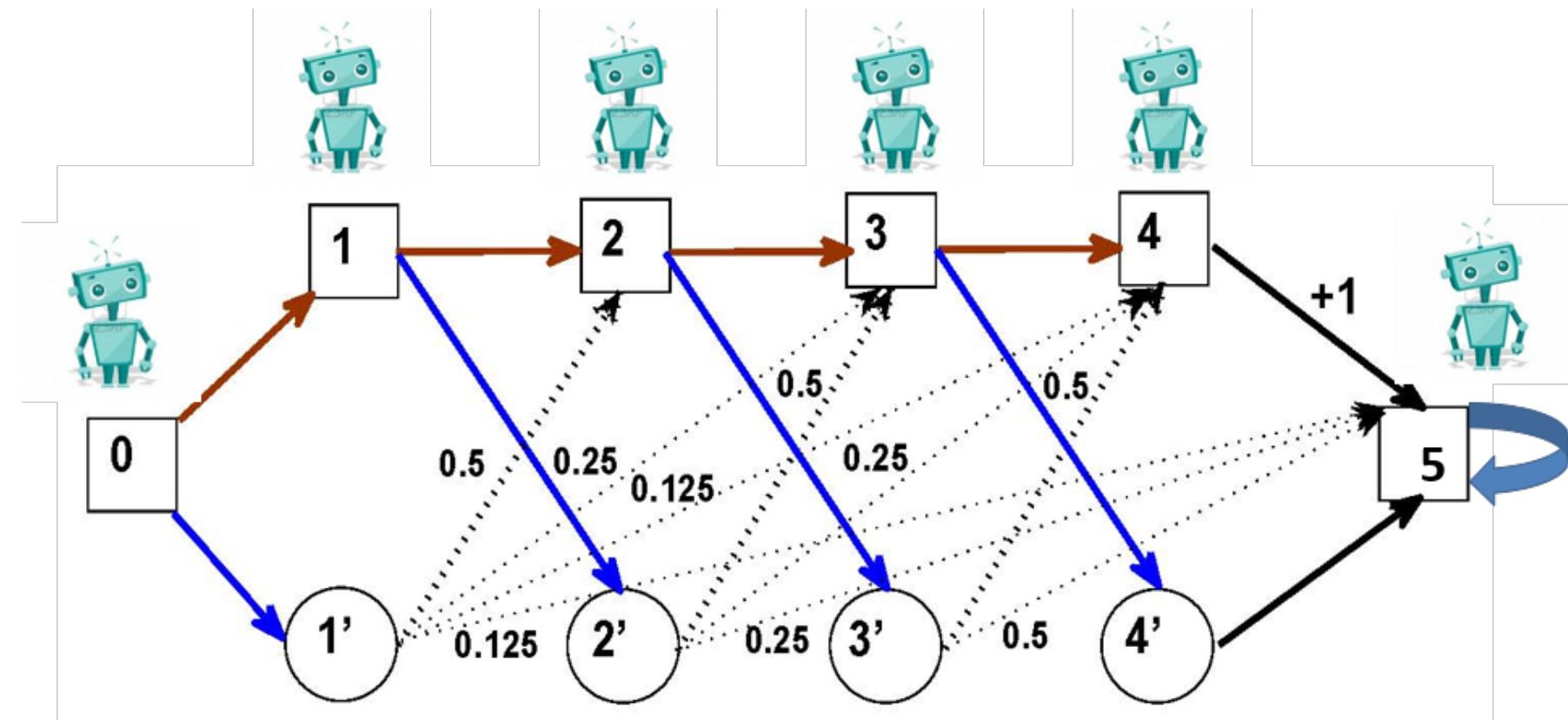
MDP Stationary Policy and Cost-to-Go Value

