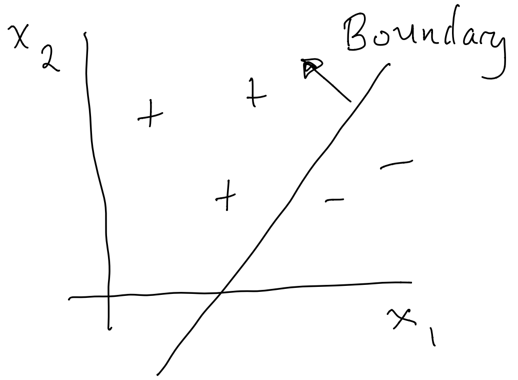
CS 181

Linear Classification



$D = \{ (\underline{x}_1, y_1), \dots, (\underline{x}_n, y_n) \}$

$\underline{x}_n \in \mathbb{R}^D$

$y_n \in \{+1, -1\}$

Goal: predict $\hat{y}$, on a new example $\underline{x}$

<u>Method 1</u> Non-parametric

k-Nearest Neighbor kernel methods

↳ majority vote on the k-NNs

✓ flexible ⊗ Slow on big data set

⊗ Not interpretable

Method 2 Linear classification

discriminant function

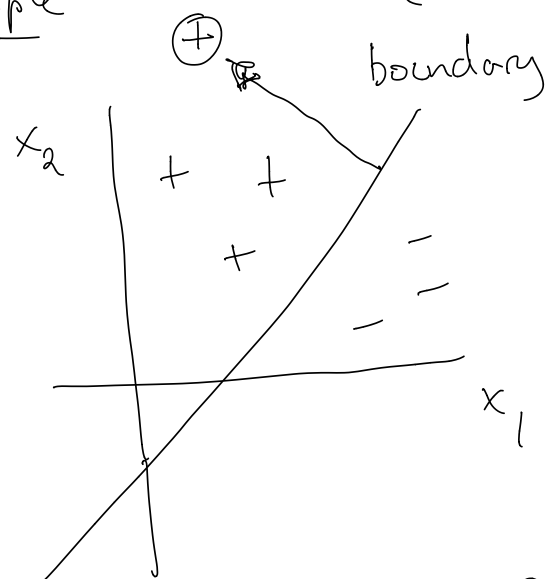
$$h(\underline{x}, \underline{w}) = \underline{w}^T \underline{x} + w_0$$

predict $\hat{y} = \begin{cases} +1 & \text{if } h(\underline{x}, \underline{w}) > 0 \\ -1 & \text{otherwise} \end{cases}$

Example

$h(\underline{x}, \underline{w}) = 0$

boundary



orthogonal vector

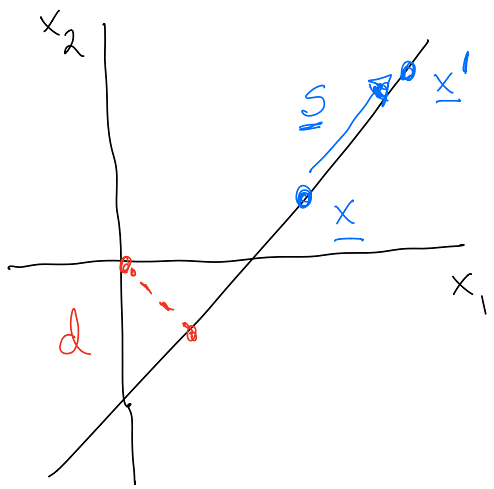
$$\underline{w} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

$$h(\underline{x}, \underline{w}) = -x_1 + x_2 + 1$$

Understanding boundary

Generally, hyperplane

$$S = \{ \underline{x} : \underline{w}^T \underline{x} + w_0 = 0 \}$$



$$\underline{x} + \underline{s} = \underline{x}'$$

$$\underline{w}^T \underline{s} = \underline{w}^T \underline{x}' - \underline{w}^T \underline{x}$$

$$= (\underline{w}^T \underline{x}' + w_0) - (\underline{w}^T \underline{x} + w_0)$$

$$= 0 - 0 = 0$$

So, $\underline{w}$ is orthogonal to the boundary.

