Machine Learning I: Supervised Methods

B. Keith Jenkins

Announcements

- Slido poll code: 1666822
  - slido.com
- Homework 4 is due Friday
- Midterm exam will be:
  - Wed., March 6, 2024
  - During regular lecture time
  - OHE 122 (on-campus and local students)
  - Format, etc., to be announced

Today's lecture

- MSE techniques for classification
  - Criterion function and interpretation
  - Pseudoinverse learning
  - Widrow-Hoff (LMS) learning
- Nonlinear classification and regression
  - Introduction
  - Ex: quadratic polynomials in $\underline{x}$
  - General nonlinear transformation or basis-set expansion
  - Learning in nonlinear classifiers and regressors

* v1.1: corrections made on $\underline{z}$ (page 2, 4).

Mean-squared-error techniques: Classification (augmented notation)

(2-class problems, reflected data points)

Can use above (MSE regression) techniques, except need a target output y_n.

Could choose: $y_n = z_n = \begin{Bmatrix} +1, & \underline{x}_n \in S_1 \\ -1, & \underline{x}_n \in S_2 \end{Bmatrix}$ ($\Leftarrow \underline{x}_n, y_n$ unreflected)

but can be more general: use b_n as a "target value", specified by the user.

(1) $\therefore$ $$\boxed{J(\underline{w}) = \frac{1}{N}\sum_{n=1}^{N}\left[g(z_n\underline{x}_n) - b_n\right]^2 = \frac{1}{N}\sum_{n=1}^{N}\left[\underline{w}^T z_n \underline{x}_n - b_n\right]^2, \quad b_n > 0 \ \forall n.}$$

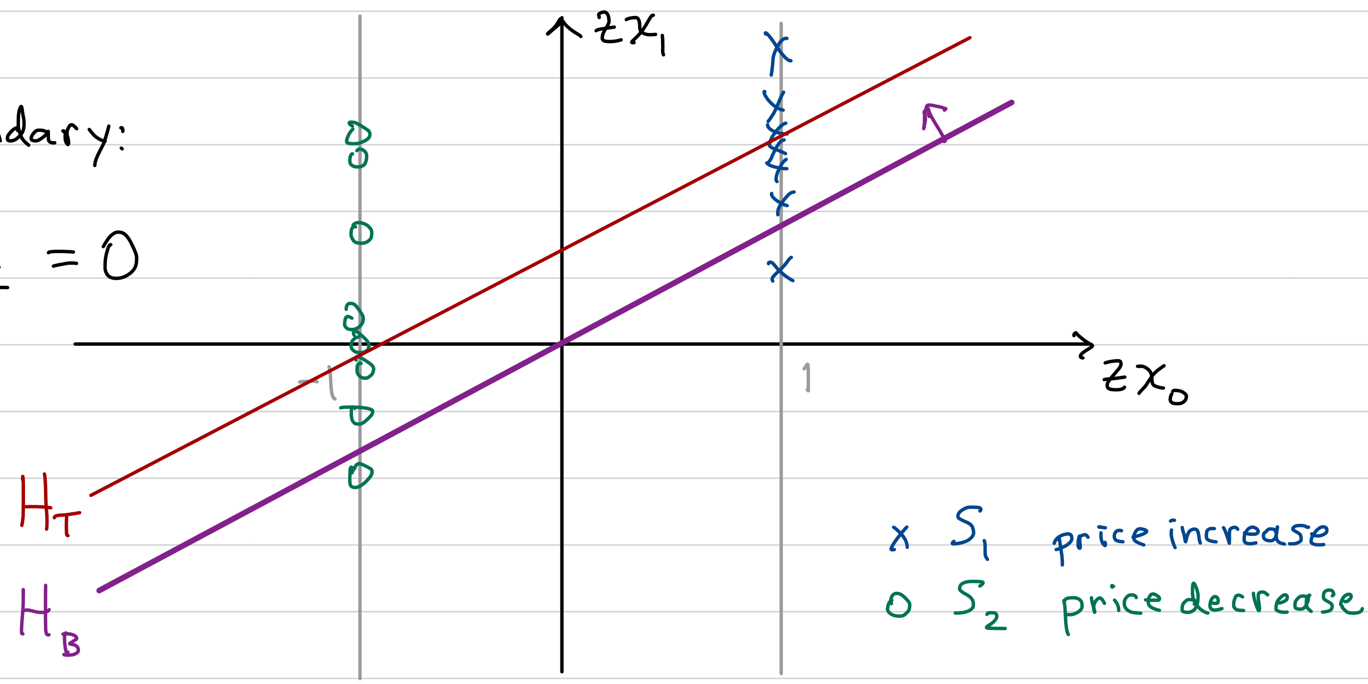
Let H_B = decision boundary:

