CS57300
PURDUE UNIVERSITY

FEBRUARY 14, 2019

DATA MINING

OPTIMIZATION

OPTIMIZATION IN MODEL LEARNING

- Consider a **space** of possible models $M=\{M_1, M_2, \ldots, M_k\}$ with parameters θ
- Search over model structures or parameters, e.g.:
  - **Parameters**: In a logistic regression model, what are regression coefficients (**w**) that maximize log likelihood on the training data?
  - **Model structure**: In a decision trees, what is the tree structure that minimizes 0/1 loss on the training data?
- Find the best model structure or parameter values that **optimize** scoring function value on the training dataset

COMBINATORIAL OPTIMIZATION VS. SMOOTH OPTIMIZATION

- **Combinatorial** optimization:
  - The model space is a finite or countably infinite set (i.e., the scoring function is discrete)
  - Systematically search through the model space, often using heuristics
  - Example: Search the best decision tree structure
- **Smooth** optimization:
  - The model space is an uncountable set (i.e., the scoring function is continuous)
  - Gradient-based optimization
  - Example: Find parameter values for Naive Bayes Classifier

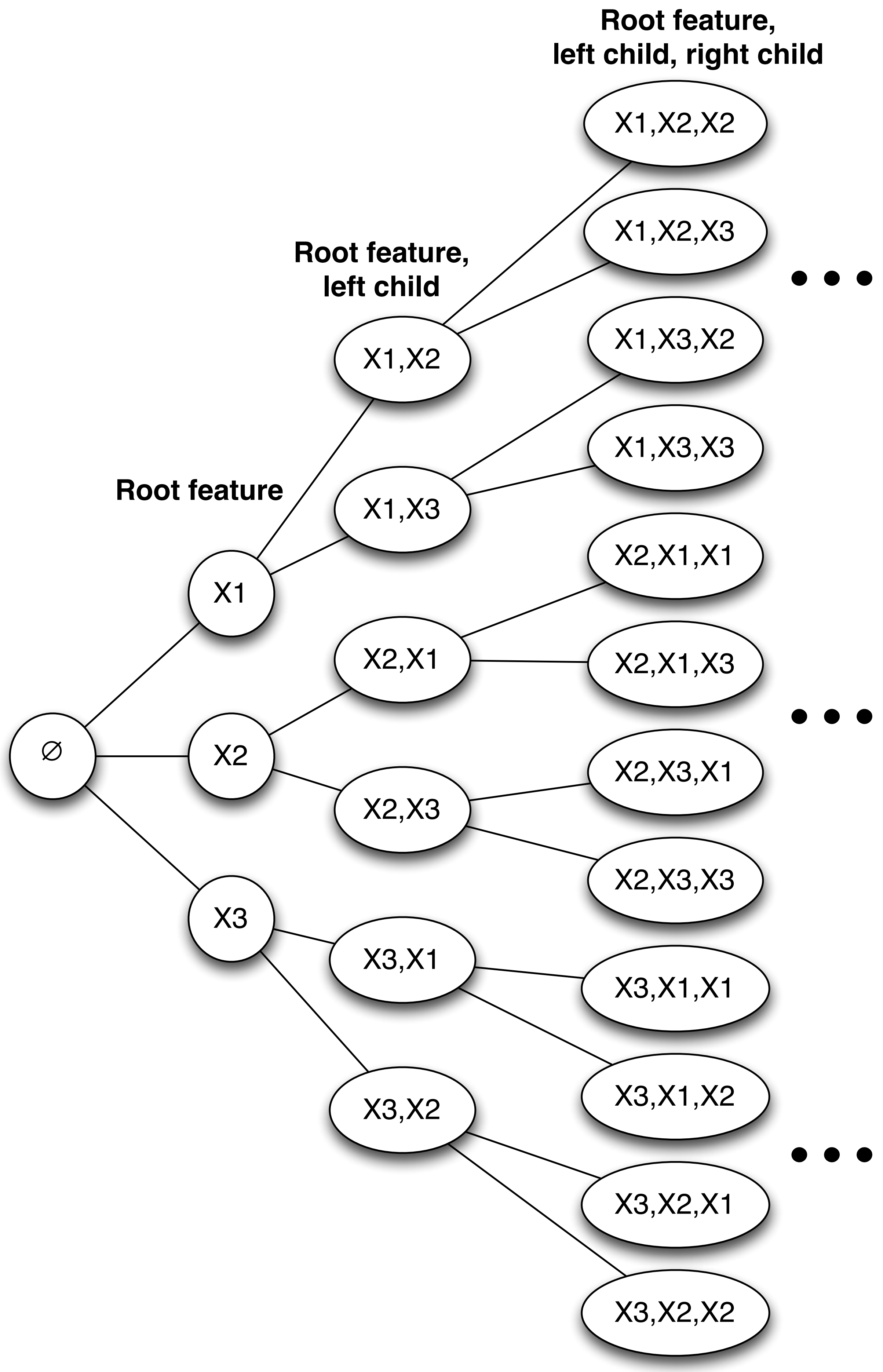
COMBINATORIAL OPTIMIZATION

COMBINATORIAL OPTIMIZATION: STATE SPACE

- S: state; the set of all possible models
- $Action(s)$: the set of all possible actions that can be performed at state s
- $Result(s, a)$: the result of performing action a on state s, which is another state
- $Score(s)$: the scoring function value of state s (i.e., for the model represented by state s)
- State space: representing each state as a node, and two nodes s and s' are connected by an edge if $s'=\text{Result}(s, a)$ for some a in $\text{Action}(s)$

STATE SPACE EXAMPLE

- Constructing the state space of decision trees where each data point has three binary variables X_1, X_2, X_3



SEARCH THROUGH THE STATE SPACE

- Start from a particular state (i.e., model)
- Evaluate the score of the current state
- If the current state is not the goal state (e.g., model with maximum score), expand the current state by applying all possible actions to the current state and generate successor states
- Pick one of the successor state, repeat, and backtrack
- Exhaustive search: systematic search through all possible states in the state space
  - e.g., depth-first search, breadth-first search, etc.

