DSC478: Programming Machine Learning Applications

Roselyne Tchoua

rtchoua@depaul.edu

School of Computing, CDM, DePaul University

Linear Regression

- Linear regression: involves a response variable y and a single predictor variable x → $y = w_0 + w_1 x$
  - The weights w_0 (y-intercept) and w_1 (slope) are regression coefficients
- Method of least squares estimates the best-fitting straight line
  - w_0 and w_1 are obtained by minimizing the sum of the squared errors (a.k.a. residuals)

$$\boxed{\begin{aligned}\sum_i {e_i}^2 &= \sum_i (y_i - \hat{y}_i)^2 \\ &= \sum_i (y_i - (w_0 + w_1 x_i))^2\end{aligned}}$$

w_1 can be obtained by setting the partial derivative of the SSE to 0 and solving for w_1, ultimately resulting in:

$$\boxed{w_1 = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}} \qquad \boxed{w_0 = \bar{y} - w_1 \bar{x}}$$

Multiple Linear Regression

Multiple linear regression involves more than one predictor variable

- Features represented as $\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_d}$
- Training data is of the form $(\mathbf{x}^1, y^1), (\mathbf{x}^2, y^2), \ldots, (\mathbf{x}^n, y^n)$
  (each $\mathbf{x}^j$ is a row vector in matrix $\mathbf{X}$, i.e., a row in the data)
- For a specific value of a feature $\boldsymbol{x_i}$ in data item $\mathbf{x}^j$ we use: x_i^j
- Ex. For 2-D data, the regression function is: $\boxed{\hat{y} = w_0 + w_1 x_1 + w_2 x_2}$

$\boldsymbol{x_1}$ Living area (feet^2)	$\boldsymbol{x_2}$ #bedrooms	$\boldsymbol{y}$ Price (1000$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮

- More generally: $\boxed{\hat{y} = f(x_1, \ldots, x_d) = w_0 + \sum_{i=1}^{d} w_i x_i = w_0 + \mathbf{w}^{T}.\mathbf{x}}$

Least Squares Generalization

$$\boxed{\hat{y} = f(x_0, x_1, \ldots, x_d) = f(\mathbf{x}) = w_0 x_0 + \sum_{i=1}^{d} w_i x_i = \sum_{i=0}^{d} w_i x_i = \mathbf{w}^{\mathrm{T}}.\mathbf{x}}$$

Calculate the error function (SSE) and determine **w**:

$$E(\mathbf{w}) = (\mathbf{y} - f(\mathbf{x}))^2 = \left(\mathbf{y} - \sum_{i=0}^{d} w_i \cdot x_i\right)^2 = \sum_{j=1}^{n} (y^i - \sum_{i=0}^{d} w_i \cdot x_i^j)^2$$

$$= (\mathbf{y} - \mathbf{X}\mathbf{w})^{\mathrm{T}} \bullet (\mathbf{y} - \mathbf{X}\mathbf{w})$$

$\mathbf{y}$ = vector of all training responses y^j

$\mathbf{X}$ = matrix of all training samples $\mathbf{x}^j$

$$\boxed{\mathbf{w} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}}$$

Closed form solution to

$$\boxed{\frac{\partial}{\partial \mathbf{w}} E(\mathbf{w}) = 0}$$

$\hat{y}^{test} = \mathbf{w} \cdot \mathbf{x}^{test}$ for test sample $\mathbf{x}^{test}$

Gradient Descent Optimization

- Linear regression can also be solved using Gradient Decent (GD) optimization approach
- GD can be used in a variety of settings to find the minimum value of functions (including non-linear functions) where a closed form solution is not available or not easily obtained (too computationally expensive for example)