- An MDP problem is defined by a given number of states, indexed by i , where each state has a number of actions, \mathcal{A}_i , to take. Each action, say $j \in \mathcal{A}_i$, is associated with an (immediate) cost c_j of taking, and a probability distribution \mathbf{p}_j to transfer to all possible states at the next time period.
- A **stationary** policy for the decision maker is a function $\pi = \{\pi_1, \pi_2, \dots, \pi_m\}$ that specifies an action in each state, $\pi_i \in \mathcal{A}_i$, that the decision maker will take at any time period; which also lead to a **cost-to-go** value for each state.
- The MDP is to find a stationary policy to minimize/maximize the expected discounted sum over the **infinite horizon** with a discount factor $0 \leq \gamma < 1$:

$$\sum_{t=0}^{\infty} \gamma^t E[c^{\pi_{i^t}}(i^t, i^{t+1})].$$

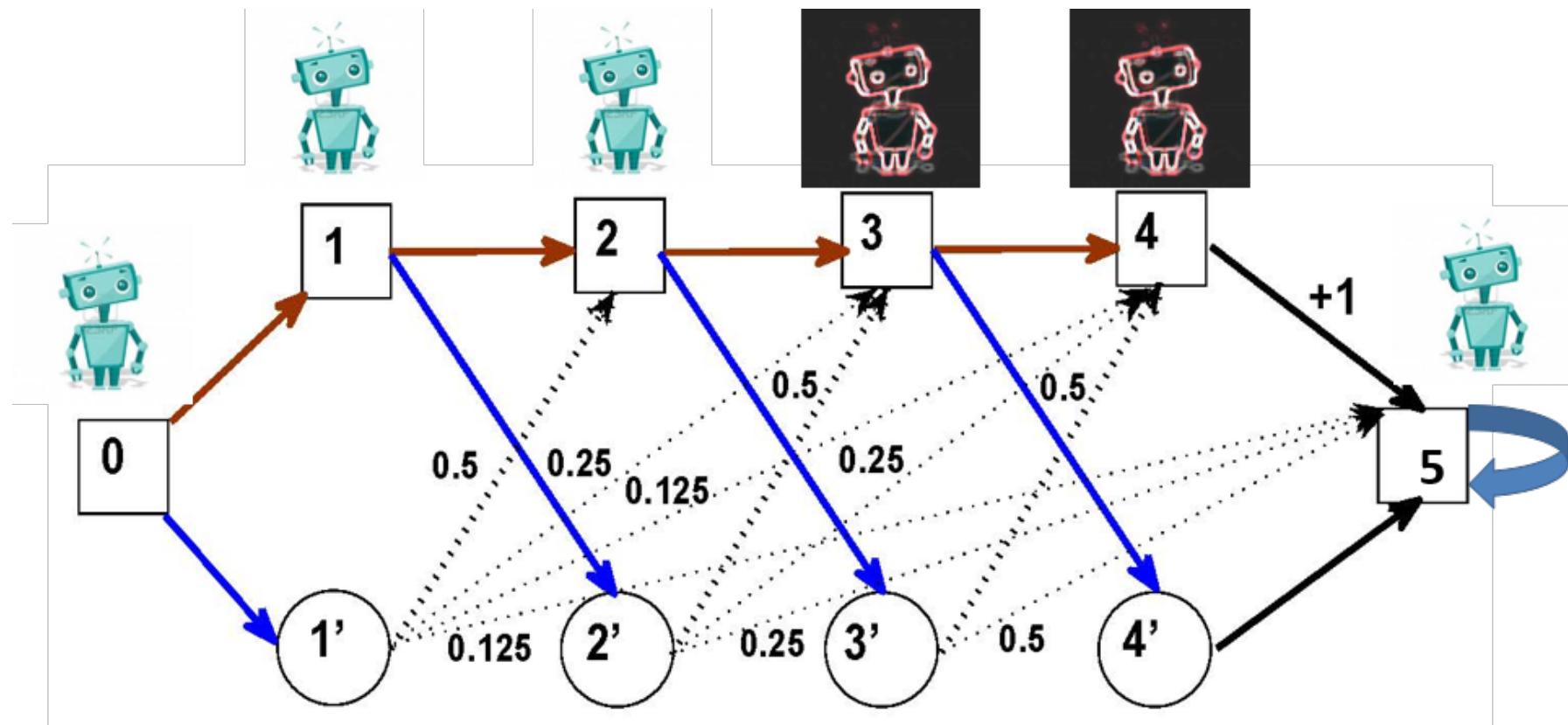
- If the states are partitioned into two sets, one is to minimize and the other is to maximize the discounted sum, then the process becomes a two-person turn-based zero-sum **stochastic game**.