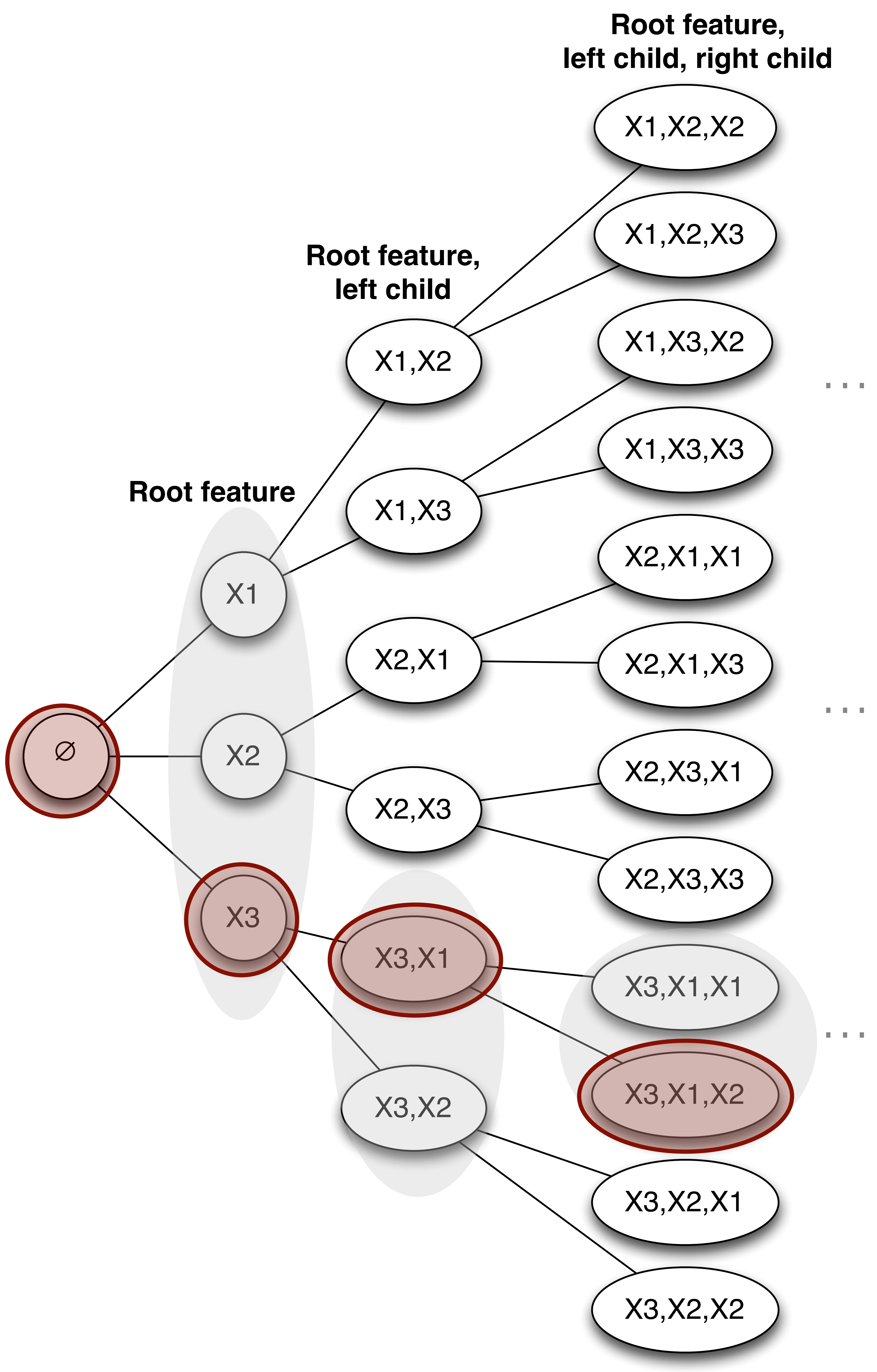
HEURISTIC SEARCH

- Typically, there is an exponential number of models in the model space, making it intractable to exhaustively search the space
  - Thus, it is generally impossible to return a model that is **guaranteed** to have the best score
- Instead, we have to resort to **heuristic** search techniques
  - Methods are evaluated experimentally and shown to have good performance on average
  - **Greedy** search: Given a current model M, look for the successor of M and move to the best of these (if any have a score better than M)

GREEDY SEARCH

- Choose an initial state M^0 corresponding to a particular model structure (e.g., an empty tree)
- Let M^i be the model considered at the i-th iteration
- For each iteration i
  - Construct all possible models $\{M^{j1}, \ldots, M^{jk}\}$ adjacent to M^i (as defined by action operators)
  - Evaluate scores for all models $\{M^{j1}, \ldots, M^{jk}\}$
  - Choose to move to the adjacent model with best score: $M^{i+1} = M^{j.best}$
  - Repeat until there is no possible further improvement in the score

Which states does greedy search consider?



SMOOTH OPTIMIZATION

SMOOTH OPTIMIZATION

- **Smooth** scoring functions:
  - If a function is *smooth*, it is differentiable and the derivatives are continuous, then we can use gradient-based optimization
    - If function is *convex*, we can often solve the minimization problem using **convex optimization** or **gradient descent**
    - If function is smooth but non-linear, we can use iterative search over the surface of S to find a local minimum (e.g., hill-climbing)