↳ $\underline{w}$ sets the orientation.

d as the unsigned orthonormal distance between boundary + origin

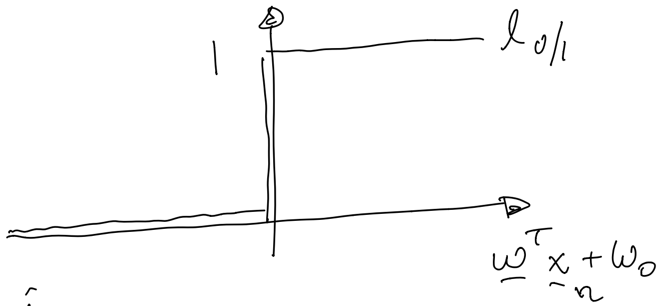
$$d = \frac{|w_0|}{\|w\|_2}$$

↳ w_0 sets the distance to the origin

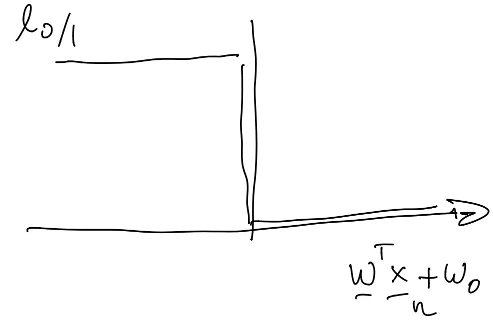
0/1 Loss function

Recall $\hat{y} = \begin{cases} +1 & , \text{ if } \underline{w}^T x + w_0 > 0 \\ -1 & \text{otherwise} \end{cases}$

For a negative example $(\underline{x}_n, y_n)$



For a positive example



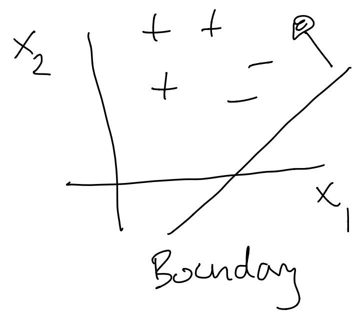
Define

$$l_{0/1}(z) = \begin{cases} 1, & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\mathcal{L}_D(\underline{w}) = \sum_n l_{0/1}\left(-y_n(\underline{w}^T \underline{x}_n + w_0)\right) = \sum_{n : y_n \neq \hat{y}_n} 1$$

$\rightarrow$ scales with # mistakes

Problem



Want to use gradient descent

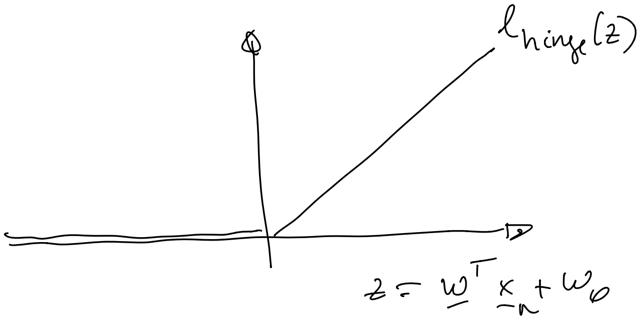
$$\underline{w}_{t+1} \leftarrow \underline{w}_t - \eta \frac{\partial \mathcal{L}(\underline{w})}{\partial(\underline{w})}$$

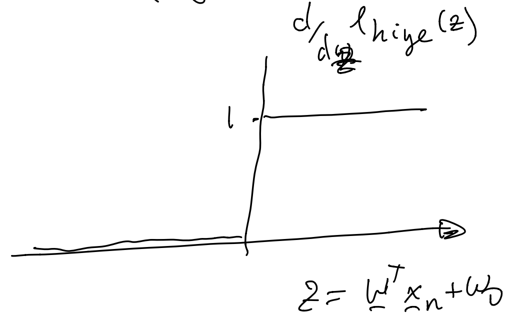
η step size (>0)

But, gradients of 0/1 loss function are uninformative! Zero almost everywhere.

