Machine Learning I: Supervised Methods

B. Keith Jenkins

slido.com → Slido event code: 3629199
Join as participant

Announcements

- Homework 2 is due Friday.

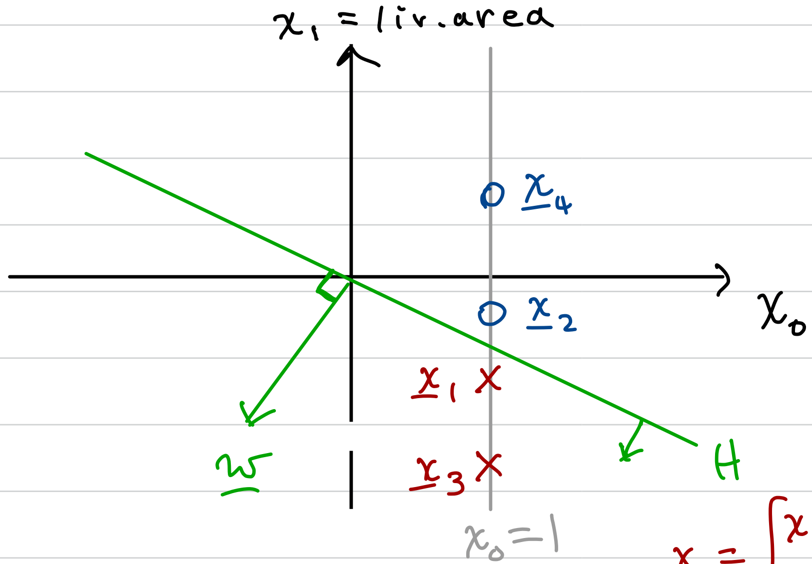
Reading

- Bishop 3.0, 3.1 (MSE regression)
- Bishop 4.1.3 (MSE classification)

Today's lecture

- Vector and feature-space representations (part 2)
  - Finish weight space
  - Reflected data points
- The learning problem
  - Problem statement and approach
  - Criterion functions
  - Optimization approaches
  - Gradient descent
- Perceptron learning algorithm
  - Perceptron criterion function
  - Perceptron learning (batch GD)
  - Perceptron learning (sequential GD)

Feature space (augmented) $D=1$ feature



× S_1 price decr.

o S_2 price incr.

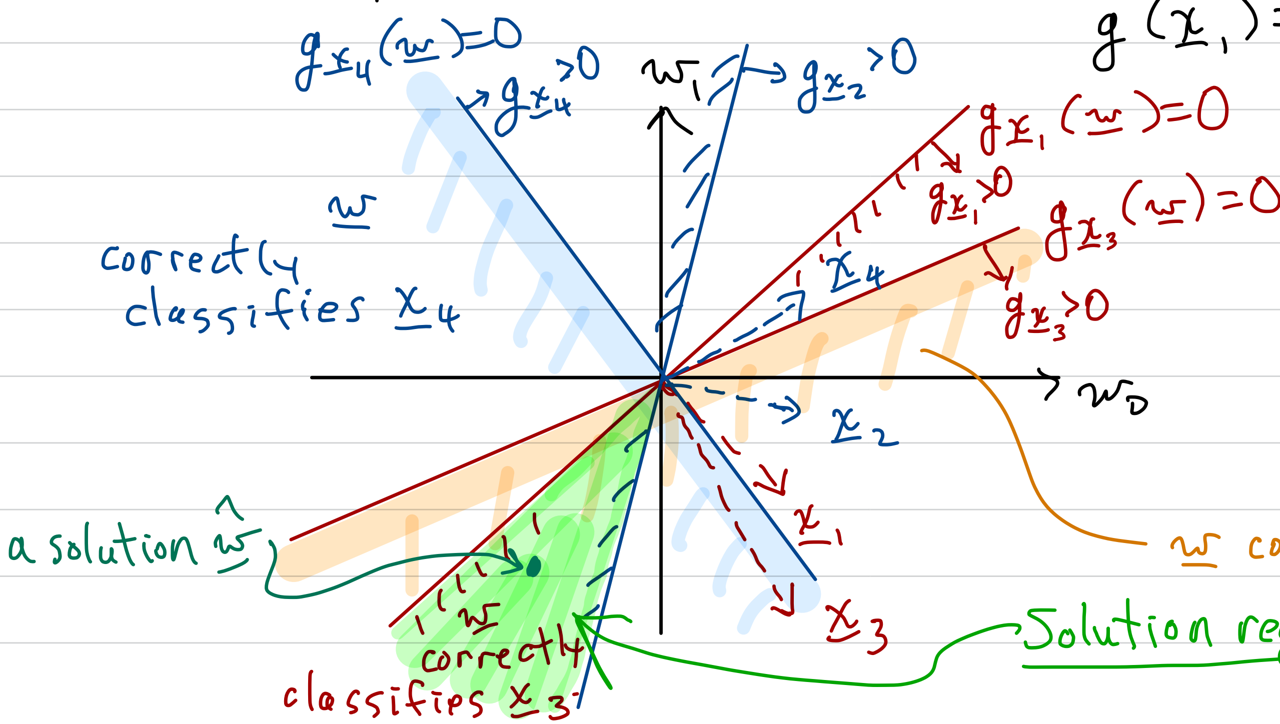
$$\underline{x}_1 = \begin{bmatrix} x_{10} \\ x_{11} \end{bmatrix}$$