CONVEX OPTIMIZATION PROBLEMS

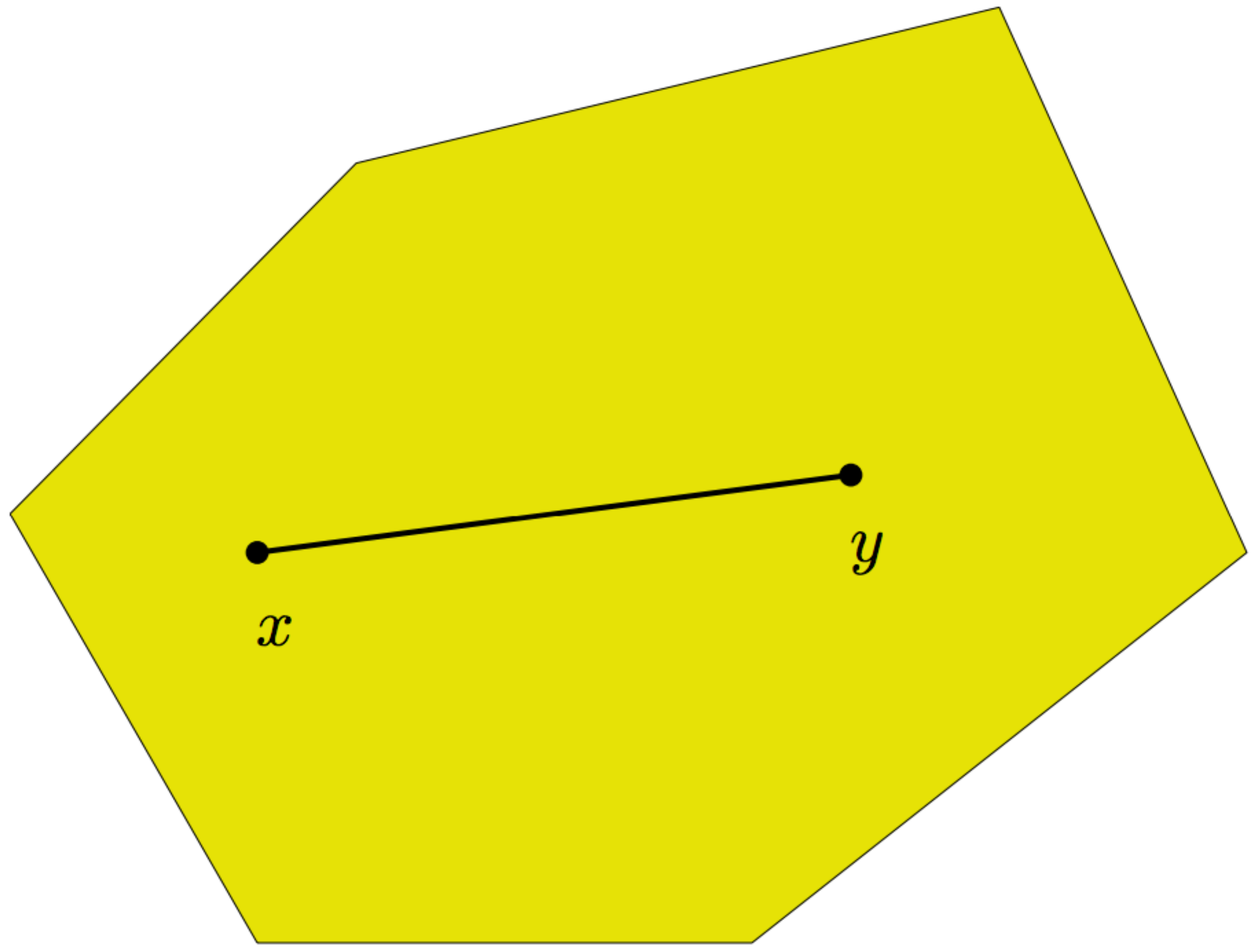
$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in C \end{array}$$

- *x* is the optimization variable (*e.g., model parameters*)
  f (*e.g., score function*) is a **convex function**
  C is a **convex set** (*e.g., constraints on model parameters*)
- For convex optimization problems, all locally optimal points are globally optimal

