Data Mining & Machine Learning

CS37300
Purdue University

September 25, 2017

Searching over models/patterns

- Consider a **space** of possible models $M=\{M_1, M_2, ..., M_k\}$ with parameters θ
- Search could be over model structures or parameters, e.g.:
  - **Parameters**: In a linear regression model, what are regression coefficients (**β**) that minimize squared loss on the training data?
  - **Model structure**: In a decision trees, what is the tree structure that minimizes 0/1 loss on the training data?

Combinatorial optimization

Optimization

- **Non-smooth** functions:
    - If the function is *discrete*, then traditional optimization methods that rely on smoothness are not applicable (e.g., gradient descent needs the derivative). Instead we need to use **combinatorial optimization**
    - *Example: Choosing what features (structure) to add to a decision tree*

Search algorithms for discrete spaces

- Conduct the search by:
  - Considering a particular state (*model*)
  - Testing to see if it is the goal state (*model with maximum score*)
  - And if not, expand the current state to generate successor states by applying all possible actions
    (*determine alternative models to consider next*)
- Search strategies differ in their choice of how to expand states

Heuristic search

- Typically, there is an exponential number of models in the (discrete) search 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 other models near M and move to the best of these (if any have a score better than M)

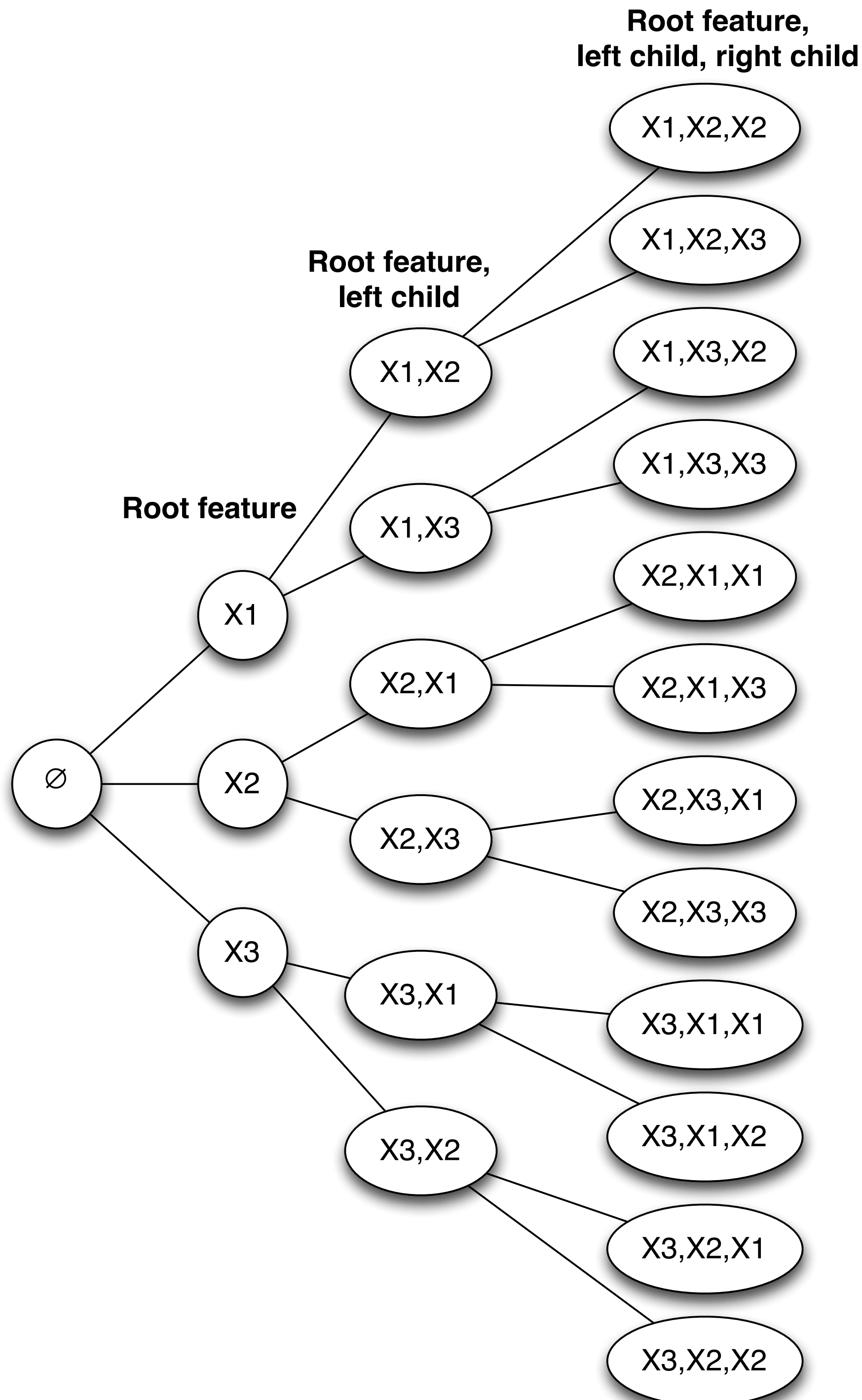
Greedy search

- Choose an initial state M^0 corresponding to a particular model structure (e.g., an empty tree)
- Let M^i be the model structure location 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 search 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

