CSE 575
Statistical Machine Learning

Lecture 6
YooJung Choi

Fall 2022

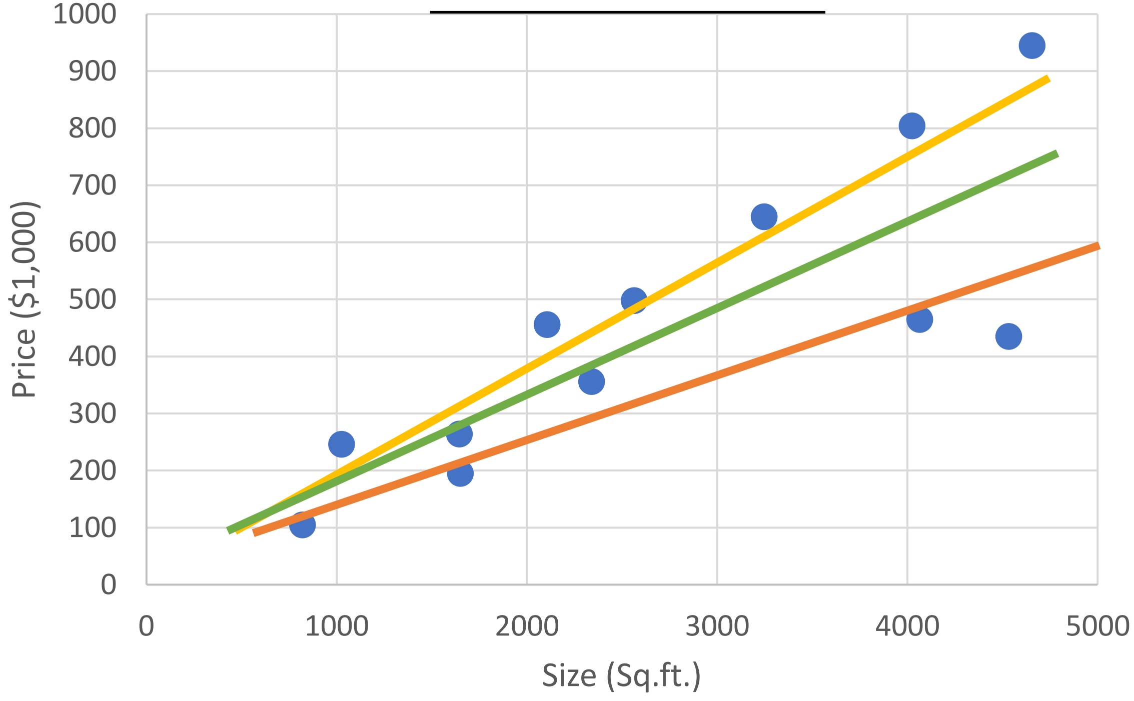
Supervised learning

- Given: a training data set $\{(\mathbf{x}_n, t_n)\}_{n=1}^{N}$ of inputs $\mathbf{x}$ and target value t
- Goal: learn a model that predicts the value of t for a new $\mathbf{x}$
- *Regression* if t is continuous (e.g. $t \in \mathbb{R}$)
- *Classification* if t is discrete / categorical (e.g. $t \in \{0,1\}$, $t \in \{\text{dog, cat, bird}\}$)

$$\mathbf{x} \longrightarrow \boxed{y} \longrightarrow t$$

Linear regression

Size (Sq.ft.)	Price ($1,000)
821	105
1645	264
⋮	⋮
2106	456



$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{D} w_j x_j$$

$$y(\text{size}, \mathbf{w}) = w_0 + w_1 \cdot \text{size}$$

We can also have multiple features: e.g.

$$y(\text{size}, \#\text{rooms}, \mathbf{w})$$
$$= w_0 + w_1 \cdot \text{size} + \mathrm{w}_2 \cdot \#rooms$$