GD Optimization: Basic Idea

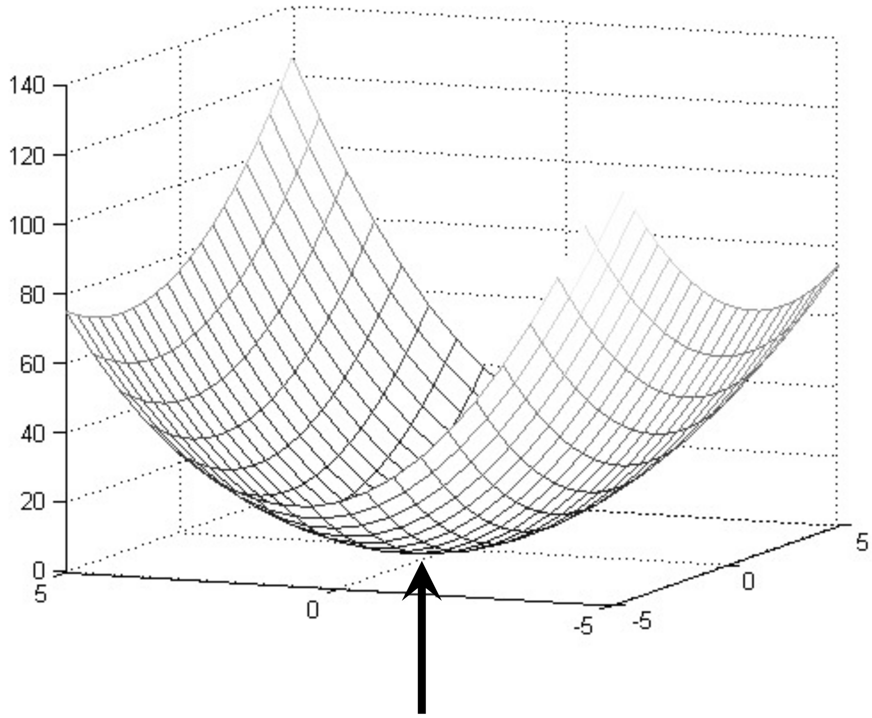
- Given an objective function $J(\mathbf{w})$ (e.g., sum of squared errors), with w as a vector of variables $w_0, w_1, \ldots, w_d$, iteratively minimize $J(\mathbf{w})$ by **finding the gradient** of the function surface in the variable-space and **adjusting the weights** in the opposite direction
- The gradient is a vector with each element representing the slope of the function in the direction of one of the variables
- Each element is the partial derivative of function with respect to one of variables

$$\nabla J(\mathbf{w}) = \nabla J(w_1, w_2, \ldots, w_d) = \left[\frac{\partial f(\mathbf{w})}{\partial w_1} \quad \frac{\partial f(\mathbf{w})}{\partial w_2} \quad \cdots \quad \frac{\partial f(\mathbf{w})}{\partial w_d} \right]$$

GD Optimization

- An example - quadratic function in 2 variables:

  f(**x**) =



- *f*(**x**) is minimum where gradient of f(**x**) is zero in all directions

GD Optimization

Gradient is a vector

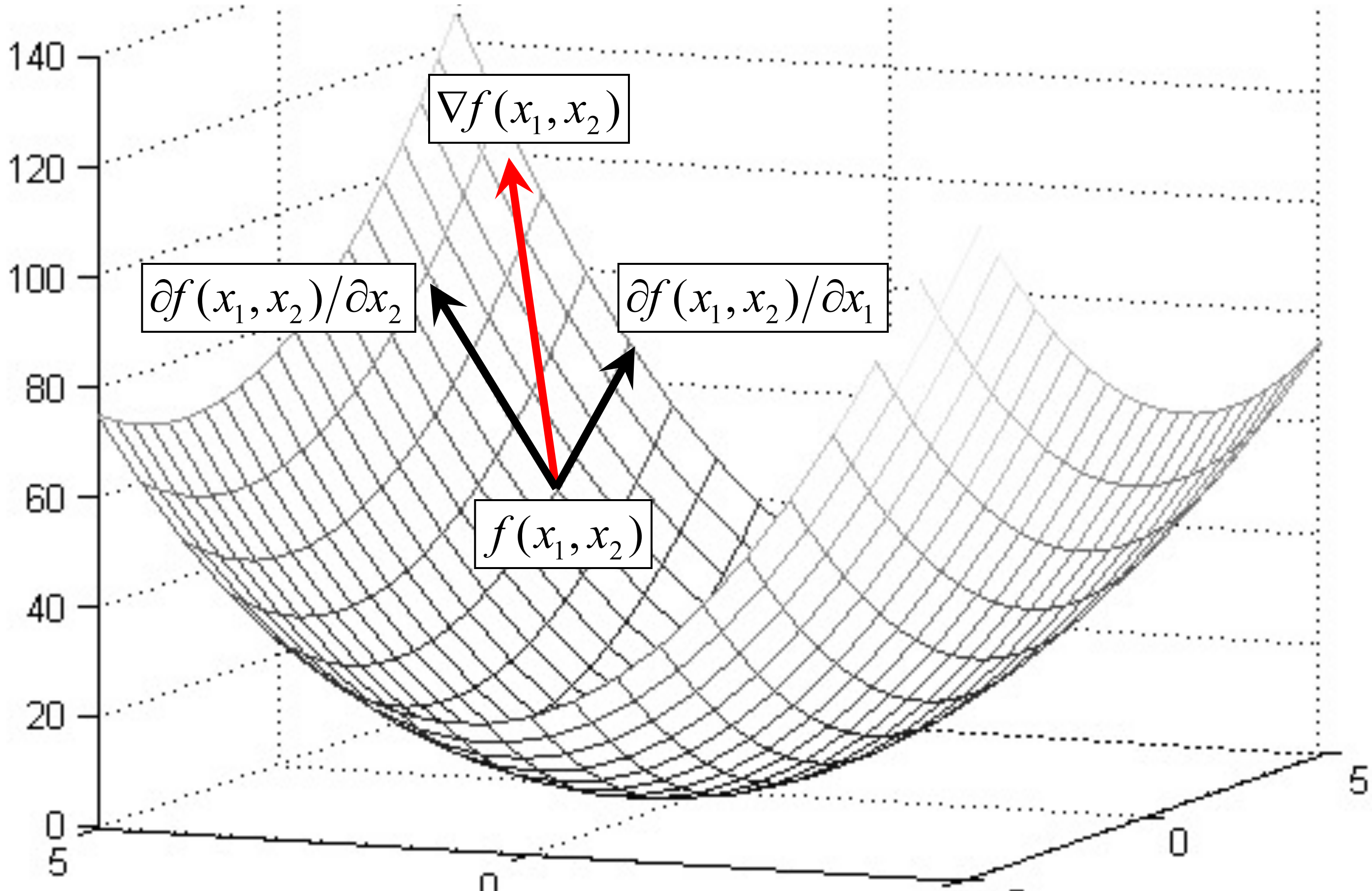
- Each element is the slope of function along direction of one of variables
- Each element is the partial derivative of function with respect to one of variables
- Example:

$$f(\mathbf{x}) = f(x_1, x_2) = x_1^{\ 2} + x_1 x_2 + 3x_2^{\ 2}$$

$$\nabla f(\mathbf{x}) = \nabla f(x_1, x_2) = \left[\frac{\partial f(\mathbf{x})}{\partial x_1} \quad \frac{\partial f(\mathbf{x})}{\partial x_2} \right] = \left[2x_1 + x_2 \quad x_1 + 6x_2 \right]$$

GD Optimization

Gradient vector points in direction of steepest ascent of function



GD Optimization

This two-variable example is still simple enough that we can find minimum directly

$$f(x_1, x_2) = x_1^{\ 2} + x_1 x_2 + 3x_2^{\ 2}$$
$$\nabla f(x_1, x_2) = \begin{bmatrix} 2x_1 + x_2 & x_1 + 6x_2 \end{bmatrix}$$

- Set both elements of gradient to 0
- Gives two linear equations in two variables
- Solve for x_1, x_2

$$2x_1 + x_2 = 0 \qquad\qquad x_1 + 6x_2 = 0$$
$$x_1 = 0 \qquad\qquad x_2 = 0$$

GD Optimization

Finding minimum directly by closed form analytical solution often difficult or impossible

- Quadratic functions in many variables
  - system of equations for partial derivatives may be ill-conditioned
  - example: linear least squares fit where redundancy among features is high
- Other convex functions
  - global minimum exists, but there is no closed form solution
  - example: maximum likelihood solution for logistic regression
- Nonlinear functions
  - partial derivatives are not linear
  - example: $f(x_1, x_2) = x_1(\sin(x_1 x_2)) + x_2^2$
  - example: sum of transfer functions in neural networks