CONVEX SET

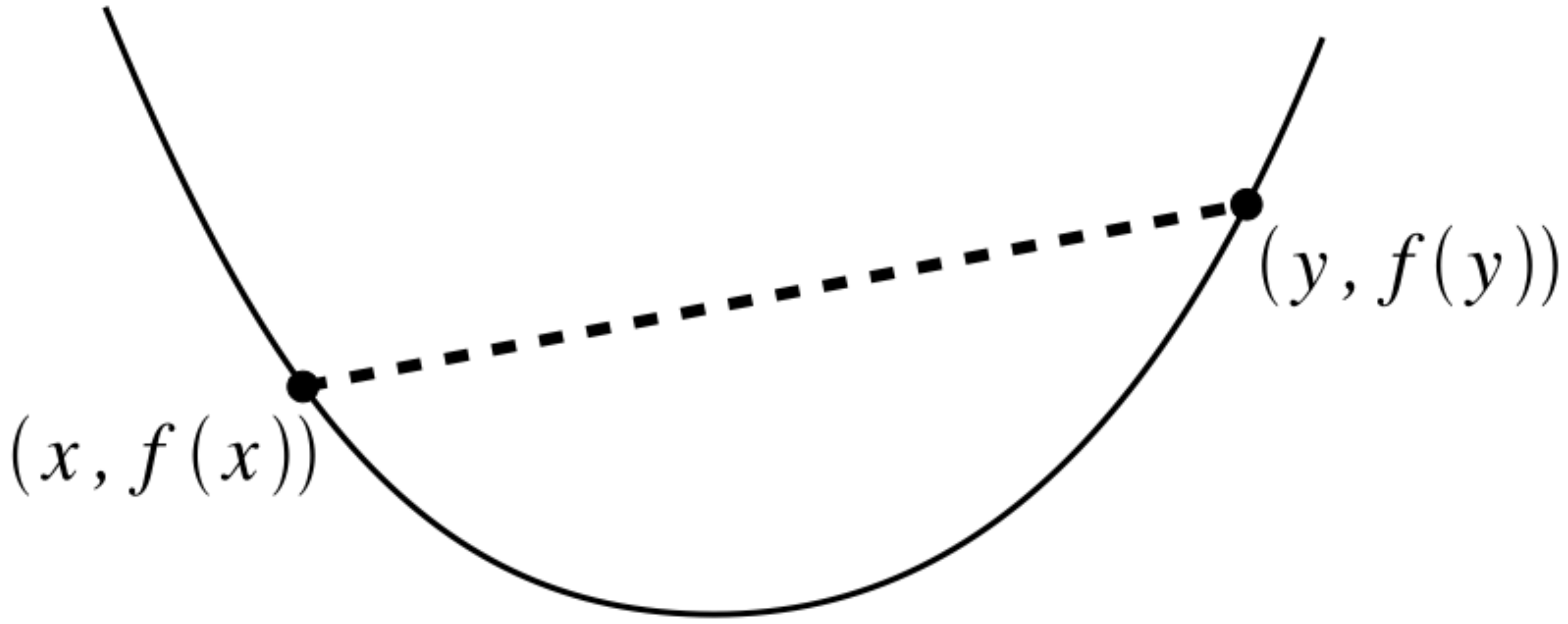
- A set C is convex if for any $x, y \in C$ and any θ with $0 \leq \theta \leq 1$ we have

$$\theta x + (1 - \theta) y \in C$$



CONVEX FUNCTIONS

- In graph of convex function *f*, the line connecting two points must lie above the function: $f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y)$ for all $0 \leq \alpha \leq 1$
- Practical test for convexity: a twice differentiable function *f* of a variable *x* is convex on an interval if an only if for any *x* in the interval: $f''(x) \geq 0$
  - Strictly convex if $f''(x) > 0$



- Sum of convex functions is convex; max of convex functions is convex

SOLVE CONVEX OPTIMIZATION PROBLEM

- Minimize a convex function without any constraints on the variables
  - If $f'(x)=0$ then x is a stationary point of f
  - If $f'(x)=0$ and $f''(x)$ is not negative then x is a local minimum of f (for convex function, this is also a global minimum)
  - If f is a strictly convex function, any stationary point of ***f*** is the unique global minimum of f
- What about minimizing a convex function with constraints?

USE LAGRANGE MULTIPLIERS TO SOLVE CONVEX OPTIMIZATION

- For a standard form of convex optimization problem (f_0 are f_i are convex, h_i is linear):

$$\begin{array}{lll} \text{minimize} & f_0(x) & \\ \text{subject to} & f_i(x) \leq 0, & \text{for } i = 1, \ldots, m. \\ & h_i(x) = 0, & \text{for } i = 1, \ldots, k. \end{array}$$