Fix: Hinge loss

For a negative training example ($y_n = -1$)

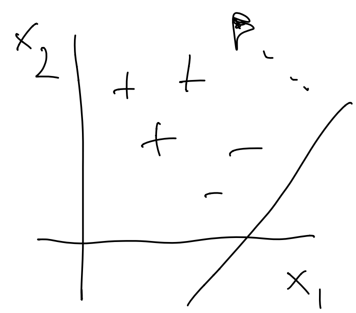




$$\ell_{hinge}(z) = \begin{cases} z & \text{if } z \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

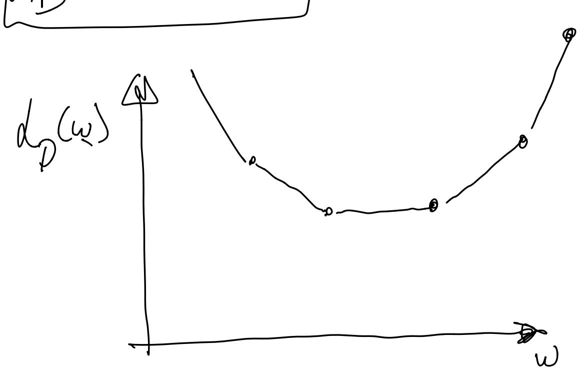
$$L_D(\underline{w}) = \sum_n \ell_{hinge}\left(-y_n(\underline{w}^T \underline{x} + w_0)\right) = \sum_{n: y_n \neq \hat{y}_n} -y_n(\underline{w}^T \underline{x}_n + w_0)$$

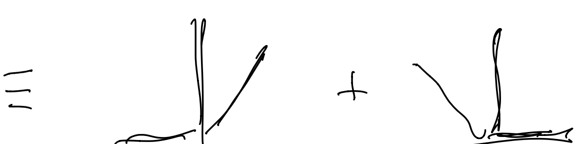
L_D increases with more mistakes & worse mistakes



Gradient descent can now work

$\boxed{L_D \text{ is Convex}}$





positive sum of convex is convex!

Comments

① Convex function (+ differentiable) can be minimized via grad descent } for small enough η

② Convex opt. is polynomial time solvable (0/1 NP-hard)

Gradient descent

Repeat:

Update $\underline{w}$ with step size η × gradient

$$\frac{1}{n} \sum_n \frac{\partial}{\partial \underline{w}} \ell_{hinge}(\underline{x}_n, \underline{w}_n)$$

☑ Stable (converges for fixed η)

☒ Costly per step

Stochastic gradient descent

Repeat:

Pick random minibatch $B \subseteq D$ (perhaps $|B| = 1$)

Update $\underline{w}$ with step size η × gradient on B

$$\frac{1}{|B|} \sum_{n \in B} \frac{\partial}{\partial \underline{w}} \ell_{hinge}(\underline{x}_n, \underline{w}_n)$$

☑ Faster on large data

☒ Noisy gradient

(need an adaptive learning rate ("step size") to converge, e.g., $\eta_t \propto \frac{1}{t}$)

Aside

* Absorbed w_0 into $\underline{w}$ *

$$\frac{\partial}{\partial \underline{w}} L_D(\underline{w}) = \frac{\partial}{\partial \underline{w}} \left(- \sum_{n: y_n \neq \hat{y}_n} y_n (\underline{w}^T \underline{x}_n) \right)$$

$$= - \sum_{n: y_n \neq \hat{y}_n} y_n \underline{x}_n$$

SGD on hinge loss $(|B| = 1)$

Perceptron

Rosenblatt (1958)

$\eta = 1$

① Pick $(\underline{x}_n, y_n)$ at random

② If $y_n = \hat{y}$, do nothing

Else $\underline{w}_{t+1} \leftarrow \underline{w}_t + \eta y_n \underline{x}_n$

Thm Perceptron converges iff and only if data is linearly seperable

$\boxed{\text{Remark 1}}$

Multiclass classification

$$y_k \in \{C_1, \dots, C_K\}$$

"All vs one"

Train K separate binary classifiers.