Linear basis function models

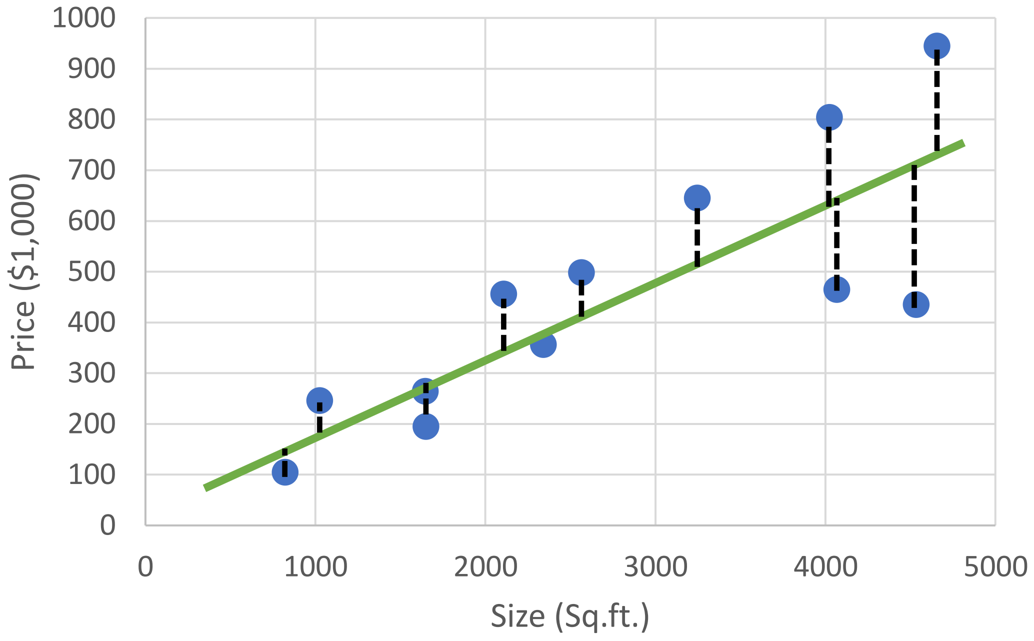
- Fit a function of the form:

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})$$

  where $\phi_j(\mathbf{x})$ are the *basis functions* ($\phi_0(\mathbf{x}) = 1$)

- E.g. polynomial curve fitting: $\phi_j(x) = x^j$
- *Linear* in the parameters $\mathbf{w}$
- More on this later. For now, assume $\boldsymbol{\phi}(\mathbf{x}) = \mathbf{x}$ with a dummy feature $x_0 = 1$

Linear regression



$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{D} w_j x_j = \mathbf{w}^T \mathbf{x}$$

Sum of squares error

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} (y(\mathbf{x}_n, \mathbf{w}) - t_n)^2 = \frac{1}{2} \sum_{n=1}^{N} (\mathbf{w}^T \mathbf{x}_\mathrm{n} - t_n)^2$$

Least squares

Set the derivative to 0 and solve for **w**:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}_\mathrm{n} - t_n)^2 = \frac{1}{2}(\mathbf{Xw}-\mathbf{t})^\mathrm{T}(\mathbf{Xw}-\mathbf{t}) = \frac{1}{2}(\mathbf{w}^T\mathbf{X}^T\mathbf{Xw} - 2\mathbf{w}^T\mathbf{X}^T\mathbf{t} + \mathbf{t}^T\mathbf{t})$$

$$\nabla E(\mathbf{w}) = -\mathbf{X}^T\mathbf{t} + \mathbf{X}^T\mathbf{Xw} = 0 \quad \Rightarrow \quad \mathbf{X}^T\mathbf{Xw} = \mathbf{X}^T\mathbf{t}$$ *"Normal Equation"*

$$\mathbf{X} = \begin{bmatrix} - & \mathbf{x}_1^T & - \\ & \vdots & \\ - & \mathbf{x}_N^T & - \end{bmatrix}, \qquad \mathbf{w} = \begin{bmatrix} w_0 \\ \vdots \\ w_D \end{bmatrix}, \qquad \mathbf{Xw} = \begin{bmatrix} \mathbf{w}^T\mathbf{x}_1 \\ \vdots \\ \mathbf{w}^T\mathbf{x}_N \end{bmatrix}$$