$$H: \quad g(\underline{x}) = 0 = \underline{w}^T \underline{x}$$

$$H: \quad w_0 x_0 + w_1 x_1 = 0$$

coefficients variables

Weight space (augm.)



$$g(\underline{x}_1) = g_{\underline{x}_1}(\underline{w}) = x_{10} w_0 + x_{11} w_1$$

coefficients variables

$$g_{\underline{x}_1}(\underline{w}) = 0 \Rightarrow \text{line}$$

($\underline{w}$ for which $\underline{w}^T \underline{x}_1 = 0$)

$\underline{w}$ correctly classifies $\underline{x}_1$.

Solution region – all $\underline{w}$ correctly classify $\underline{x}_1, \underline{x}_2, \underline{x}_3, \underline{x}_4$.

Slido:
w^T means w^T
x^(2) means $x^{(2)}$

slido.com
3629199

Reflected data points (only for $C=2$)

→ augmented notation

→ linear discriminant fcns. $g(\underline{x})$ (can be generalized to nonlinear – later)

$$g(\underline{x}) = \underline{w}^T \underline{x} \underset{\Gamma_2}{\overset{\Gamma_1}{\gtrless}} 0$$

For any labelled data pts. $\underline{x}^{(1)} \in S_1$, $\underline{x}^{(2)} \in S_2$:

(1)
$$\begin{cases} \underline{w}^T \underline{x}^{(1)} > 0 \Rightarrow \underline{x}^{(1)} \text{ correctly classified} & (<0 \Rightarrow \text{incorrect}) \\ \underline{w}^T \underline{x}^{(2)} < 0 \Rightarrow \underline{x}^{(2)} \text{ correctly classified} & (>0 \Rightarrow \text{incorrect}) \end{cases}$$

Let $z_n^{(k)} = z_n \triangleq \begin{Bmatrix} +1, & k=1 & (S_1) \\ -1, & k=2 & (S_2) \end{Bmatrix}$ = class indicator

then consider $\underline{w}^T z_n \underline{x}_n$ >0 when? <0 when?

(2)
$$\boxed{\begin{cases} \underline{w}^T z_n \underline{x}_n > 0 \Rightarrow \underline{x}_n \text{ is correctly classified} \\ (<0 \Rightarrow \text{incorrect}) \end{cases}}$$

$z_n \underline{x}_n$ is the reflected data point. ($\underline{x}_n$ is the unreflected data pt.)

More Notation

Reflected data points (2-class problems, augmented space, n^{th} data point):