$$g(z\underline{x}) = \underline{w}^T z\underline{x} = 0$$

Let $b_n = b \ \forall n$ for visualization

Let H_T = target hyperplane

* $\underline{w}^T z\underline{x} - b = 0$



Let H_T = target hyperplane: $g(z\underline{x}_T) = \underline{w}^T z\underline{x}_T = b \quad \forall \underline{x}_T$ on H_T.

Distance $d_S(H_B, z\underline{x}) = \dfrac{g(z\underline{x})}{\|\underline{w}\|} = d_1$ (in feature space)

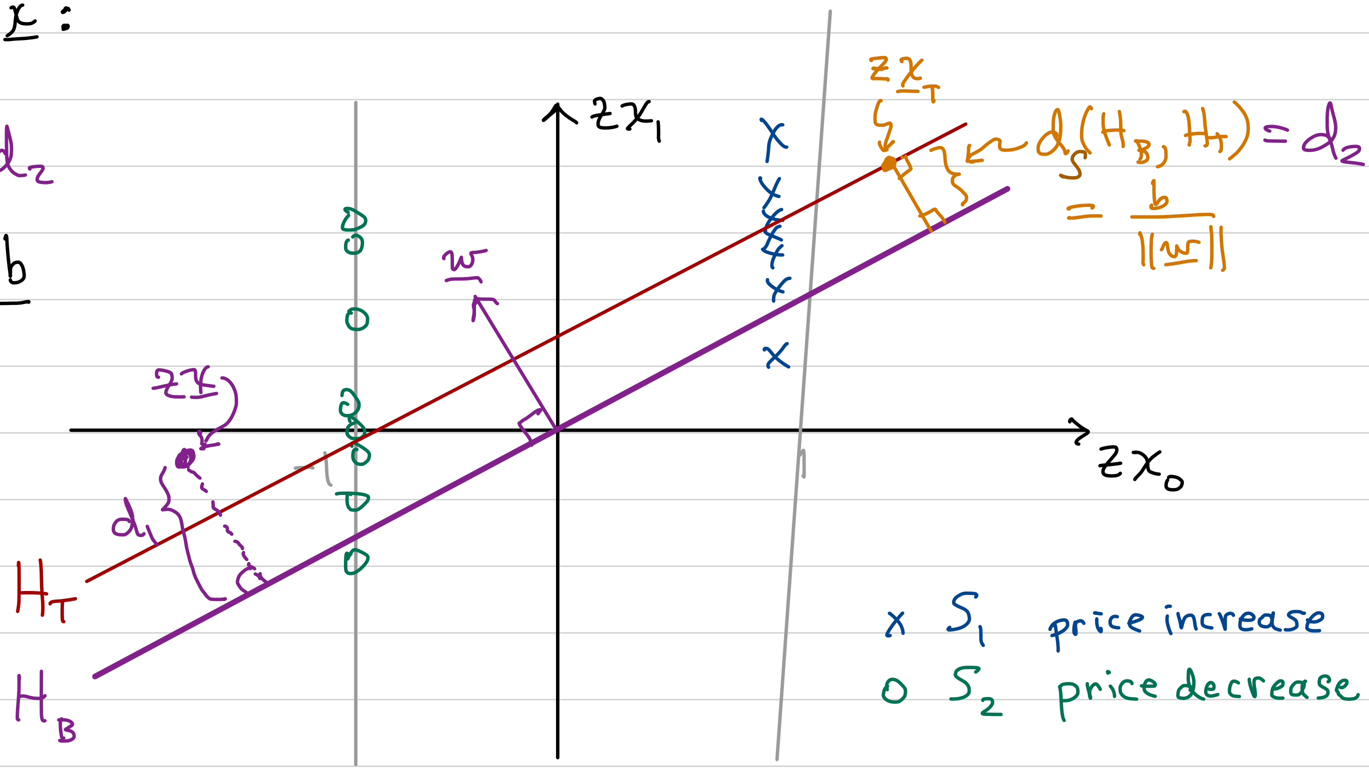
For $z\underline{x}_T$ on H_T, $d_S(H_B, z\underline{x}_T) = \dfrac{\underline{w}^T z\underline{x}_T}{\|\underline{w}\|} = \dfrac{b}{\|\underline{w}\|}$