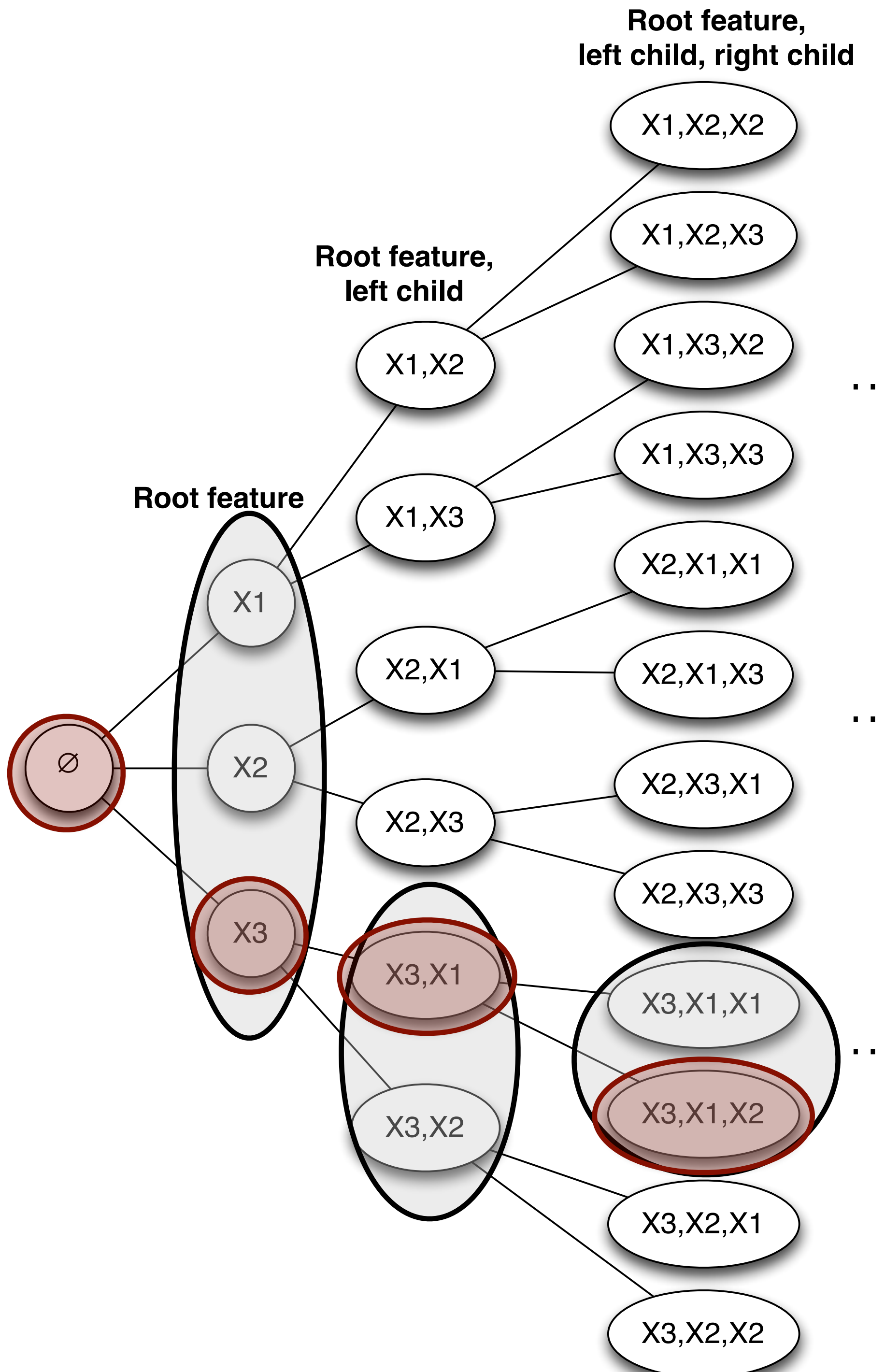
What is search space for decision-tree using data with three binary attributes X_1, X_2, X_3

Tree search algorithm

- Input:
  - Initial state? State space?
  - Set of actions?
  - How to choose next state?
  - Goal test?
- Output:
  - ?


Root feature,
left child, right child
Root feature,
left child
Root feature
∅
X1
X2
X3
X1,X2
X1,X3
X2,X1
X2,X3
X3,X1
X3,X2
X1,X2,X2
X1,X2,X3
X1,X3,X2
X1,X3,X3
X2,X1,X1
X2,X1,X3
X2,X3,X1
X2,X3,X3
X3,X1,X1
X3,X1,X2
X3,X2,X1
X3,X2,X2

Which states does greedy search consider?



Questions to ask about search procedures

- Is the search **exhaustive**?
  - I.e., does it either ***explicitly*** or ***implicitly*** consider all models in the space?
- Is the search **optimal**?
  - I.e., is it ***guaranteed*** to return the model with the best score?
  - Global vs. local optimum?