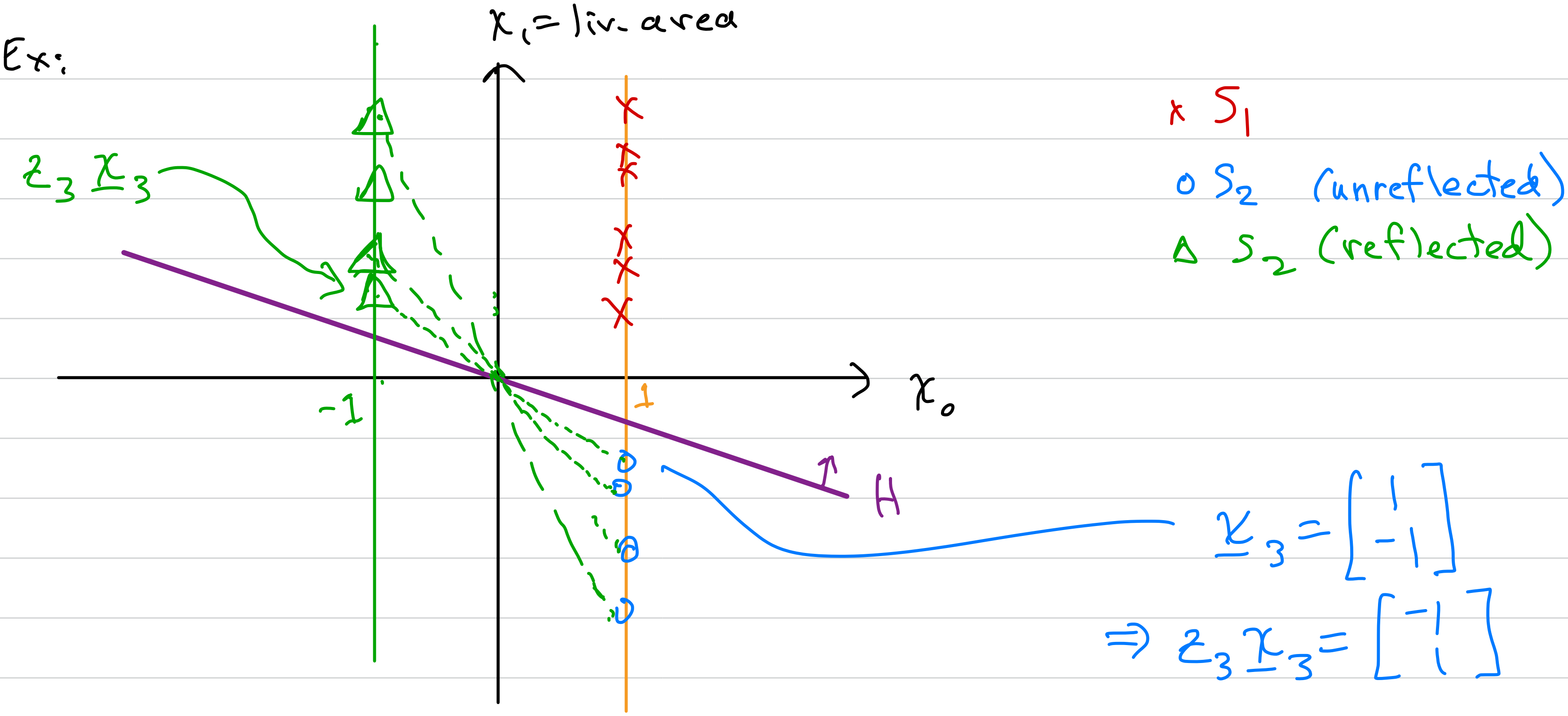
$$z_n^{(k)} = z_n = \text{class indicator} = \begin{cases} +1, & S_1 \\ -1, & S_2 \end{cases} \qquad (\text{Bishop } t_n)$$

$$z_n^{(k)} \underline{x}_n^{(k)} = z_n \underline{x}_n = \text{reflected data point} \qquad (\text{Bishop } t_n \tilde{\underline{x}}_n)$$

Matrix of reflected data points (2-class, augmented):

$$\underline{\underline{X}} = \begin{bmatrix} z_1 \underline{x}_1^T \\ z_2 \underline{x}_2^T \\ \vdots \\ z_N \underline{x}_N^T \end{bmatrix}_{[N \times (D+1)]} \qquad (\text{Bishop } \widetilde{\mathbf{X}})$$

Ex:



2D augmented feature space $(D = 1)$

Now: $\underline{w}^T z_n \underline{x}_n > 0 \Rightarrow$ correctly classified.

refl. data pt.