(2) $\therefore\ d_S(H_B, H_T) = \dfrac{b}{\|\underline{w}\|} = d_2$

and for any point $z\underline{x}$:

(3) $d_S(H_T, z\underline{x}) = d_1 - d_2$

$$= \frac{\underline{w}^T z\underline{x} - b}{\|\underline{w}\|}$$



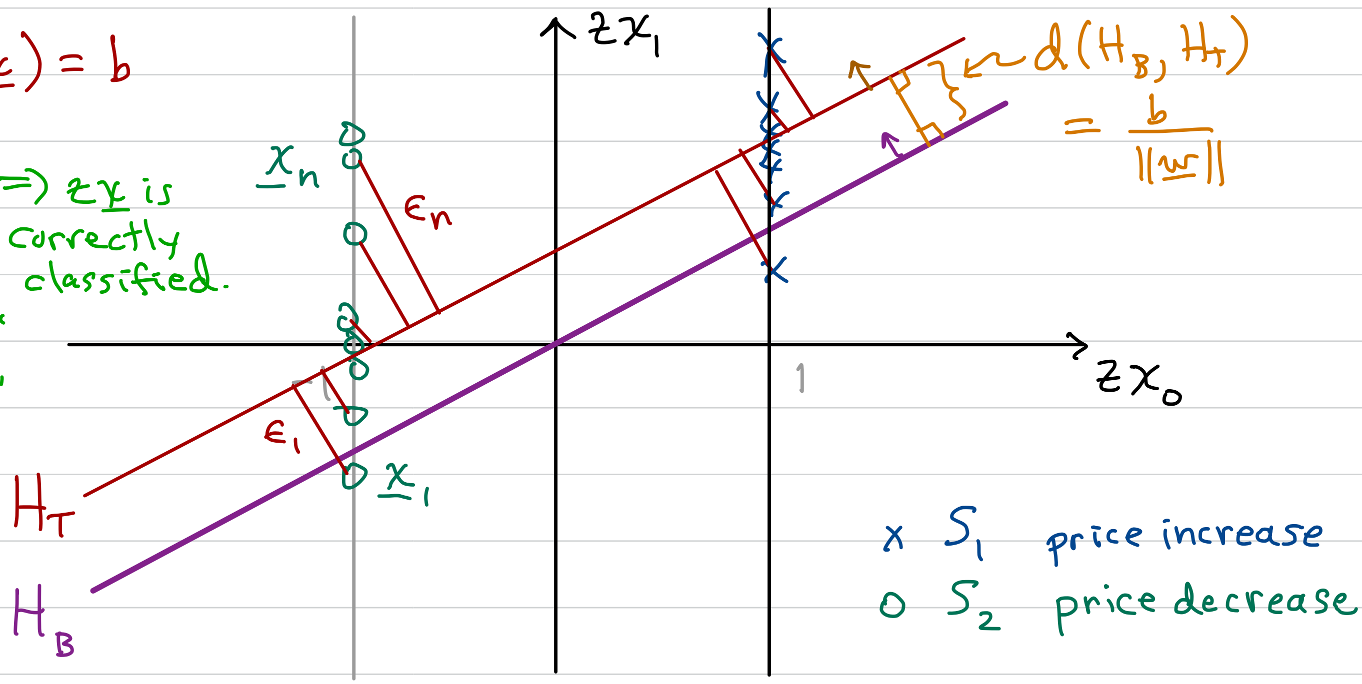
* $\Rightarrow \epsilon_n \triangleq \underline{w}^T z_n \underline{x}_n - b = \|\underline{w}\|\, d_s(H_T, z_n \underline{x}_n)$ from (3).