GD Optimization

- Given an objective (e.g., error) function E(w) = E(w0, w1, …, wd)
- Process (follow the gradient downhill):
  1. Pick an initial set of weights (random) w = (w0, w1, …, wd)
  2. Determine the descent direction: $-\nabla E$ (wt)
  3. Choose a learning rate: η
  4. Update your position: $\mathbf{w}^{t+1} = \mathbf{w}^{t} - \eta \cdot \nabla E(\mathbf{w}^{t})$
  5. Repeat from 2) until stopping criterion is satisfied

Note: this step involves simultaneous updating of each weight w_i

- Typical stopping criteria
  - $\nabla E(\mathbf{w}^{t+1}) \sim 0$
  - some validation metric is optimized

GD Optimization

- In Least Squares Regression: $E(\mathbf{w}) = \left(\mathbf{y} - \sum_{i=0}^{d} w_i \cdot x_i \right)^2 = (\mathbf{y} - \mathbf{w}^{\mathrm{T}}.\mathbf{x})^2$

- Process (follow the gradient *downhill*):
  1. Select initial $\mathbf{w} = (w_0, w_1, \ldots, w_d)$
  2. Compute $-\nabla E(\mathbf{w})$
  3. Set η
  4. Update: $\mathbf{w} := \mathbf{w} - \eta \cdot \nabla E(\mathbf{w})$
  5. Repeat until $\nabla E(\mathbf{w}^{t+1}) \sim 0$

$$w_j := w_j - \eta \frac{1}{2n} \sum_{i=1}^{n} (\mathbf{w}^{\mathrm{T}}.\mathbf{x}^i - y^i) x_j^i$$
$$\text{for } j = 0, 1, \ldots, d$$