$g(z_n \underline{x}_n) > 0.$

Learning Algorithms: Problem Statement and Approach

The Learning Problem

Given a (training) set of labeled data points $\underline{x}_n^{(k)}$, find $\underline{w}$ (or $\underline{w}^{(k)}\ \forall k$) that yields optimal (in some sense) decision boundaries and regions. (or optimal output values $\hat{y}$ for regression problems)

→ Find the best value(s) of $\underline{w}$ or $\underline{w}^{(k)}$

Learning Algorithms – Approach

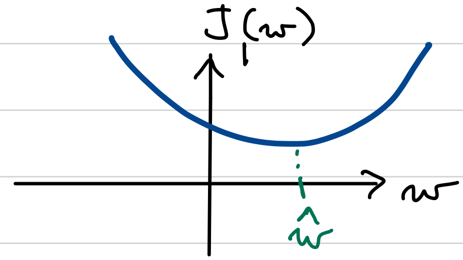
1. Construct a criterion function $J(\underline{w})$
   > So that a minimum of J occurs at a "good" or "optimal" value of $\underline{w}$.
2. (Optionally) set up constraints, for example:
   > To state additional preferences among possible choices of $\underline{w}$.

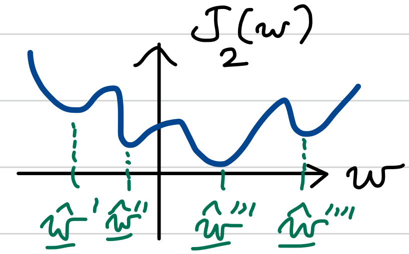
   > To include prior knowledge.
3. Use an optimization procedure to find $\hat{\underline{w}} = \arg\min_{\underline{w}} J(\underline{w})$.
   > Taking any constraints (from step 2) into account
4. Results in an "optimal" choice of $\underline{w} = \hat{\underline{w}}$.
   > "Optimal" decision boundaries and regions. (or values $\hat{y}$)

Criterion Functions $J(\underline{w})$

Desirable properties of $J(\underline{w})$

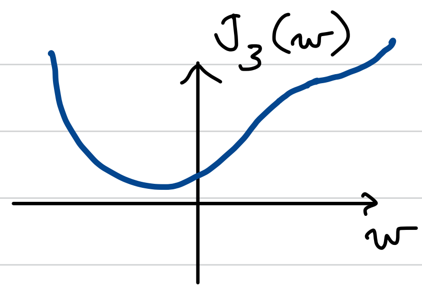
1. One minimum

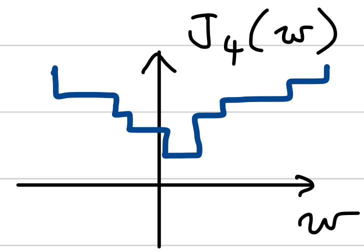




→ Strictly convex $J(\underline{w})$ will ensure this
→ Convex $J(\underline{w})$ is o.k.

2. Differentiable





→ Allows derivative-based optimization techniques

$J_4(w)$ is piecewise constant and has discontinuities — better for discrete search.

3. Optimization problems can include constraints:

e.g., must satisfy $\sum_{j=1}^{D} |w_j| = 1.$

→ Unconstrained optimization is simpler
→ Sometimes constrained optimization is better

Optimization Procedures for Convex, Differentiable $J(\underline{w})$; Unconstrained Case

Possible approaches:

(i) $\nabla_{\underline{w}} J(\underline{w}) = \dfrac{\partial J(\underline{w})}{\partial \underline{w}} = \underline{0} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$, then solve for $\hat{\underline{w}}$ algebraically

$\Rightarrow \hat{\underline{w}} = (\cdots\cdots)$

(ii) Gradient descent (many versions/variants)
- Iteratively update (move) $\underline{w}$ in direction of steepest descent (decrease) of J.

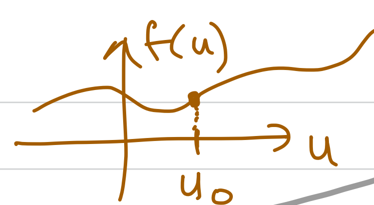
also, e.g.:

$\left.\begin{array}{l}\text{(iii) Newton's method (uses 2}^{\text{nd}}\text{ order derivatives)} \\ \\ \text{(iv) Search techniques}\end{array}\right\}$ won't cover in EE559.

Background

Let $\underline{u}$ be a dummy variable (e.g., $\underline{u} = \underline{x}$, or $\underline{u} = \underline{w}$).