- The Lagrangian function of it is

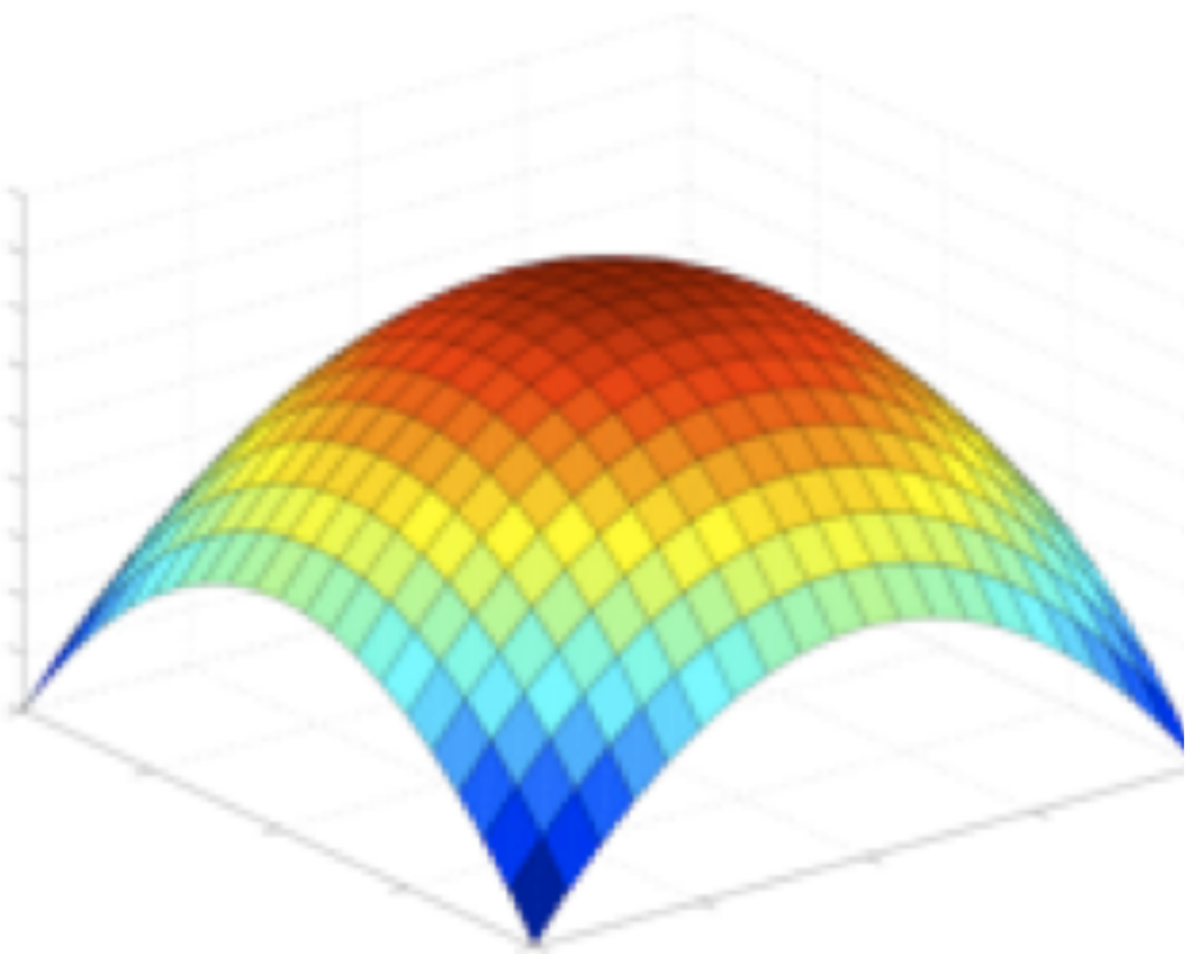
$$L(x, \lambda, v) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{k} v_i h_i(x)$$

  - $\lambda_i \geq 0$ is the **Lagrange multiplier** for the *i*-th inequality constraint, v_i is the **Lagrange multiplier** for the *i*-th equality constraint
  - Solve the constrained optimization problem by finding the stationary point of the Lagrangian function