Size (Sq.ft.)	Price ($1,000)
821	105
1645	264
⋮	⋮
2106	456

$$\mathbf{X} = \begin{bmatrix} 1 & 821 \\ \vdots & \vdots \\ 1 & 2106 \end{bmatrix}$$

$$\nabla E(\mathbf{w}) = \begin{bmatrix} \frac{\partial E}{\partial w_0} \\ \vdots \\ \frac{\partial E}{\partial w_D} \end{bmatrix}$$

$$f(\mathbf{w}) = w_0^2 + w_0 w_1$$

$$\nabla f(\mathbf{w}) = \begin{bmatrix} 2w_0 + w_1 \\ w_0 \end{bmatrix}$$

Normal Equation

$$\mathbf{X}^T\mathbf{X}\mathbf{w} = \mathbf{X}^T\mathbf{t}$$

If $\mathbf{X}^T\mathbf{X}$ is invertible, unique solution $\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{t}$ (often when $N \gg D$)

Otherwise (e.g. if $N < D$), infinitely many solutions for $\mathbf{w}^*$ (we can use the pseudo-inverse)

To compute the closed-form solution $\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{t}$:

- Compute $\mathbf{X}^T\mathbf{t}$: $O(ND)$ time
- Compute $\mathbf{X}^T\mathbf{X}$: $O(ND^2)$ time
- Compute $(\mathbf{X}^T\mathbf{X})^{-1}$: $O(D^3)$ time
- Compute $(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{t}$: $O(D^2)$ time
- Total time $O(ND^2 + D^3)$

When D is large, this is too expensive!