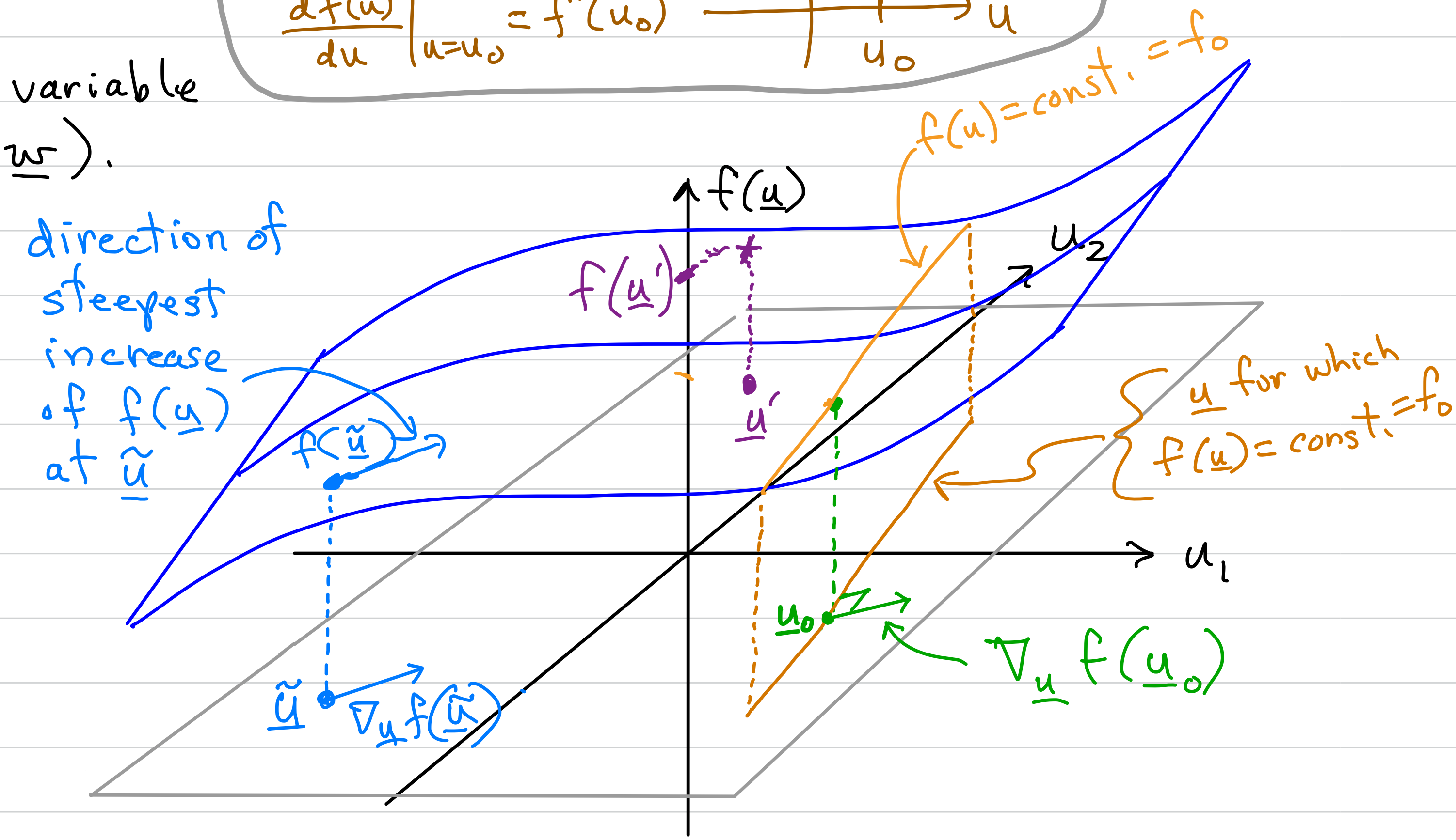
Scalar case: $\left.\dfrac{df(u)}{du}\right|_{u=u_0} = f'(u_0)$



$\nabla_{\underline{w}} J(\underline{w})$

$\downarrow$

$$\nabla_{\underline{u}} f(\underline{u}) \triangleq \begin{bmatrix} \frac{\partial}{\partial u_1} \\ \frac{\partial}{\partial u_2} \\ \vdots \\ \frac{\partial}{\partial u_D} \end{bmatrix}$$