Classify a new example as

$$\arg\max_k h(\underline{x}, \underline{w}_k)$$

$\boxed{\text{Remark 2}}$

Basis functions

Metris

	True 0	True 1
Predict 0	TN	FN
Predict 1	FP	TP

TNR or SPEC

$$\frac{TN}{TN + FP}$$

"reject all negative?"

TPR or SENS or Recall

$$\frac{TP}{FN + TP}$$

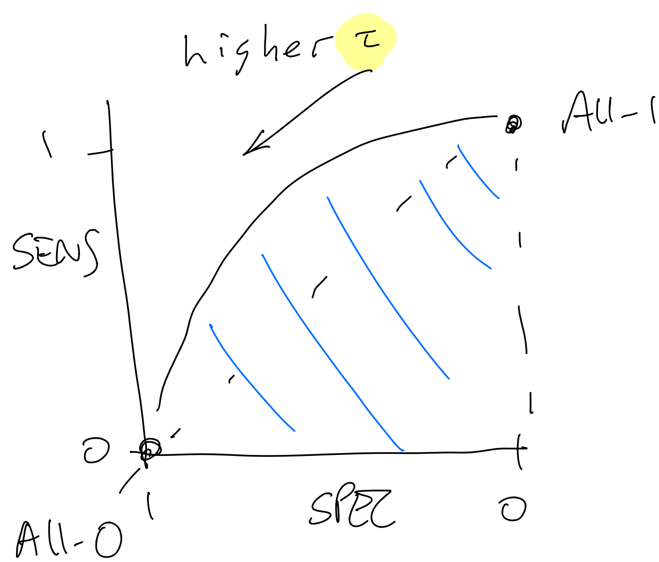
"find all positive?"

ROC curve / AUC

new parameter

$$\text{Classify } \hat{y} = \begin{cases} +1 & \text{if } h(\underline{w}, \underline{x}) > \text{threshold } \tau \\ -1 & \text{otherwise} \end{cases}$$

Generally we've used $\tau = 0$



"Area under curve"

AUC $\in [0.5, 1]$

Higher better

F1-score / Prediction & Recall

TN	FN
FP	TP

PREC

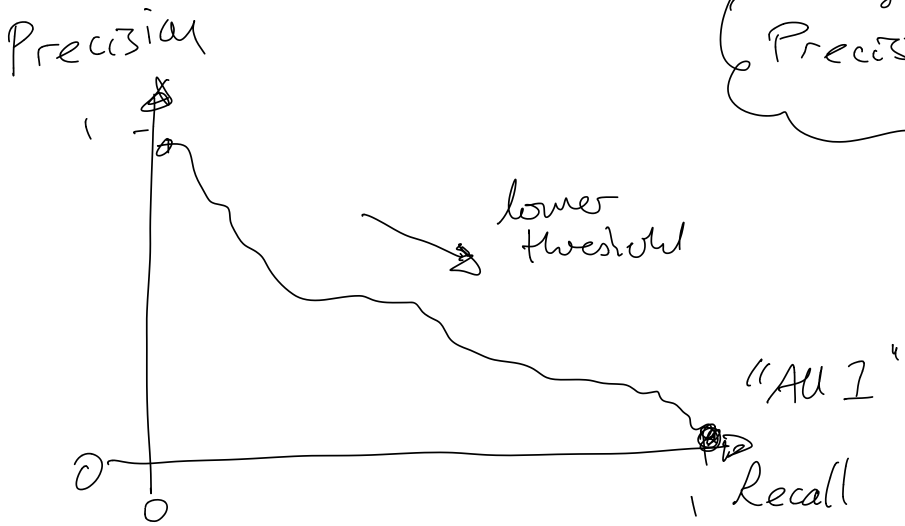
$$\frac{TP}{FP + TP}$$

RECALL

$$\frac{TP}{FN + TP}$$

$$\text{F1-score} = 2 \times \frac{P \times R}{P + R}$$

(harmonic mean)



Precision vs. recall