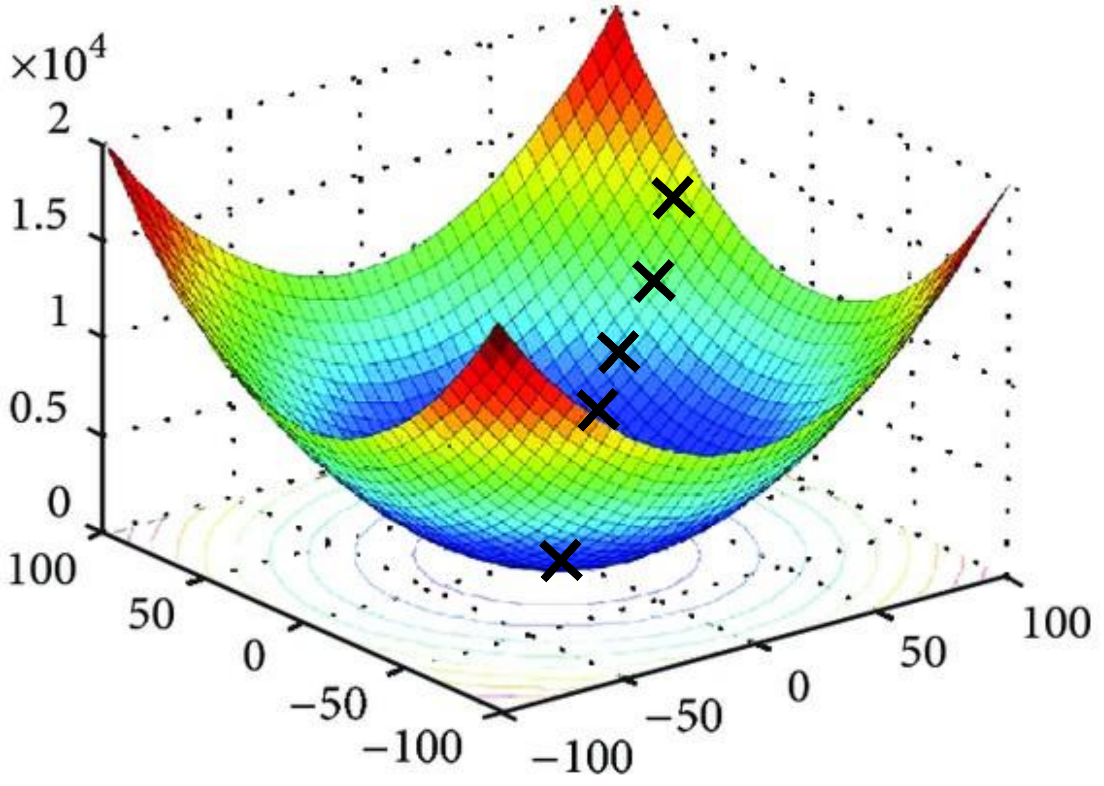
Gradient descent

- Iteratively update parameters in the direction of negative gradient:

Learning rate

$$\mathbf{w}^{(k+1)} \leftarrow \mathbf{w}^{(k)} - \eta \nabla_{\mathbf{w}} E(\mathbf{w}^{(k)})$$

- For a convex *E*, converges to the global minimum



Gradient descent for least squares

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N} (\mathbf{w}^T \mathbf{x}_{\mathrm{n}} - t_n)^2 , \qquad \nabla E(\mathbf{w}) = \mathbf{X}^T \mathbf{X}\mathbf{w} - \mathbf{X}^T \mathbf{t}$$

$$\mathbf{w}^{(k+1)} \leftarrow \mathbf{w}^{(k)} - \eta \cdot \mathbf{X}^T (\mathbf{X}\mathbf{w}^{(k)} - \mathbf{t})$$

Size (Sq.ft.)	Price ($1,000)
821	105
1645	264
⋮	⋮
2106	456

$$\mathbf{X} = \begin{bmatrix} 1 & 821 \\ \vdots & \vdots \\ 1 & 2106 \end{bmatrix}, \ \mathbf{t} = \begin{bmatrix} 105 \\ \vdots \\ 456 \end{bmatrix}$$

$$\mathbf{w}^{(k)} = \begin{bmatrix} 10 \\ 0.2 \end{bmatrix}$$

$$\mathbf{X}\mathbf{w}^{(k)} = \begin{bmatrix} 174.2 \\ \vdots \\ 431.2 \end{bmatrix} = \begin{bmatrix} \mathbf{w}^{(\mathrm{k})\mathrm{T}} \mathbf{x}_1 \\ \vdots \\ \mathbf{w}^{(\mathrm{k})\mathrm{T}} \mathbf{x}_N \end{bmatrix}$$

$$\mathbf{X}\mathbf{w}^{(k)} - \mathbf{t} = \begin{bmatrix} 69.2 \\ \vdots \\ -24.8 \end{bmatrix}$$

$$\mathbf{X}^T (\mathbf{X}\mathbf{w}^{(k)} - \mathbf{t}) = 69.2 \begin{bmatrix} 1 \\ 821 \end{bmatrix} + \cdots - 24.8 \begin{bmatrix} 1 \\ 2106 \end{bmatrix}$$

Gradient descent for least squares

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}_\mathrm{n} - t_n)^2, \qquad \nabla E(\mathbf{w}) = -\mathbf{X}^T\mathbf{t} + \mathbf{X}^T\mathbf{X}\mathbf{w}$$

$$\begin{aligned}\mathbf{w}^{(k+1)} &\leftarrow \mathbf{w}^{(k)} - \eta \cdot \mathbf{X}^T(\mathbf{X}\mathbf{w}^{(k)} - \mathbf{t}) \\ &= \mathbf{w}^{(k)} - \eta\sum_{n=1}^{N}\left(\mathbf{w}^{(k)T}\mathbf{x}_\mathrm{n} - t_n\right)\mathbf{x}_\mathrm{n}\end{aligned}$$