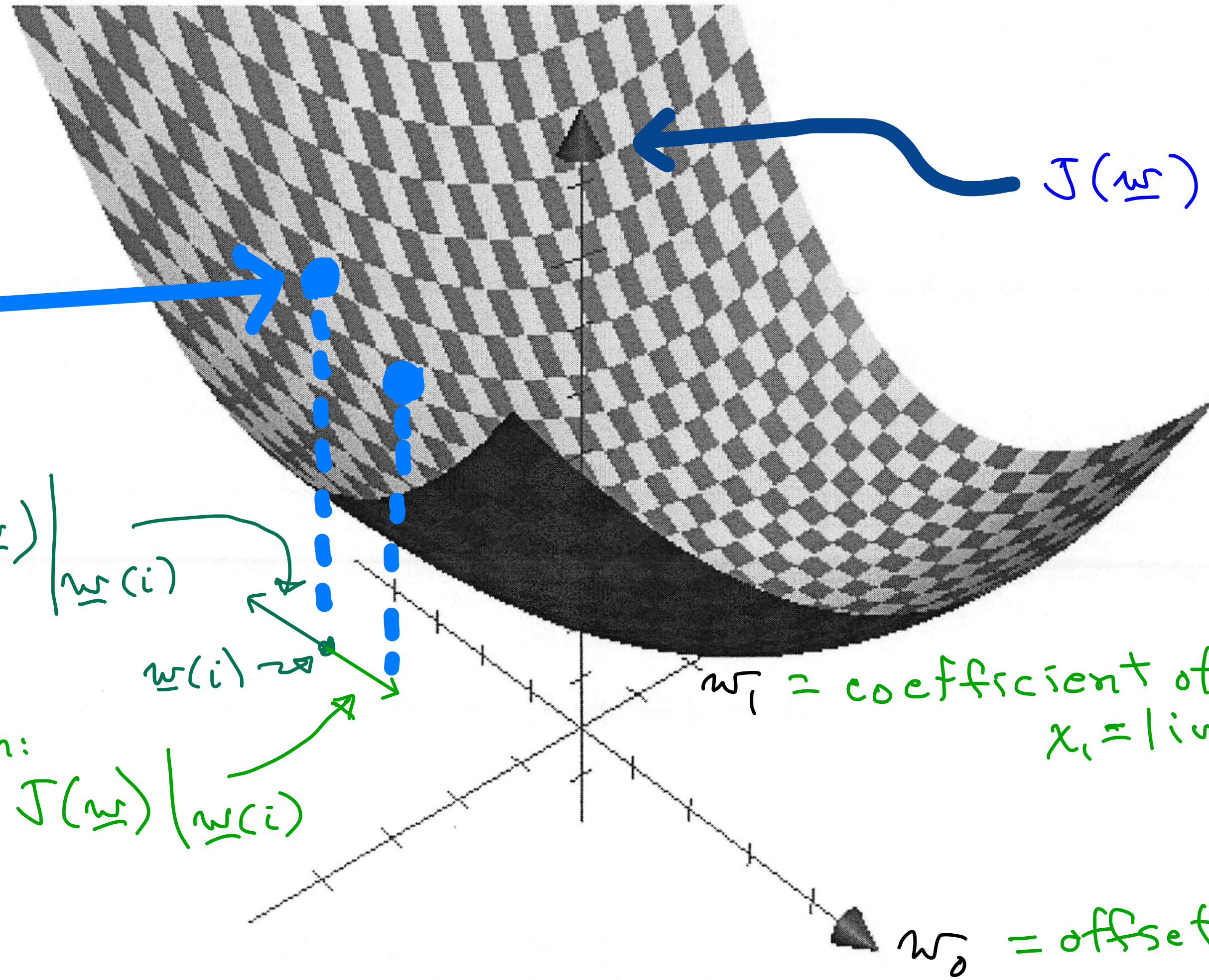


$\left.\nabla_{\underline{u}} f(\underline{u})\right|_{\underline{u}=\underline{u}_0} = \nabla_{\underline{u}} f(\underline{u}_0)$ points in what direction?

$\Rightarrow$ direction of steepest ascent (fastest positive change) of $f(\underline{u})$.

$\Rightarrow$ also, at $\underline{u} = \underline{u}_0$, $\nabla_{\underline{u}} f(\underline{u}) \perp$ $f(\underline{u}) = $ const. curve.