Smooth optimization

Optimization

- **Smooth** 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 solve the minimization problem in closed form: $\nabla S(\theta)$ using **convex optimization**
    - 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

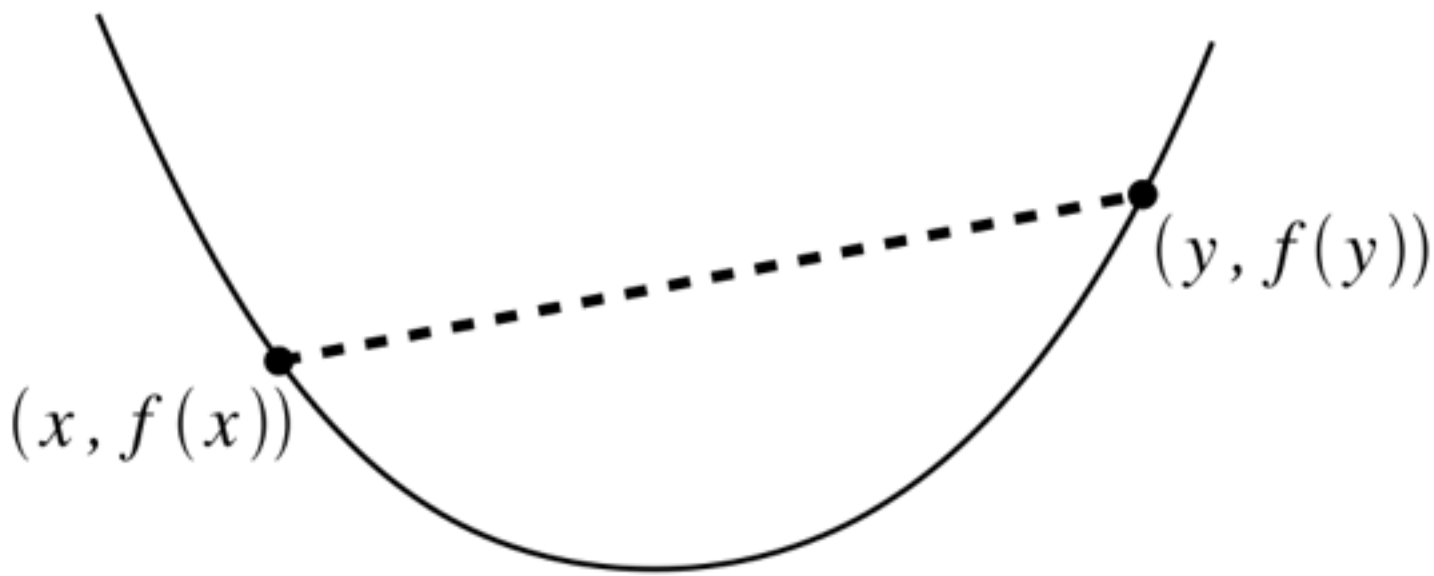
$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in C \end{array}$$

- Where ***f*** is a convex function (*score function*)
  C is a convex set (*constraints on model parameters or structure*)
  x is the optimization variable (*includes data and parameters*)
- For convex optimization problems, all locally optimal points are globally optimal
- Example algorithms: Quadratic programming (SVMs), least squares estimation, *maximum likelihood estimation*

Convex optimization

Convex functions

- In graph of convex function, the line connecting two points must lie above the function