Each update takes $O(ND)$ time

Works well even for large D

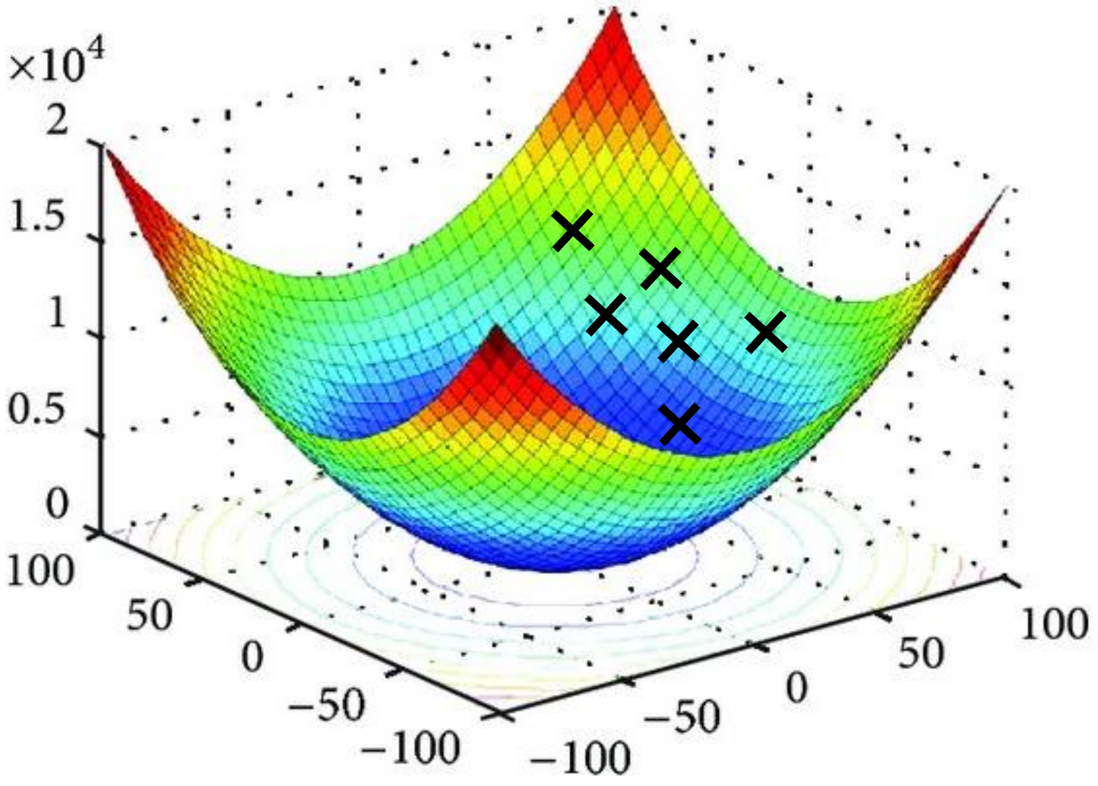
But N could be extremely large!

Stochastic gradient descent

- Iteratively update parameters using one example at a time

For $\mathrm{n} = 1, \ldots, N$:

$$\mathbf{w}^{(k+1)} \leftarrow \mathbf{w}^{(k)} - \eta \cdot \left(\mathbf{w}^{(k)T}\mathbf{x}_{\mathrm{n}} - t_n\right)\mathbf{x}_{\mathrm{n}}$$



Learning algorithms

- Normal Equation:
  - Closed-form solution
  - Slow if D is large
  - Matrix inversion is expensive

- Gradient Descent:
  - Works well with large D
  - May take many iterations
  - Need to choose learning rate η
  - Each update is slow if N is large

- Stochastic Gradient Descent:
  - Each iteration is fast even for large N
  - May take even more iterations

Linear basis function models

- Fit a function of the form:

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})$$

where $\phi_j(\mathbf{x})$ are the *basis functions* ($\phi_0(\mathbf{x}) = 1$)

- E.g. polynomial curve fitting: $\phi_j(x) = x^j$
- Many possible choices for basis functions (Gaussian, sigmoidal, ...)
- $y(\mathbf{x}, \mathbf{w})$ can be non-linear in $\mathbf{x}$ but is still linear in $\mathbf{w}$, simplifying analysis