Augmented notation



G.D. update: direction: $-\nabla_{\underline{w}} J(\underline{w})\Big|_{\underline{w}(i)}$

$$\boxed{\underline{w}(i+1) = \underline{w}(i) - \eta(i)\, \nabla_{\underline{w}} J(\underline{w})\Big|_{\underline{w}(i)} \; ; \quad \eta(i) \geq 0}$$

Gradient descent

$\eta(i)$ = learning rate parameter

Notation:

Let $[\![\ ]\!]$ denote indicator function:

$$[\![h]\!] \triangleq \begin{cases} 1 & \text{iff } h \text{ is true} \\ 0 & \text{iff } h \text{ is false.} \end{cases}$$

Criterion Function – possible choices?

→ 2-class problems → Augmented notation

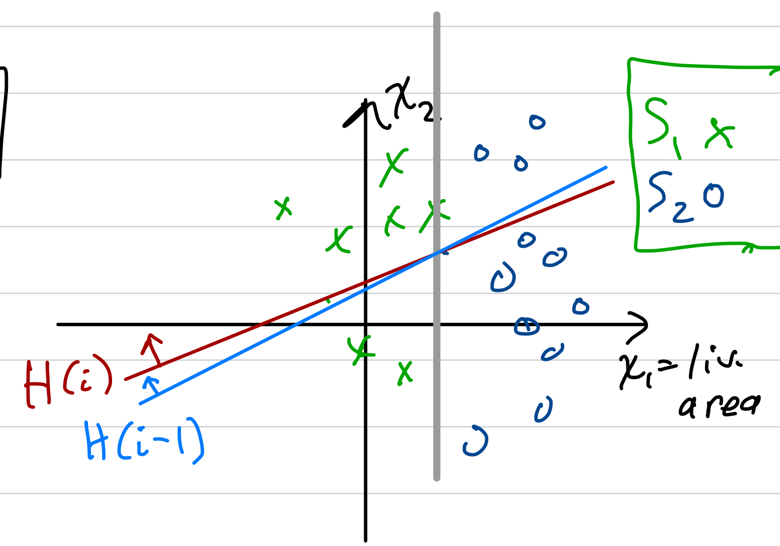
MSE? $J(\underline{w}) \stackrel{?}{=} \sum_{n=1}^{N} \| y(\underline{x}_n) - \hat{y}(\underline{x}_n) \|_2^2$

(y: true label, +1 or −1; $\hat{y}$: predict, +1 or −1)

⇒ J is piecewise constant.

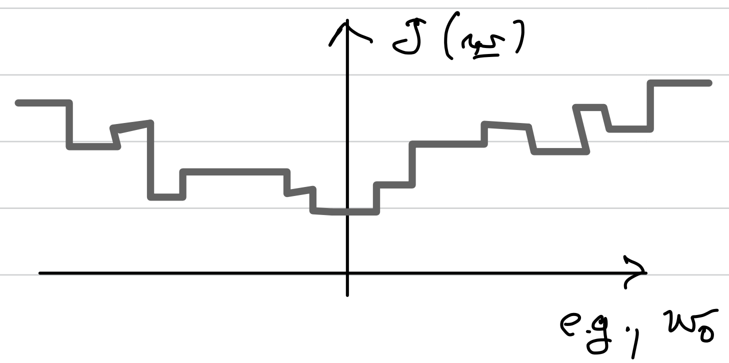
Try: $J_{er}(\underline{w})$ = error rate on training set

$$J_{er}(\underline{w}) = \frac{1}{N} \sum_{n=1}^{N} [\![g(z_n \underline{x}_n) \le 0]\!]$$



⇒ $J_{er}(\underline{w})$ is piecewise constant w/ discontinuities.