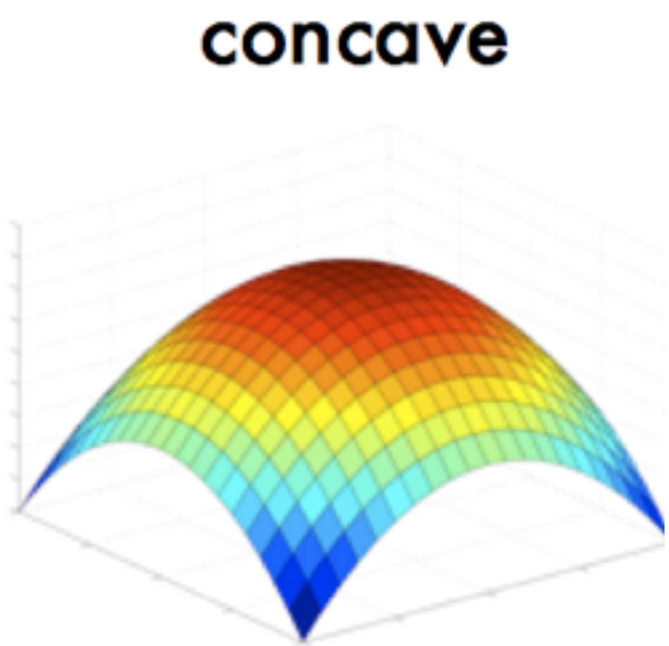


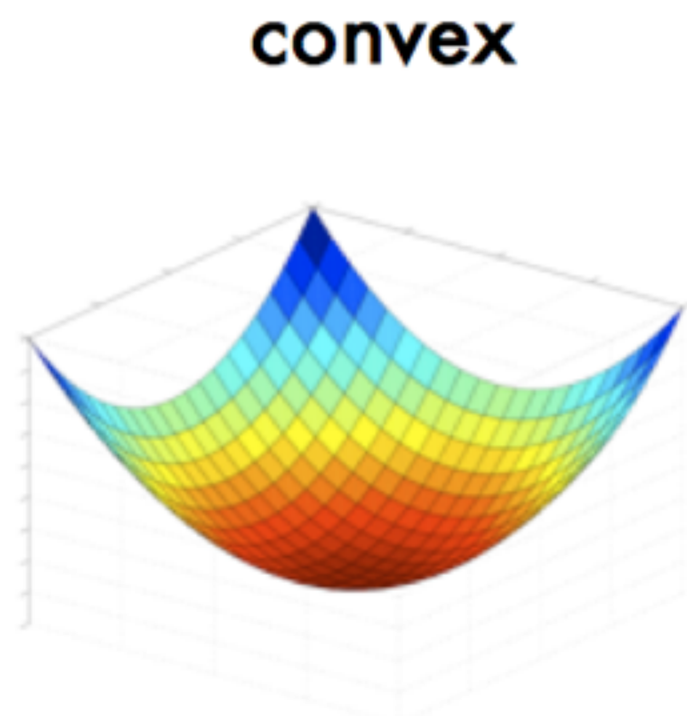
A function f is *convex* if:

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y) \text{ for all } 0 \leq \alpha \leq 1$$

Concave vs convex

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





Testing concavity (for maximization)

For a function f with parameters Θ

- Convexity conditions:

If f is continouous on Θ and $f''(\theta) \leq 0$ for all interior points θ

Then ***f*** is a strictly concave function

- Sufficient conditions:

If $f'(\theta) = 0$ then we say θ is a *stationary point* of f
If $f'(\theta) = 0$ and $f''(\theta) \leq 0$ then θ is a *local maximum* of f

If ***f*** is a strictly concave function, any stationary point of ***f*** is the unique global maximum of ***f***

Score function: Likelihood

- Let $D = \{x(1), ..., x(n)\}$
- Assume the data *D* are independently sampled from the same distribution:

$$p(X|\theta)$$

- The likelihood function represents the probability of the data as a function of the model parameters:

$$\begin{aligned} L(\theta|D) &= L(\theta|x(1), ..., x(n)) \\ &= p(x(1), ..., x(n)|\theta) \\ &= \prod_{i=1}^{n} p(x(i)|\theta) \end{aligned}$$

If instances are independent, likelihood is product of probs

Maximum likelihood estimation

- Most widely used method of parameter estimation
- "Learn" the best parameters by finding the values of θ that maximizes likelihood:

$$\hat{\theta}_{MLE} = \arg\max_{\theta} L(\theta)$$

- Often easier to work with loglikelihood:

$$\begin{aligned} l(\theta|D) &= log\, L(\theta|D) \\ &= log \prod_{i=1}^{n} p(x(i)|\theta) \\ &= \sum_{i=1}^{n} logp(x(i)|\theta) \end{aligned}$$

Maximum likelihood estimation

- Define likelihood, take derivative, set to 0, and solve
- Example
  - Toss a weighted coin 100 times, observe 30 heads
  - What is the MLE estimate for the *p* parameter of the Binomial distribution that generated the data?
  - First define likelihood

$$
\begin{aligned}
L(p|H=30, n=100) &= P(H=30|n=100, p) \\
&= \binom{100}{30} p^{30}(1-p)^{70}
\end{aligned}
$$

Example (cont')

Take derivative, set to 0, solve

$$
\begin{aligned}
0 &= \frac{d}{dp}\left(\binom{100}{30} p^{30}(1-p)^{70}\right) \\
&\propto 30p^{29}(1-p)^{70} - 70p^{30}(1-p)^{69} \\
&= p^{29}(1-p)^{69}[30(1-p) - 70p] \\
&= p^{29}(1-p)^{69}[30 - 100p] \\
p &= 0, 1, \frac{30}{100}
\end{aligned}
$$