* $\epsilon_n^2 \propto d^2(H_T, z_n \underline{x}_n)$

* $J(\underline{w}) = \frac{1}{N}\sum_{n=1}^{N} \epsilon_n^2 = \frac{1}{N}\sum_{n=1}^{N} \|\underline{w}\|^2 d^2(H_T, z_n \underline{x}_n)$

H_T: $g(\underline{w}^T z \underline{x}) = b$

S $\begin{cases} g(\underline{w}^T z \underline{x}) > 0 \Rightarrow z\underline{x} \text{ is correctly classified.} \\ g(\,''\,) > b > 0 \quad '' \\ g(\,''\,) = b > 0 \quad '' \\ g(\,''\,) < b \quad X \end{cases}$



$J(\underline{w}) \propto$ mean-square distance of data points $z_n \underline{x}_n$ from target hyperplane.

$$J(\underline{w}) = \frac{1}{N}\sum_{n=1}^{N}\left[\underline{w}^T z_n \underline{x}_n - b_n\right]^2, \quad b_n > 0 \ \forall n \qquad (\text{here general } b_n > 0)$$

Let $\underline{\underline{X}}_r \triangleq \begin{bmatrix} z_1 \underline{x}_1^T \\ z_2 \underline{x}_2^T \\ \vdots \\ z_N \underline{x}_N^T \end{bmatrix}$, $\quad \hat{\underline{y}} = \begin{bmatrix} \underline{w}^T z_1 \underline{x}_1 \\ \underline{w}^T z_2 \underline{x}_2 \\ \vdots \\ \underline{w}^T z_N \underline{x}_N \end{bmatrix}$, $\quad \underline{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix}$

$$J(\underline{w}) = \frac{1}{N}\left\|\underline{\underline{X}}_r \underline{w} - \underline{b}\right\|_2^2 = \frac{1}{N}\left(\underline{\underline{X}}_r \underline{w} - \underline{b}\right)^T\left(\underline{\underline{X}}_r \underline{w} - \underline{b}\right)$$

How to minimize $J(\underline{w})$?

$\rightarrow$ same as for MSE regression, with $\underline{y} \rightarrow \underline{b}$.

1. Pseudoinverse learning algorithm (classification)

Let $\underline{\underline{X}}_r^- \triangleq (\underline{\underline{X}}_r^T \underline{\underline{X}}_r)^{-1} \underline{\underline{X}}_r^T$ = Moore-Penrose (left) pseudoinverse of $\underline{\underline{X}}_r$

Then $\underline{\hat{w}} = \underline{\underline{X}}_r^- \underline{b}$

$\Rightarrow$ Predictions on dataset $\underline{\underline{X}}_r$: $\left[\underline{\underline{X}}_r \underline{\hat{w}}\right]_i > 0 \Rightarrow \underline{x}_i$ is correctly classified

S

$\underline{\underline{X}}_r \underline{\hat{w}} > \underline{0} \Rightarrow$ All data pts. are correctly classified.

$(\underline{a} > \underline{b} \Rightarrow a_i > b_i \ \forall i)$.

$\Rightarrow$ Predictions on unknowns: $g(\underline{x}) = \underline{\hat{w}}^T \underline{x} \underset{\Gamma_2}{\overset{\Gamma_1}{\gtrless}} 0$.