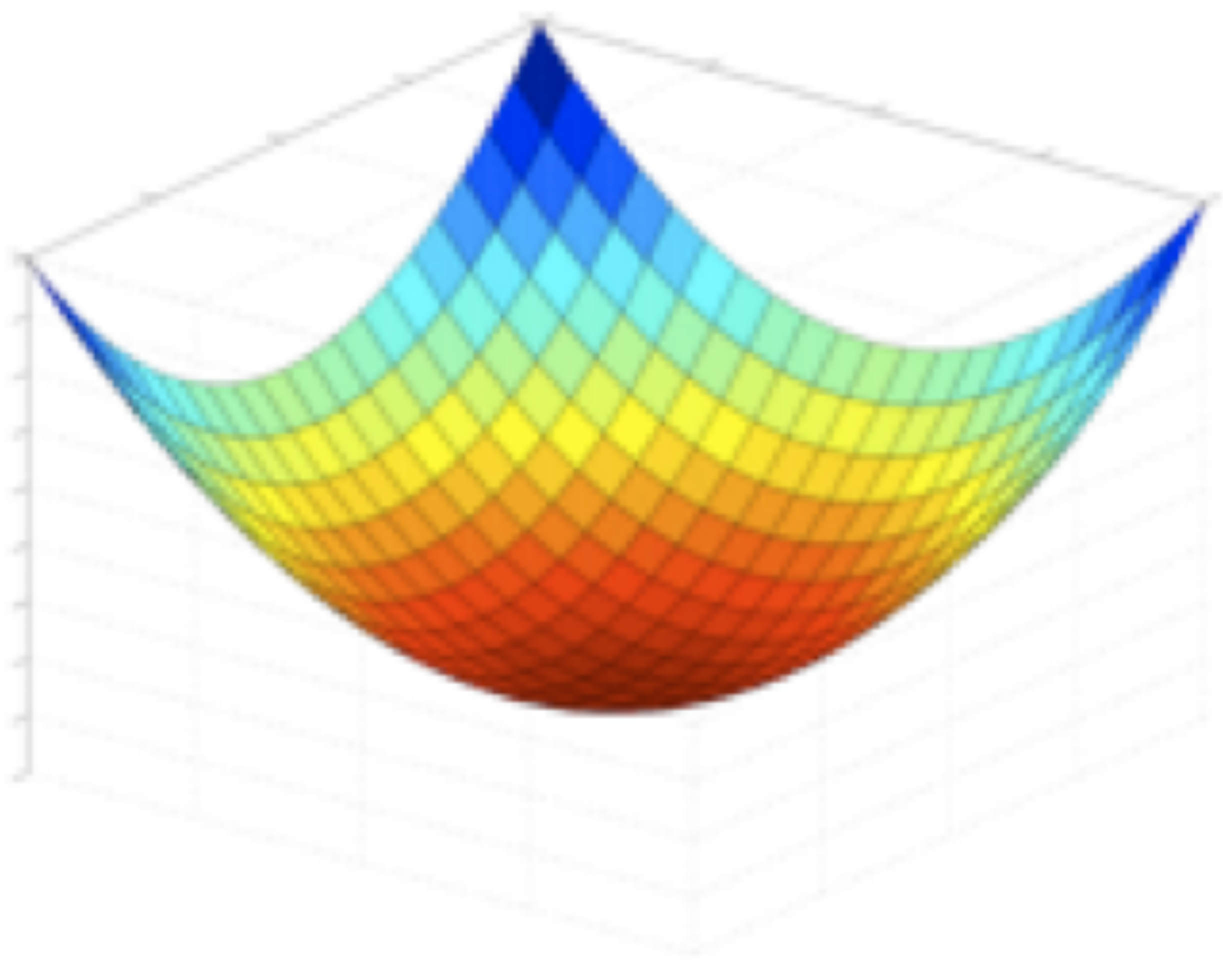
CONCAVE VS CONVEX

- Maximizing a concave function is equivalent to minimizing a convex function

concave



convex



NBC LEARNING REVISIT

- Maximize the log likelihood function
  - Likelihood: $L(\theta|D) = \prod_{i=1}^{n}\prod_{j=1}^{m} P(x_{ij}|c_i)P(c_i) = (\prod_{l=1}^{L} p_l^{N_l})(\prod_{l=1}^{L}\prod_{j=1}^{m}\prod_{k=1}^{K(j)} (q_l^{jk})^{N_l^{jk}})$
  - Log Likelihood: $logL(\theta|D) = \sum_{l=1}^{L} N_l log(p_l) + \sum_{l=1}^{L}\sum_{j=1}^{m}\sum_{k=1}^{K(j)} N_l^{jk} log q_l^{jk}$
  - Subject to constraints: $\sum_{l=1}^{L} p_l = 1, \sum_{k=1}^{K(j)} q_l^{jk} = 1$
- Lagrangian function:

$$L(p_l, q_l^{jk}, v_0, v_{lj}) = \sum_{l=1}^{L} N_l log(p_l) + \sum_{l=1}^{L}\sum_{j=1}^{m}\sum_{k=1}^{K(j)} N_l^{jk} log q_l^{jk} + v_0(\sum_{l=1}^{L} p_l - 1) + \sum_{l=1}^{L}\sum_{j=1}^{m} v_{lj}(\sum_{k=1}^{K(j)} q_l^{jk} - 1)$$