Illustration of GD Descent

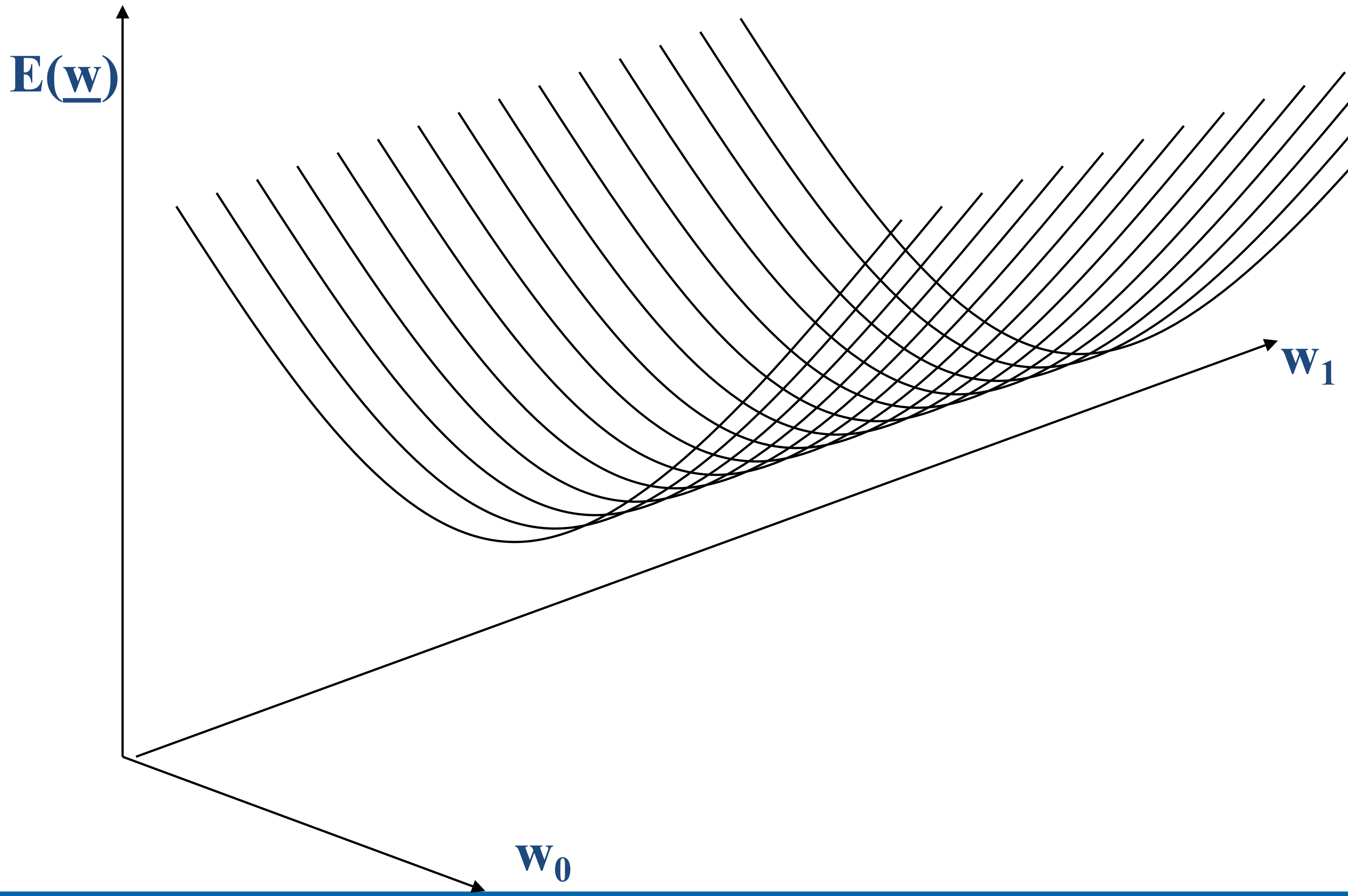