An MDGP Toy Example: Maze Robot Runners



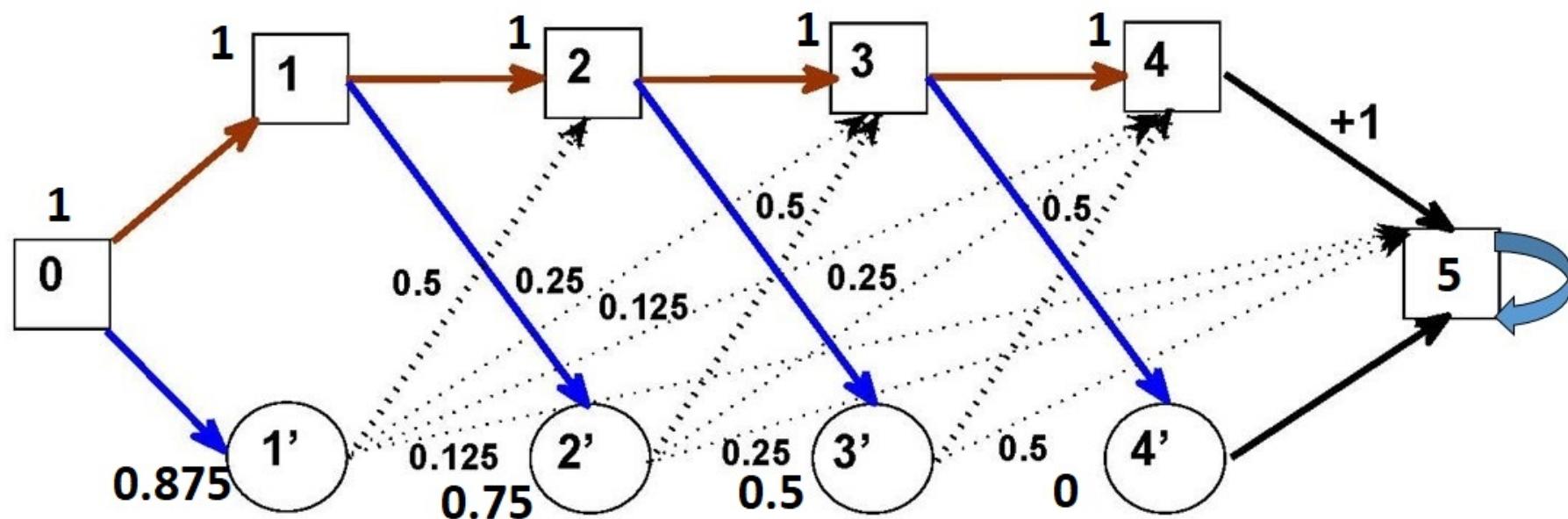
Actions are in red, blue and black; and all actions have zero cost except the state 4 to the exit/termination state 5. Which actions to take from every state to minimize the total cost (called optimal policy)?

Toy Example: Game Setting



States $\{0, 1, 2, 5\}$ minimize, while States $\{3, 4\}$ maximize.

The Cost-to-Go Values of the States



Cost-to-go values on each state when actions in red are taken: the current policy is not optimal since there are better actions to choose to minimize the cost.