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 solution
  - Repeat

<u>Gradient Descent Rule:</u>

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

where

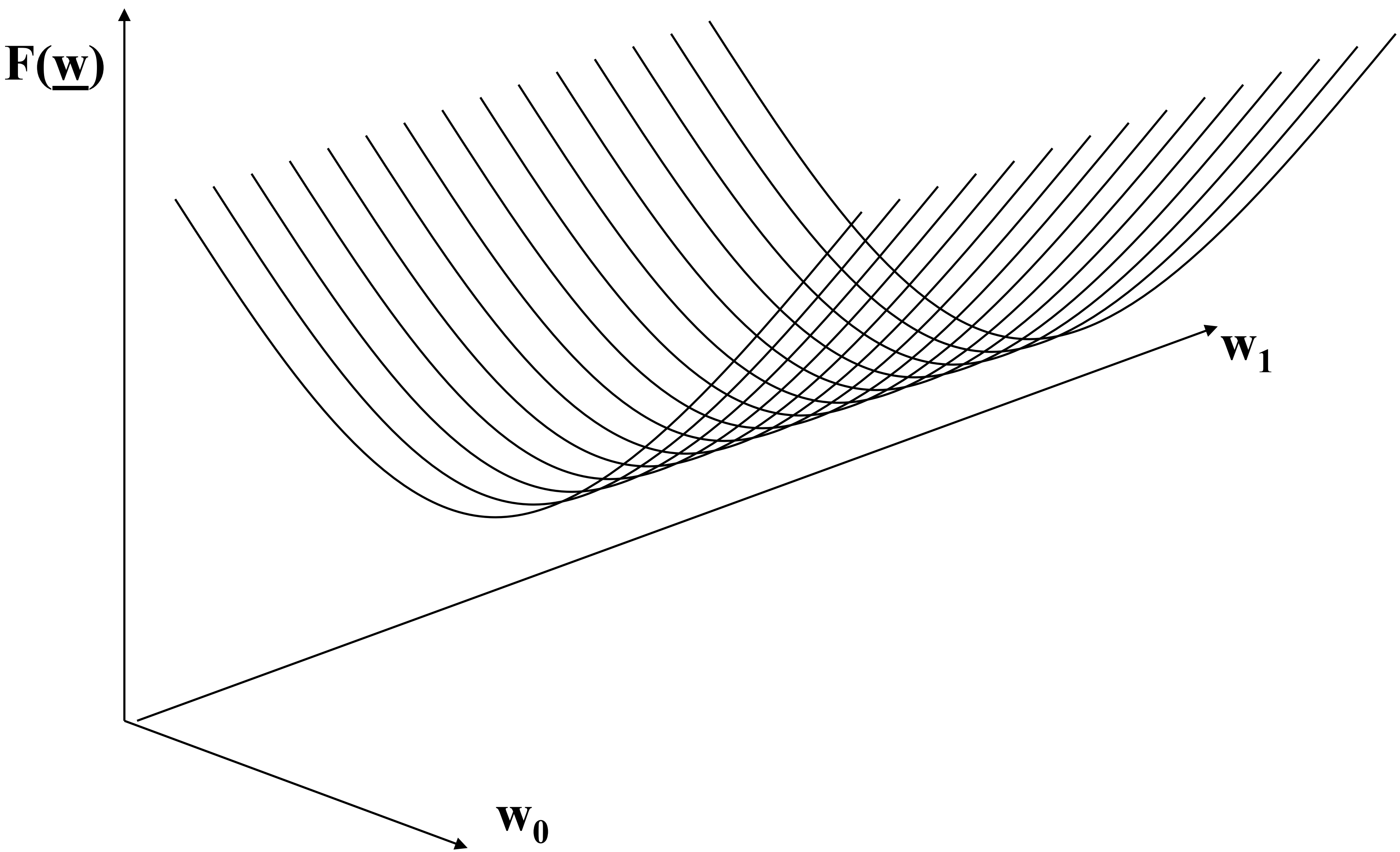
$\Delta(\underline{w})$ is the gradient and

η is the learning rate (small, positive)

Notes:

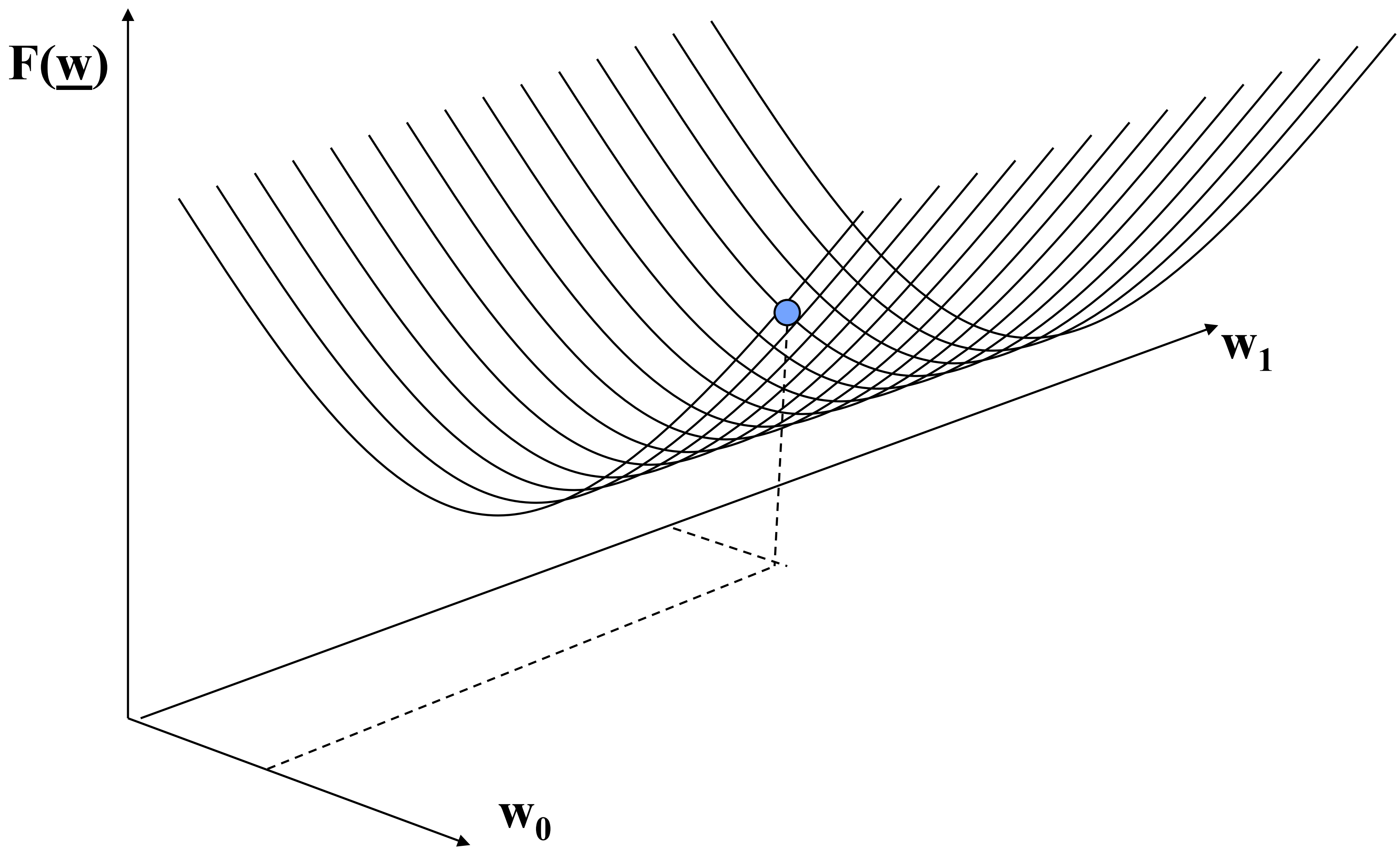
1. This moves us downhill in direction $\Delta(\underline{\mathbf{w}})$ (steepest downhill direction)
2. How far we go is determined by the value of η