Illustration of Gradient Descent

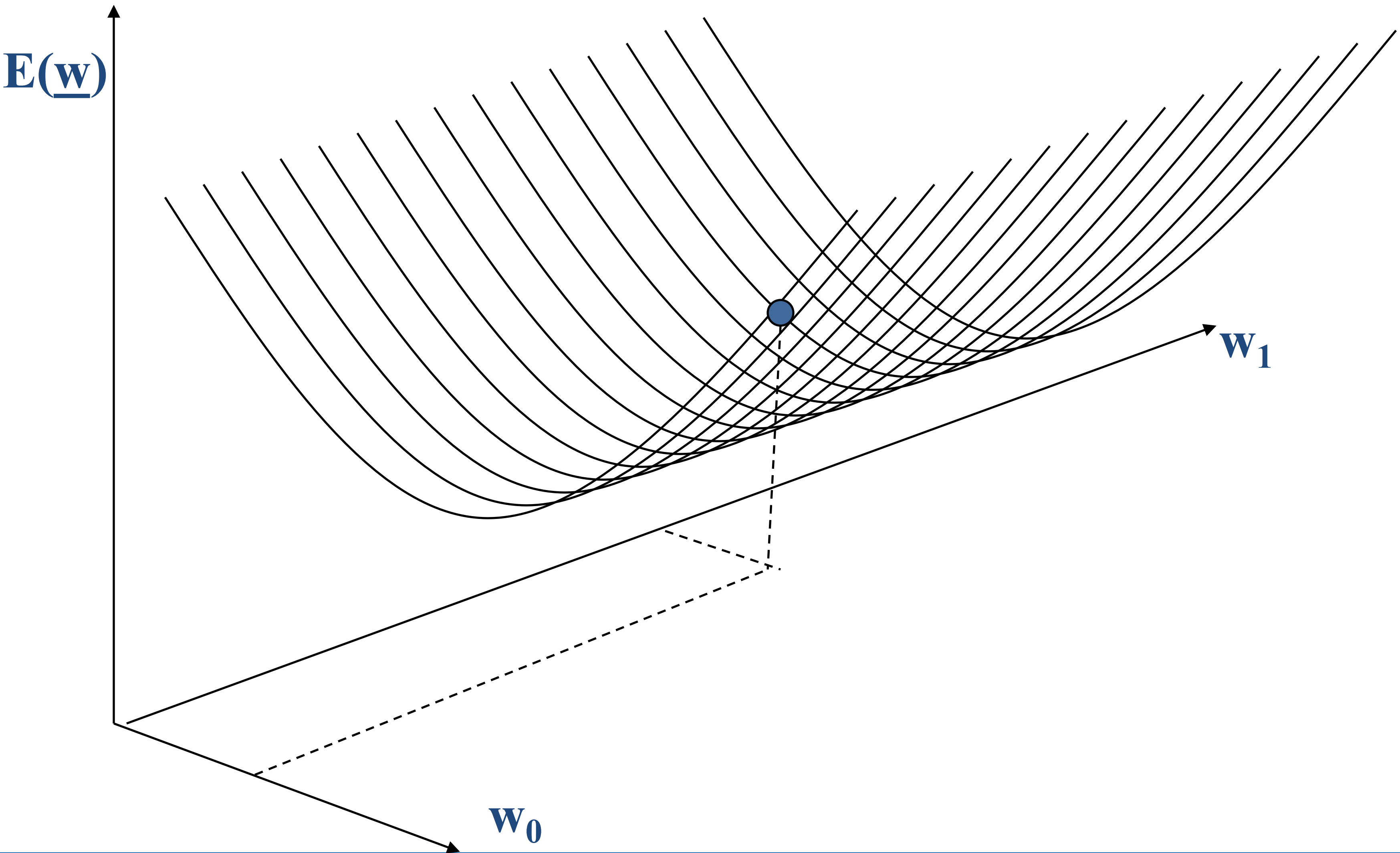