(nonreflected $\underline{x}$)

2. LMS (or Widrow-Hoff (W-H)) learning algorithm (classification)

$$\underline{w}(i+1) = \underline{w}(i) - \eta(i)\left(\underline{w}(i)^T z_n \underline{x}_n - b_n\right) z_n \underline{x}_n, \quad n = (i \bmod N) + 1$$

$(i: 0, 1, 2, \cdots; \; n: 1, 2, \cdots, N)$

Final weight vector $\hat{\underline{w}}$

$\Rightarrow$ Predictions on unknowns: $g(\underline{x}) = \hat{\underline{w}}^T \underline{x} \underset{\Gamma_2}{\overset{\Gamma_1}{\gtrless}} 0$ (unreflected $\underline{x}$)

Comments on MSE techniques for classification

1. How to choose $\underline{b}$?
   (a) if have prior info. (e.g., domain knowledge), use it.
   (b) if not, typically:
   $b_n = 1 \;\forall n$, or $\underline{b} = \underline{1}$ (for reflected data pts.)
   $\rightarrow$ Tends to work well.
   (c) $\exists$ algorithms that find $\hat{\underline{w}}$ and $\hat{\underline{b}}$ together.
   $\Rightarrow$ Ho-Kashyap algorithm. [N.R.F.]

2. For iterative GD (LMS or W-H), convergence?

(a) One can show convergence (in the mean-square sense) to the MSE solution $\hat{\underline{w}}$, if the following conditions on $\eta(i)$ are met:

$$\lim_{m \to \infty} \sum_{i=1}^{m} \eta(i) = +\infty$$

and
$$\lim_{m \to \infty} \sum_{i=1}^{m} \eta^2(i) < \infty$$

Ex: $\eta(i) = \frac{1}{i}$ satisfies these, but in practice can lead to very slow convergence.

MSE techniques for classification

+ Pseudoinv. solution will always give a $\hat{\underline{w}}$ (if $(\underline{\underline{X}}^T \underline{\underline{X}})$ is invertable) that mimimizes the MSE, even for data that is not linearly separable.

- Choice of $\underline{b}$ can affect performance.
- Calculating $\underline{\underline{X}}^-$ can be computationally expensive. Could have stability issues.
- No guarantee of 100% correct classification if data is linearly separable.
- More computation per iteration (than perceptron).
- More parameters to specify than perceptron.

Nonlinear Regression and Classification (augmented notation)

All learning algorithms we have covered so far, are linear: (augmented notation)

$g = \underline{w}^T \underline{x}$ (2 class); $g_k(\underline{x}) = \underline{w}_k^T \underline{x}$ or $g_{kj}(\underline{x}) = \underline{w}_{kj}^T \underline{x}$ (multiclass);

$\hat{y}(\underline{x}) = \hat{f}(\underline{x}) = \underline{\hat{w}}^T \underline{x}$ (regression)

Comment: are above functions linear? Linear in $\underline{x}$, and linear in $\underline{w}$.

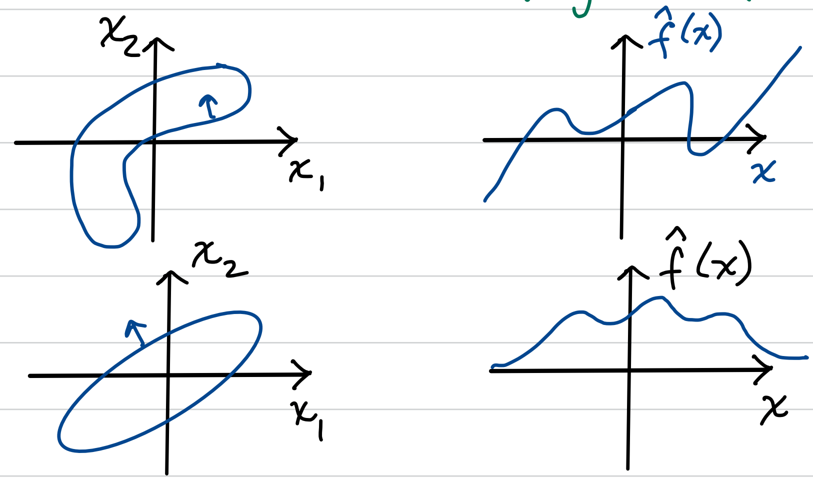
$\Rightarrow$ Now consider: nonlinear in $\underline{x}$