Linear basis function models

- Simply replace $\mathbf{x}$ and $\mathbf{X}$ with:

$$\boldsymbol{\phi}(\mathbf{x}) = \begin{bmatrix} 1 \\ \phi_1(\mathbf{x}) \\ \vdots \\ \phi_M(\mathbf{x}) \end{bmatrix}, \qquad \boldsymbol{\Phi} = \begin{bmatrix} - \boldsymbol{\phi}^T(\mathbf{x_1}) - \\ \vdots \\ - \boldsymbol{\phi}^{\mathrm{T}}(\mathbf{x_N}) - \end{bmatrix}$$

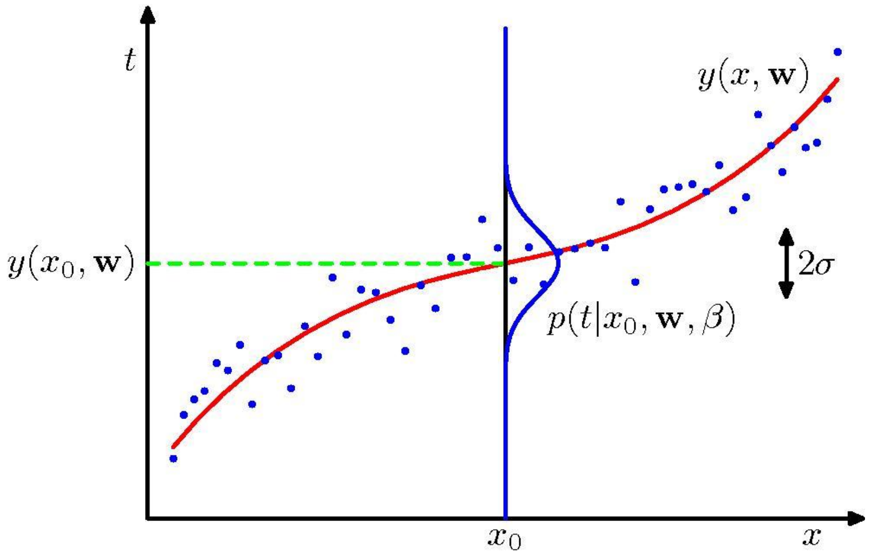
- Note: D-dimensional inputs $\mathbf{x}$ to M-dimensional features $\boldsymbol{\phi}(\mathbf{x})$

- Limitation: fixed basis functions
- Future topic: using adaptive basis functions (e.g. neural networks)

MLE and least squares: Review

- Alternative view: add an additive Gaussian noise term ϵ with zero mean and precision β

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon$$



- Learn the predictive distribution

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1})$$

- MLE: given *N* samples $\{(\mathbf{x}_n, t_n)\}_{n=1,\ldots,N}$,

maximize $\ln p(\mathbf{t}|\mathbf{w}, \beta) = \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \beta E(\mathbf{w})$

Equivalent to minimizing the sum-of-squares error

Regularized least squares

- Recall: L2-regularized error function

$$\tilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}_\mathrm{n} - t_n)^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

- Penalize large coefficients

- We can still find the minimizer in a closed form:

$$\mathbf{w}^* = (\lambda\mathbf{I} + \mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{t}$$

- Also called *ridge regression*

MAP and regularized least squares: Review

- In addition to the Gaussian noise ϵ in $t = y(\mathbf{x}, \mathbf{w}) + \epsilon$, assume a Gaussian prior over the parameters $\mathbf{w}$

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$$

- Maximum posterior estimate for $\mathbf{w}$:

$$\text{maximize } \ln(p(\mathbf{t}|\boldsymbol{w}, \beta) \times p(\boldsymbol{w}|\alpha)) = C - \beta E(\mathbf{w}) - \frac{\alpha}{2}\mathbf{w}^T\mathbf{w}$$

Equivalent to minimizing the L2-regularized sum-of-squares error