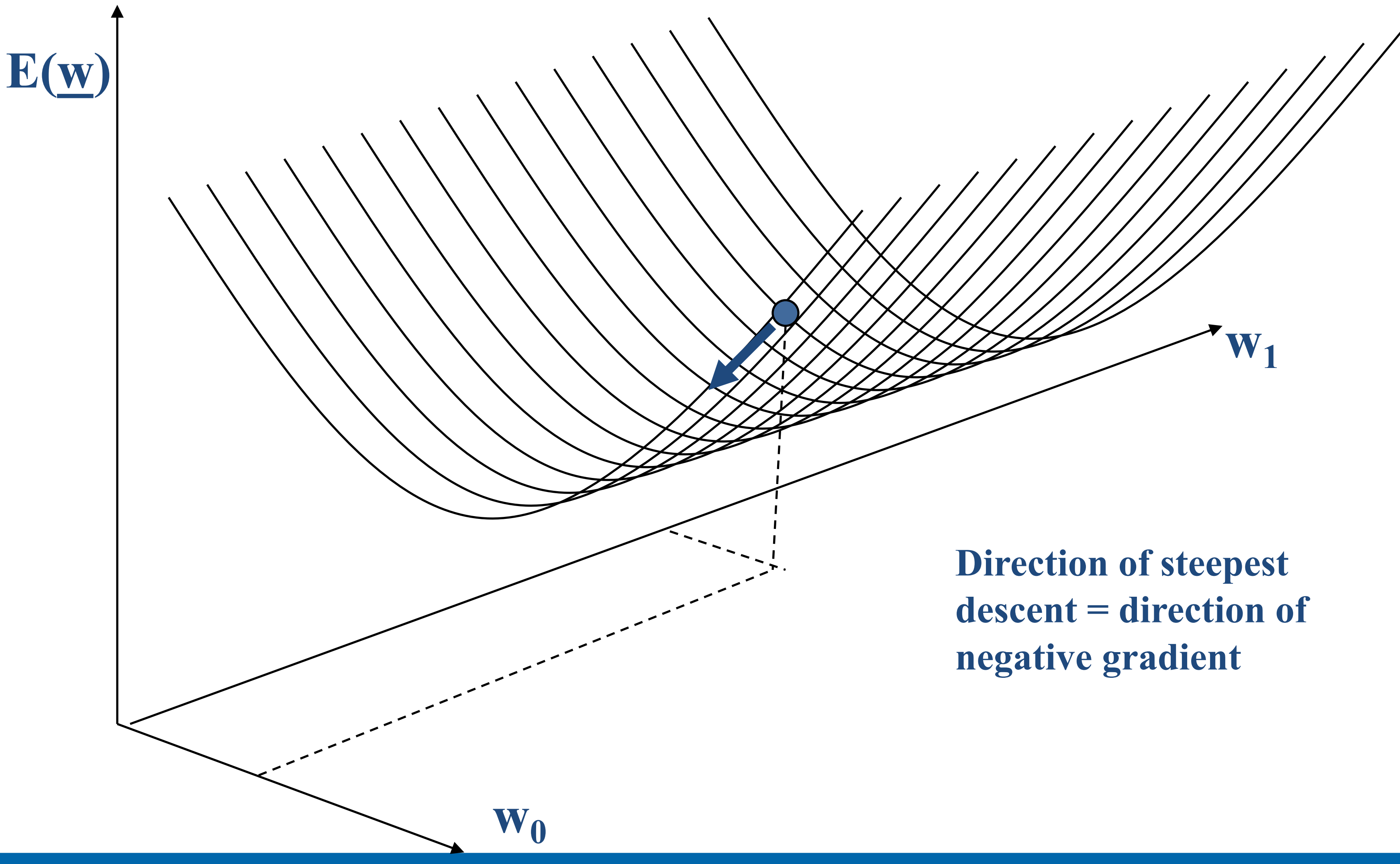
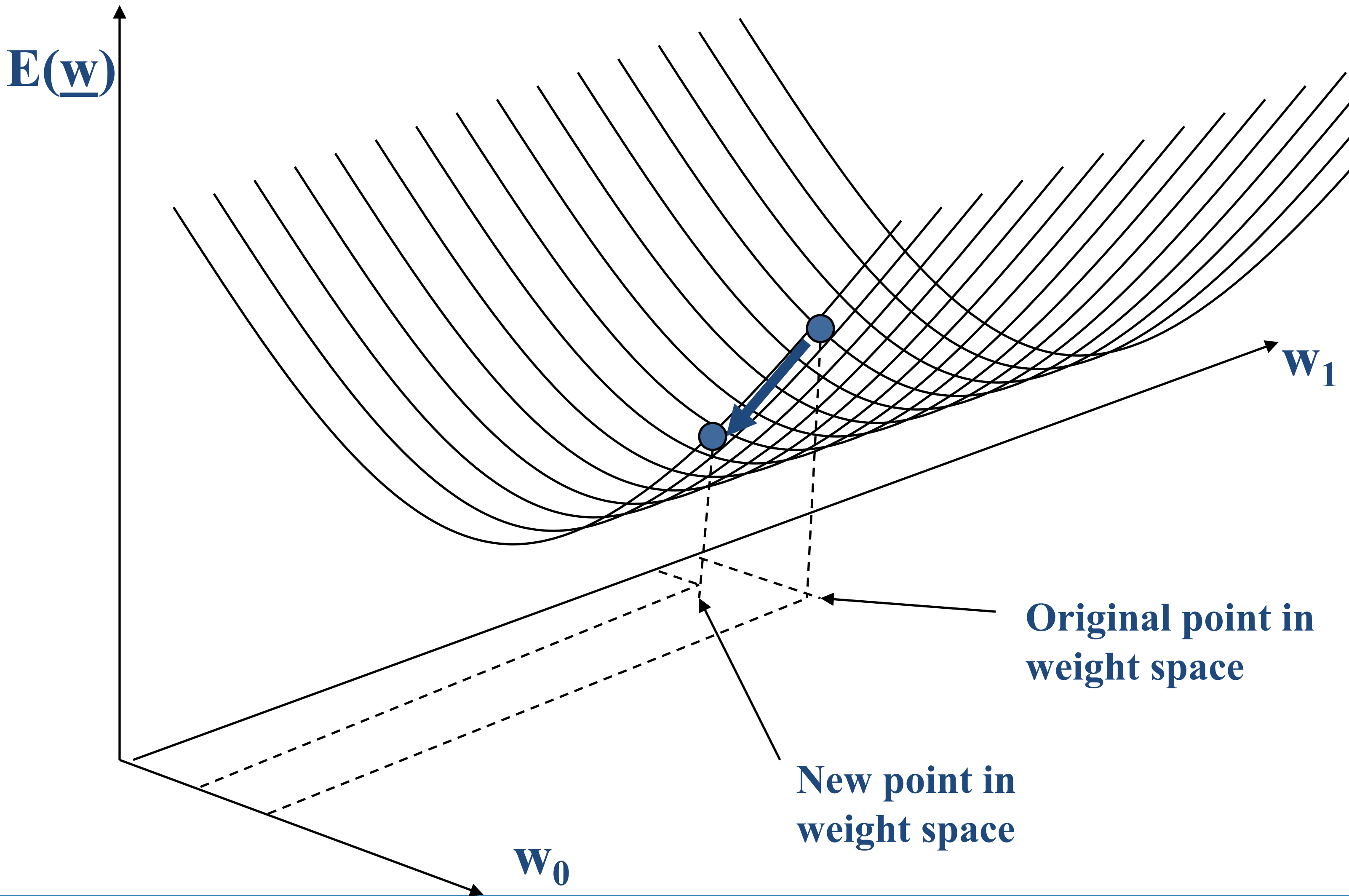