Illustration of gradient descent



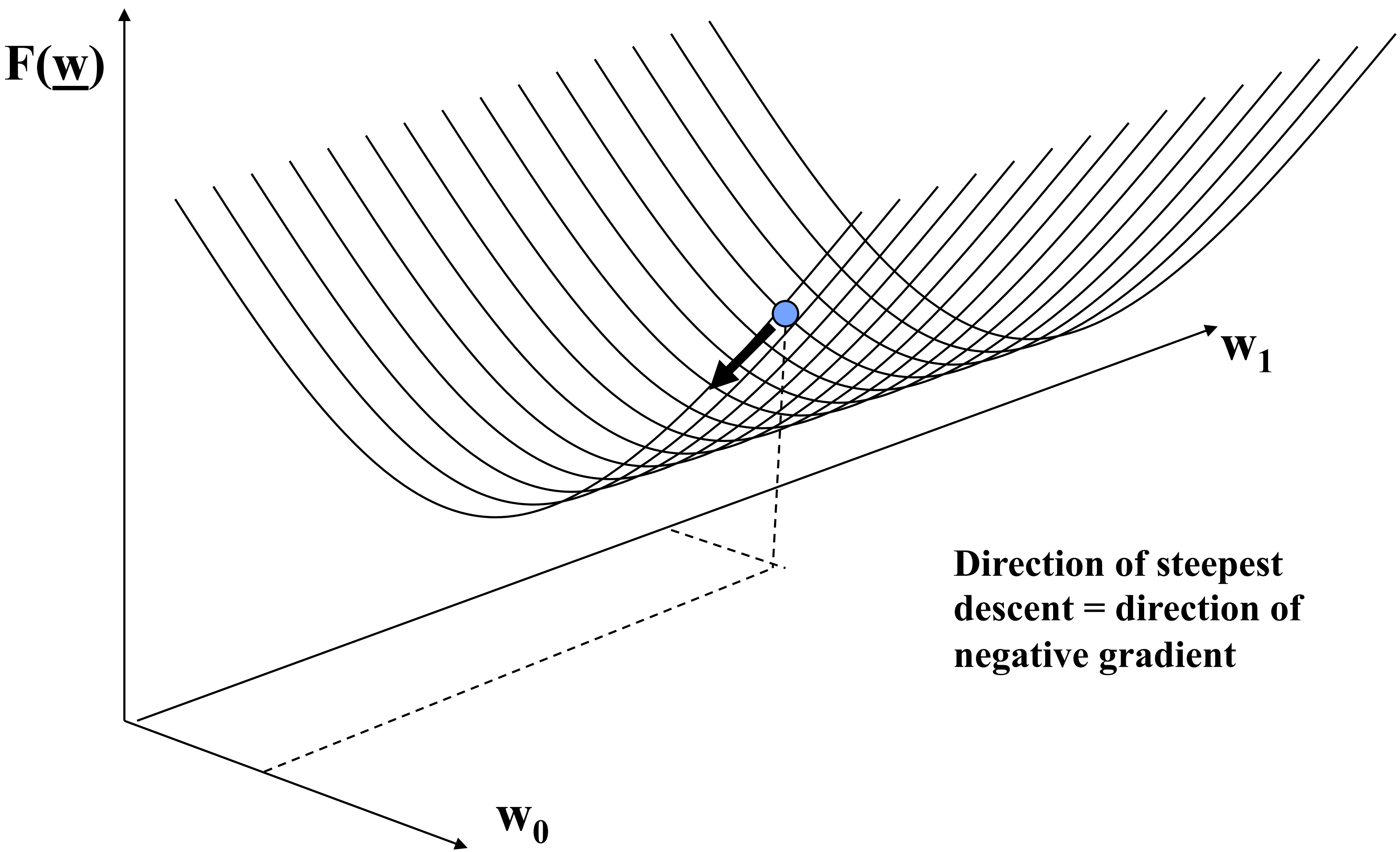
Slides adapted from CS175, UCIrvine, Padhraic Smyth