$\nabla_{\underline{w}} J_{er}(\underline{w}) = 0$ in any constant region



How about: $J_d(\underline{w}) = \sum_{n=1}^{N} \underbrace{[\![g(z_n\underline{x}_n) \le 0]\!]}_{\substack{+1 \text{ iff } \underline{x}_n \text{ is} \\ \text{misclass. or on} \\ \text{dec. boundary}}} \underbrace{d(z_n\underline{x}_n, H_{\underline{w}})}_{\substack{\text{distance } z_n\underline{x}_n \\ \text{to boundary } H_{\underline{w}}. \\ \text{(unsigned)}}}$

$$\Rightarrow J_d(\underline{w}) = -\sum_{n=1}^{N} [\![g(z_n\underline{x}_n) \le 0]\!] \frac{g(z_n\underline{x}_n)}{\|\underline{w}\|}$$

$d_s(H_{\underline{w}}, \underline{x}_n) = \dfrac{g(\underline{x}_n)}{\|\underline{w}\|}$

$d_s(H_{\underline{w}}, z_n\underline{x}_n) = \dfrac{g(z_n\underline{x}_n)}{\|\underline{w}\|}$

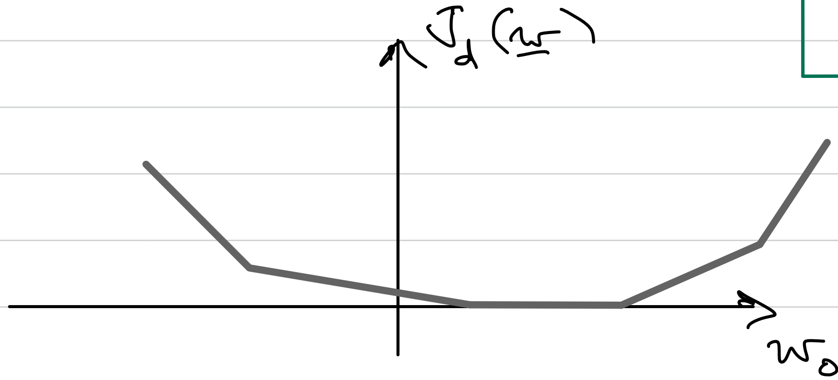
< 0 for misclassified $\underline{x}_n$

Perceptron Criterion Function $(\underline{x} = \underline{x}^{(+)}, \ \underline{w} = \underline{w}^{(+)})$

$$\boxed{J(\underline{w}) = -\sum_{n=1}^{N} [\![g(z_n\underline{x}_n) \le 0]\!] \ \underline{w}^T z_n \underline{x}_n}$$

Is $J(\underline{w})$ convex? $\rightarrow$ Yes.

Is $J(\underline{w})$ differentiable? $\rightarrow$ Almost everywhere



Perceptron Learning Algorithm $\quad (\underline{x}^{(+)}, \underline{w}^{(+)})$

Minimize J w.r.t. $\underline{w}$ by gradient descent (GD):

Use: $\nabla_{\underline{w}} (\underline{w}^T \underline{b}) = \underline{b}$

$$\nabla_{\underline{w}} J(\underline{w}) = -\sum_{n=1}^{N} [\![g(z_n \underline{x}_n) \leq 0]\!] \, z_n \underline{x}_n$$

$\Rightarrow$ GD, from $(*)$

$$\boxed{\begin{array}{l} \underline{w}(i+1) = \underline{w}(i) + \eta(i) \displaystyle\sum_{n=1}^{N} [\![g(z_n \underline{x}_n) \leq 0]\!] \, z_n \underline{x}_n, \quad \eta(i) \geq 0 \ \forall i \\ \text{Stop when } \displaystyle\sum_{n=1}^{N} [\![g(z_n \underline{x}_n) \leq 0]\!] \, z_n \underline{x}_n = 0 \\ \text{(i.e., when there are no misclassified data points)} \end{array}}$$

Perceptron Learning Algorithm using true (batch) GD

Note that $g(z_n \underline{x}_n) = \underline{w}^T(i) \, z_n \underline{x}_n$

Perceptron Learning Algorithm – Basic sequential GD version

$(\underline{x}^{(t)}, \underline{w}^{(t)})$

Update $\underline{w}(i)$ after each data point:

$$\underline{w}(i+1) = \underline{w}(i) + \eta(i) \, [\![g(z_n \underline{x}_n) \leq 0]\!] \, z_n \underline{x}_n , \quad \eta(i) \geq 0 \ \forall i$$

Stop when $\sum_{n=1}^{N} [\![g(z_n \underline{x}_n) \leq 0]\!] \, z_n \underline{x}_n = 0$

(Stop when $[\![g(z_n \underline{x}_n) \leq 0]\!] = 0$ for all of the last N iterations.)

Tip: best to randomly shuffle training data (over all classes) before iterating.