Examples: g or $\hat{y}$ is fcn. of:

- polynomials in $\underline{x}$
- exp {some fcn. of $\underline{x}$}

⋮

classification regression



Today: consider polynomials

$\hat{f}(\underline{x})$ or $g(\underline{x})$ as polynomial functions of $\underline{x}$

Ex: Quadratic functions of $\underline{x}$, $\quad g_k(\underline{x}) = ?$

$$g_k(\underline{x}) = \sum_{m=1}^{D} \sum_{n=1}^{D} w_{mn}^{(k)} x_m^{(0)} x_n^{(0)} + \sum_{n=1}^{D} w_{0n}^{(k)} x_n^{(0)} + w_{00}^{(k)}$$

$$w_{mn}^{(k)} = 0 \text{ if } m > n$$

$$\boxed{g_k(\underline{x}) = \underline{x}^{(+)T} \underline{\underline{W}}\, \underline{x}^{(+)}} \quad \text{[classification]}$$

or

$$\boxed{\hat{f}(\underline{x}) = \underline{x}^{(+)T} \underline{\underline{W}}\, \underline{x}^{(+)}} \quad \text{[regression]}$$

in which $\underline{\underline{W}}_{[(D+1)\times(D+1)]}$ is upper triangular

How many degrees of freedom in each $g_k(\underline{x})$ or $\hat{f}(\underline{x})$?

d.o.f. = # of w scalar variables

	Quadratic ($m \neq n$)	Quadratic ($m = n$)	Linear	Constant
# scalar w variables:	$\binom{D}{2} = \frac{D!}{2!(D-2)!} = \frac{D(D-1)}{2}$	$D = \frac{2D}{2}$	$D = \frac{2D}{2}$	1

$$\text{Total} = \frac{D(D-1) + 2D + 2D}{2} + 1 = \frac{D^2 + 3D}{2} + 1 \triangleq D' + 1, \quad D' = \frac{1}{2}(D^2 + 3D)$$

Define new notation:

Let:

$$\underline{u} = \underline{\phi}(\underline{x}) \triangleq \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_D \\ x_1^2 \\ x_1 x_2 \\ x_1 x_3 \\ \vdots \\ x_1 x_D \\ x_2^2 \\ x_2 x_3 \\ \vdots \\ x_2 x_D \\ \vdots \\ x_{D-1} x_D \\ x_D^2 \end{bmatrix}, \quad \underline{w}^{(k)\prime} = \underline{w}_k' \triangleq \begin{bmatrix} w_{00}^{(k)} \\ w_{01}^{(k)} \\ w_{02}^{(k)} \\ \vdots \\ w_{0D}^{(k)} \\ w_{11}^{(k)} \\ w_{12}^{(k)} \\ \vdots \\ w_{DD}^{(k)} \end{bmatrix}$$

Both vectors have dimensionality: $D'+1$.

$g_k(\underline{x}) \longrightarrow g_k(\underline{u}) = ?$

Then:

$$\boxed{\begin{array}{lll} \text{Discr. fcn.:} & g_k(\underline{u}) = \underline{w}_k'^T \underline{u} = \underline{w}_k'^T \underline{\phi}(\underline{x}) \\ \text{Regr. fcn:} & \hat{f}(\underline{u}) = \underline{w}'^T \underline{u} = \underline{w}'^T \underline{\phi}(\underline{x}) \end{array}}$$

S Both are linear in $\underline{u}$, and linear in $\underline{w}'$.

$\boxed{\Rightarrow \text{Use } \underline{u}\text{-space (or } \underline{\phi}\text{-space) as a new ("expanded") feauture space!}}$

<u>More generally</u>, let $\underline{\phi}(\underline{x})$ be a set of basis functions $\phi_{j'}(\underline{x})$, $j' = 0, 1, 2, \cdots, D'$, polynomial or other (nonlinear) functions of $\underline{x}$.