Illustration of gradient descent



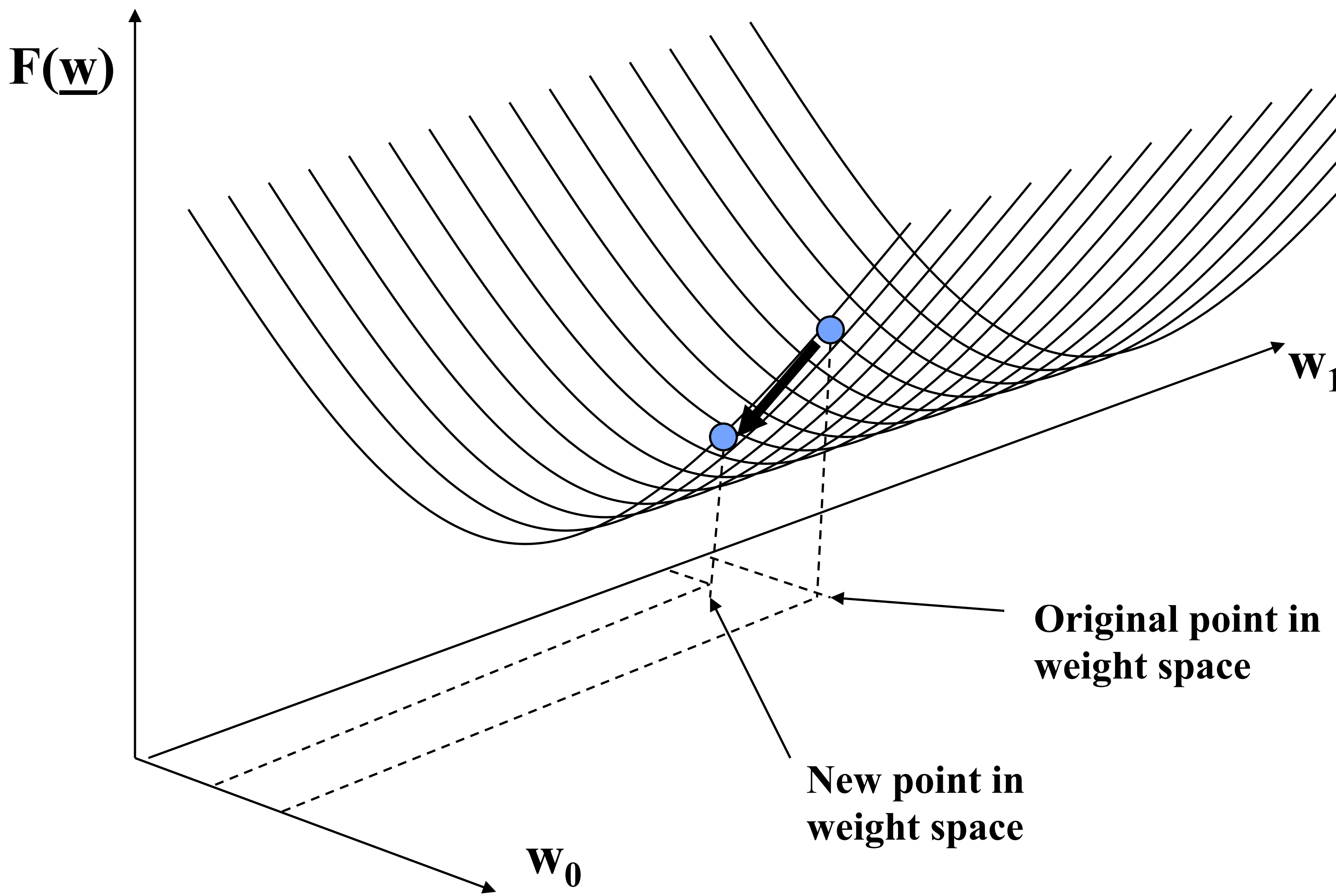
Slides adapted from CS175, UCIrvine, Padhraic Smyth