The Optimal Cost-to-Go Value Vector

Let $\mathbf{y} \in \mathbf{R}^m$ represent the **cost-to-go** values of the m states, i th entry for i th state, of a given policy.

The MDP problem entails choosing an optimal policy where the corresponding cost-to-go value vector \mathbf{y}^* satisfying:

$$y_i^* = \min\{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^*, \forall j \in \mathcal{A}_i\}, \forall i,$$

with optimal policy

$$\pi_i^* = \arg \min\{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^*, \forall j \in \mathcal{A}_i\}, \forall i.$$

In the Game setting, the conditions becomes:

$$y_i^* = \min\{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^*, \forall j \in \mathcal{A}_i\}, \forall i \in I^-,$$

and

$$y_i^* = \max\{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^*, \forall j \in \mathcal{A}_i\}, \forall i \in I^+.$$

They both are **fix-point** or **saddle-point** optimization problems. The MDP problem can be cast as a linear program; see next page.

The Equivalent LP Formulation for MDP

This model can be reformulated as an LP:

$$\text{maximize}_{\mathbf{y}} \quad \sum_{i=1}^m y_i$$

$$\text{subject to} \quad y_1 - \gamma \mathbf{p}_j^T \mathbf{y} \leq c_j, \quad j \in \mathcal{A}_1$$

⋮

$$y_i - \gamma \mathbf{p}_j^T \mathbf{y} \leq c_j, \quad j \in \mathcal{A}_i$$

⋮

$$y_m - \gamma \mathbf{p}_j^T \mathbf{y} \leq c_j, \quad j \in \mathcal{A}_m.$$

Theorem 1 When \mathbf{y} is maximized, there must be at least one inequality constraint in \mathcal{A}_i that becomes equal for every state i , that is, maximal \mathbf{y} is a fixed point solution.

The Maze Runner Example

The Fixed-Point formulation:

$$\begin{aligned}y_0 &= \min\{0 + \gamma y_1, 0 + \gamma(0.5y_2 + 0.25y_3 + 0.125y_4 + 0.125y_5)\} \\y_1 &= \min\{0 + \gamma y_2, 0 + \gamma(0.5y_3 + 0.25y_4 + 0.25y_5)\} \\y_2 &= \min\{0 + \gamma y_3, 0 + \gamma(0.5y_4 + 0.5y_5)\} \\y_3 &= \min\{0 + \gamma y_4, 0 + \gamma y_5\} \\y_4 &= 1 + \gamma y_5 \\y_5 &= 0 \text{ (or } y_5 = 0 + \gamma y_5)\end{aligned}$$

The LP formulation:

$$\text{maximize}_y \quad y_0 + y_1 + y_2 + y_3 + y_4 + y_5$$

subject to change each equality above into inequality

The Interpretations of the LP Formulation

The LP variables $\mathbf{y} \in \mathbf{R}^m$ represent the expected present **cost-to-go** values of the m states, respectively, for a given policy.

The LP problem entails choosing variables in \mathbf{y} , one for each state i , that maximize $\mathbf{e}^T \mathbf{y}$ so that it is the **fixed point**