Illustration of Gradient Descent



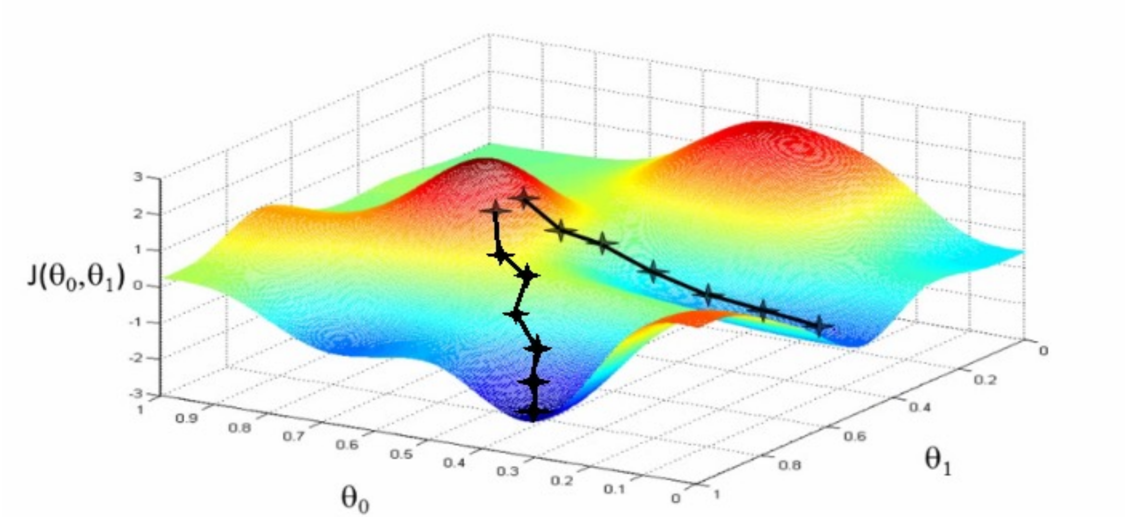
Direction of steepest descent = direction of negative gradient

Illustration of Gradient Descent



GD Optimization Problems

- Choosing step size (learning rate)
  - too small $\rightarrow$ convergence is slow and inefficient
  - too large $\rightarrow$ may not converge
- Can get stuck on “flat” areas of function
- Easily trapped in local minima



Stochastic GD

- Application to training a machine learning model:
  1. Choose one sample from training set: x^i
  2. Calculate objective function for that single sample:
  3. Calculate gradient from objective function:
  4. Update model parameters a single step based on gradient and learning rate:

$$\boxed{w_j := w_j - \eta(\mathbf{w}^{\mathrm{T}}.\mathbf{x}^i - y^i)x_j^i \quad \text{for } j = 0,\ldots,d}$$

  5. Repeat from 1) until stopping criterion is satisfied
- Typically, entire training set is processed multiple times before stopping
- Order in which samples are processed can be fixed or random