Illustration of gradient descent



Direction of steepest descent = direction of negative gradient

Illustration of gradient descent

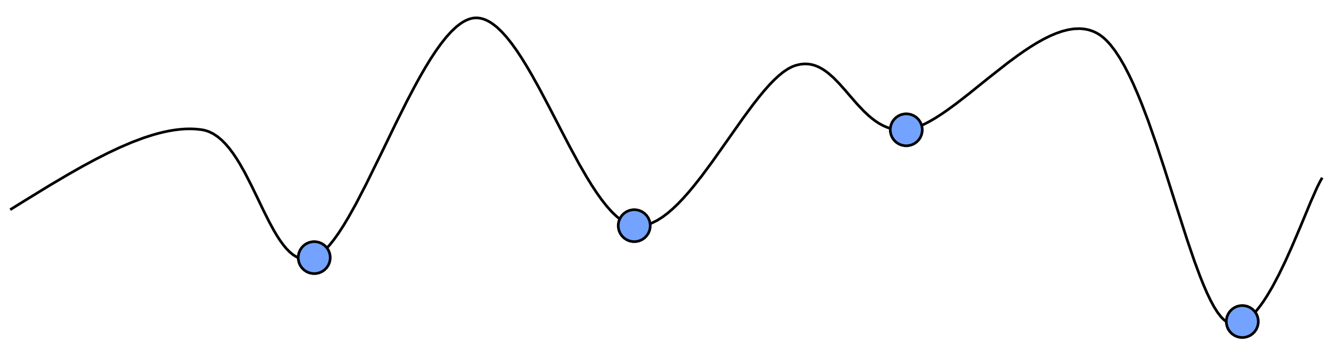


Slides adapted from CS175, UCIrvine, Padhraic Smyth

Non-convex optimization

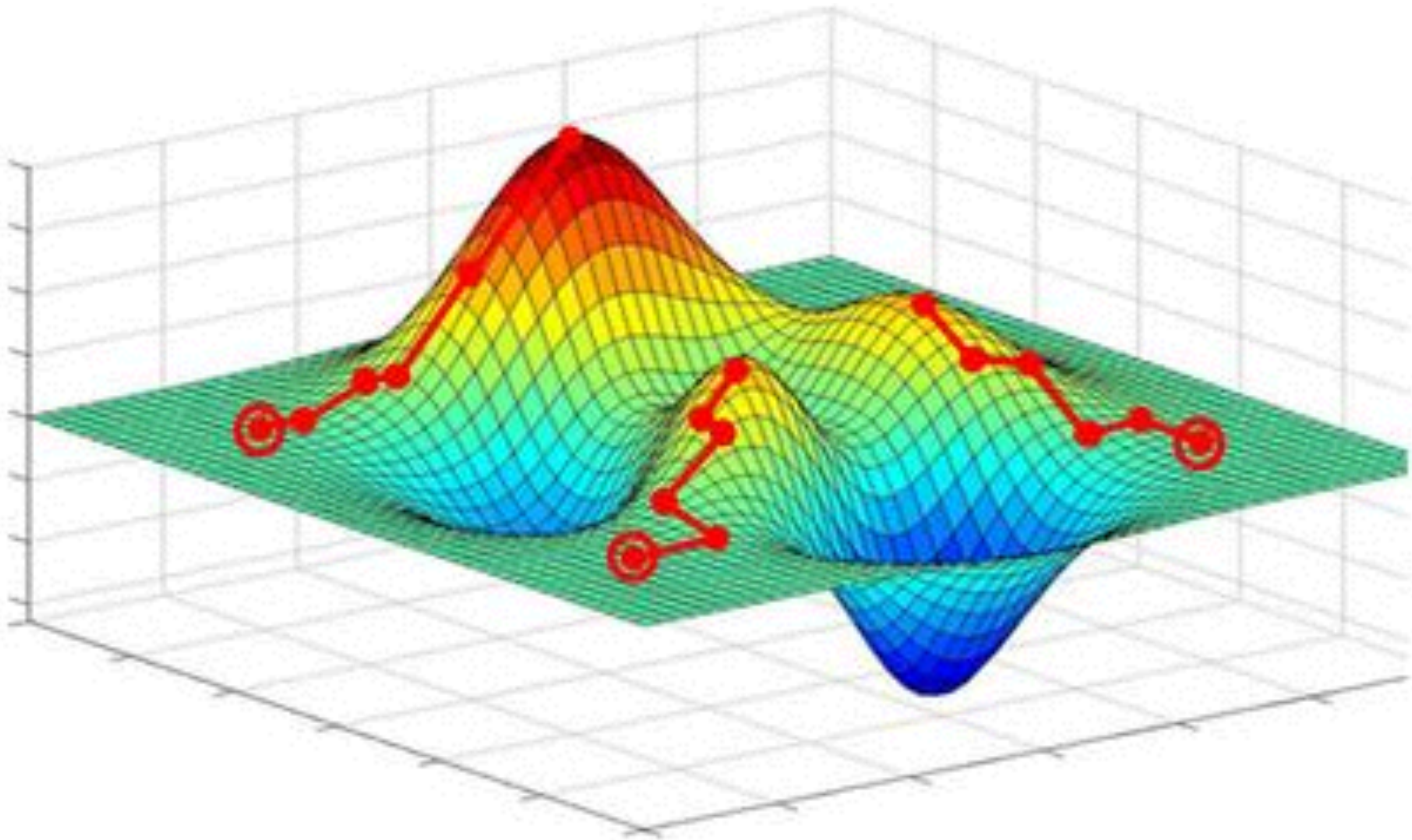
Gradient descent

- Works on any objective function F(w)
  - as long as we can evaluate the gradient Δ(w)
  - this can be very useful for minimizing complex functions F
- Can be used in hill-climbing search to find local minima in smooth, but non-convex functions



- If function has multiple local minima, gradient descent goes to the closest local minimum:
  - solution: random restarts from multiple places in weight space

Gradient ascent example



Graphic: http://www.forexartilect.com

Local search

- Iterative search that continually moves in direction of increasing value
  - Examples: gradient descent, hill climbing, coordinate descent
- Consider state-space landscape where:
  - Location=state; Elevation=**score function** for state