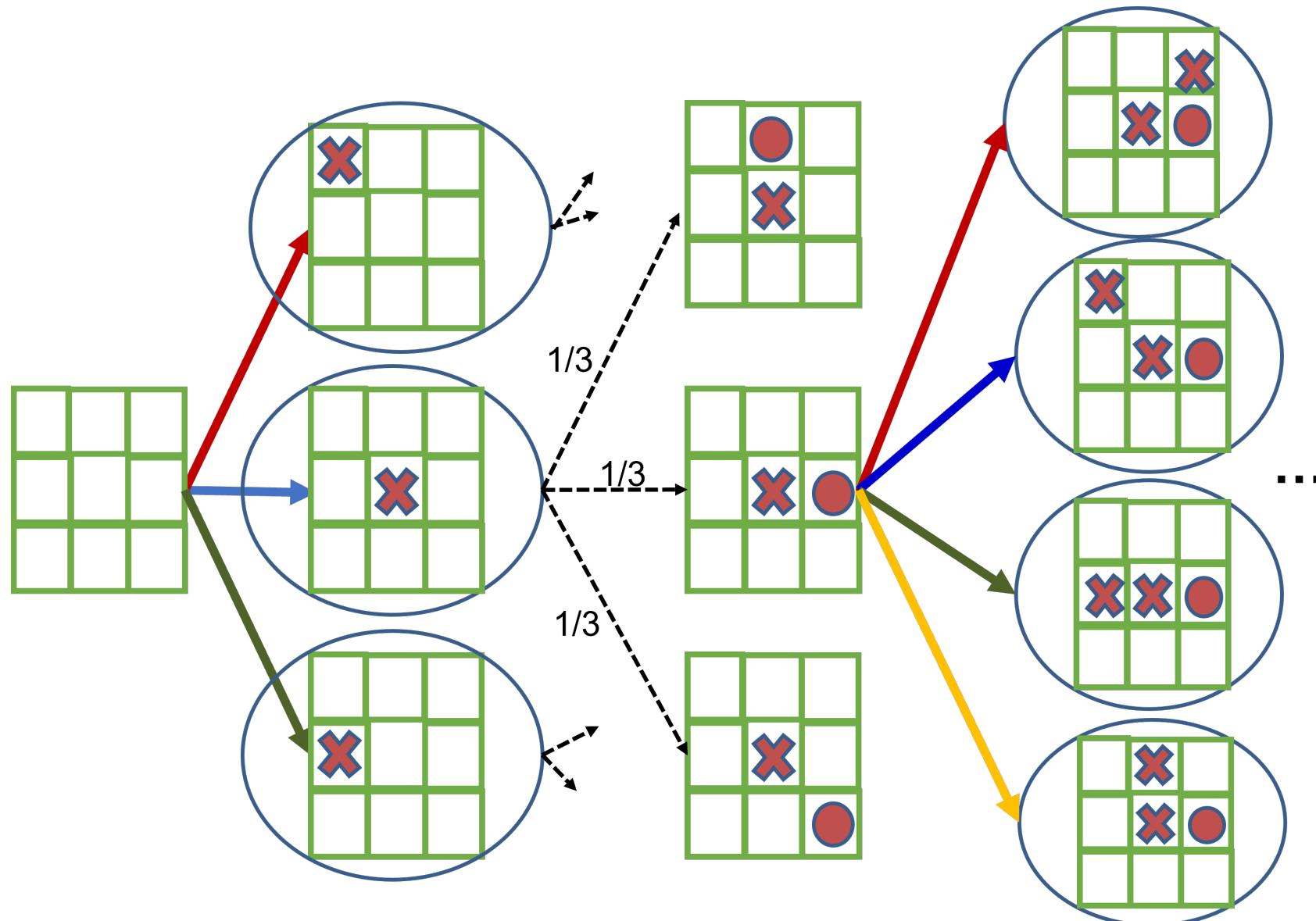
$$y_i^* = \min_{j \in \mathcal{A}_i} \{\mathbf{c}_{j_i} + \gamma \mathbf{p}_{j_i}^T \mathbf{y}\}, \quad \forall i,$$

with an optimal policy

$$\pi_i^* = \arg \min \{\mathbf{c}_j + \gamma \mathbf{p}_j^T \mathbf{y}, j \in \mathcal{A}_i\}, \quad \forall i.$$

It is well known that there exist a **unique** optimal stationary policy value vector \mathbf{y}^* where, for each state i , y_i^* is the minimum expected present cost that an individual in state i and its progeny can incur.

States/Actions in the Tic-Tac-Toe Game



Action Costs in the Tic-Tac-Toe Game