NBC LEARNING REVISIT

$$L(p_l, q_l^{jk}, v_0, v_{lj}) = \sum_{l=1}^{L} N_l log(p_l) + \sum_{l=1}^{L} \sum_{j=1}^{m} \sum_{k=1}^{K(j)} N_l^{jk} log q_l^{jk} + v_0 (\sum_{l=1}^{L} p_l - 1) + \sum_{l=1}^{L} \sum_{j=1}^{m} v_{lj} (\sum_{k=1}^{K(j)} q_l^{jk} - 1)$$

$$\Downarrow$$

$$\frac{N_l}{p_l} + v_0 = 0, \frac{N_l^{jk}}{q_l^{jk}} + v_{lj} = 0$$

$$\Downarrow$$

$$p_l = -\frac{N_l}{v_0}, q_l^{jk} = -\frac{N_l^{jk}}{v_{lj}} \quad + \quad \sum_{l=1}^{L} p_l = 1, \sum_{k=1}^{K(j)} q_l^{jk} = 1$$

$$\Downarrow$$

$$p_l = \frac{N_l}{N}, q_l^{jk} = \frac{N_l^{jk}}{N_l}$$

LOGISTIC REGRESSION LEARNING

- Logistic regression: $P(y=1|\mathbf{x}) = \dfrac{1}{1+e^{-(\mathbf{w}^T\mathbf{x}+w_0)}}$
  - Maximize (log) likelihood: $\mathbf{w} = (\mathbf{w}, w_0), \mathbf{x_i} = (\mathbf{x_i}, 1)$

$$
\begin{aligned}
logL(\mathbf{w}|D) &= \sum_{i=1}^{N} logp(y_i|\mathbf{w}) \\
&= \sum_{i=1}^{N} log[(\frac{1}{1+e^{-\mathbf{w}^T\mathbf{x_i}}})^{y_i}(\frac{e^{-\mathbf{w}^T\mathbf{x_i}}}{1+e^{-\mathbf{w}^T\mathbf{x_i}}})^{1-y_i}] \\
&= \sum_{i=1}^{N} (y_i\mathbf{w}^T\mathbf{x_i} - log(1+e^{\mathbf{w}^T\mathbf{x_i}}))
\end{aligned}
$$

  - Minimize: $\sum_{i=1}^{N} (-y_i\mathbf{w}^T\mathbf{x_i} + log(1+e^{\mathbf{w}^T\mathbf{x_i}}))$

LOGISTIC REGRESSION LEARNING

$$minimize \sum_{i=1}^{N} (-y_i \mathbf{w}^T \mathbf{x_i} + log(1 + e^{\mathbf{w}^T \mathbf{x_i}}))$$

$$\frac{dlogL(\mathbf{w}\,|\,D)}{dw_j} = \sum_{i=1}^{N} (-y_i x_{ij} + \frac{1}{1 + e^{\mathbf{w}^T \mathbf{x_i}}} e^{\mathbf{w}^T \mathbf{x_i}} x_{ij})$$

$$= \sum_{i=1}^{N} (-y_i + \frac{1}{1 + e^{\mathbf{w}^T \mathbf{x_i}}} e^{\mathbf{w}^T \mathbf{x_i}}) x_{ij}$$

$$= \sum_{i=1}^{N} (-y_i + P(y_i = 1\,|\,\mathbf{w})) x_{ij}$$

Convex!

But no closed form solution!

GRADIENT DESCENT

- For some convex functions, we may be able to take the derivative, but it may be difficult to directly solve for parameter values
- Solution:
  - Start at some value of the parameters
  - Take derivative and use it to move the parameters in the direction of the negative gradient
  - Repeat until stopping criteria is met (e.g., gradient close to 0)

Gradient Descent Rule:

$$\underline{\mathbf{w}}_{\text{new}} = \underline{\mathbf{w}}_{\text{old}} - \eta \, \Delta(\underline{\mathbf{w}})$$

where

$\Delta(\underline{w})$ is the gradient and

η is the learning rate (small, positive)

Notes:

1. This moves us downhill in direction $\Delta(\underline{\mathbf{w}})$ (steepest downhill direction)
2. How far we go is determined by the value of η