COGS 118A, Winter 2020

Supervised Machine Learning Algorithms

# Lecture 6: Robust Estimation and Error Metrics

Zhuowen Tu

# Midterm 1

Midterm I, 01/30/2020 (Thursday)

Time: 12:30-13:50PM

Location: Ledden Auditorium

You can bring one page "cheat sheet". No use of computers/smart-phones during the exam.

Bring your pen.

Bring your calculator.

A study guide and practice questions will be provided.

Conclusions for linear regression with the least square estimation

# Linear regression:

- Univariate linear regression

Output: $y = w_0 + w_1 x_1$

- Polynomial linear regression

Output: $y = w_0 + w_1 x_1 + w_2 x_1^2 + ... + w_q x_m^q$

- Multivariate linear regression

Output: $y = w_0 + w_1 x_1 + w_2 x_2 + ... + w_m x_m$

---

They all share a general form (when generalizing X):

$$Y = X W$$

linear w.r.t. the W!

---

With an analytical solution:

$$W^* = (X^T X)^{-1} X^T Y$$

Conclusions for linear regression with the least square estimation

## Linear regression:

They all share a general form (when generalizing X):

$$Y \qquad = \qquad X \qquad\qquad W$$

Linear regression is the definition for the estimation function.

With an analytical solution:

$$W^* = (X^T X)^{-1} X^T Y$$

Least square estimation is about the estimation method. We can use methods other than the least square estimation to solve the linear regression problem.

# Robust Estimation

# Estimation and optimization

$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..n\}$$

Different choices of the penalty will lead to different robustness measure:

Squared error:

$$e = \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i; \mathbf{w}))^2$$

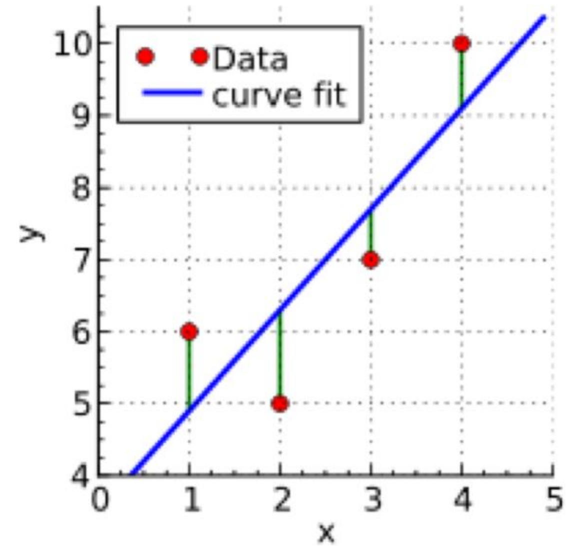
L1 error:

$$e = \sum_{i=1}^{n} |y_i - f(\mathbf{x}_i; \mathbf{w})|$$

# Estimation and optimization

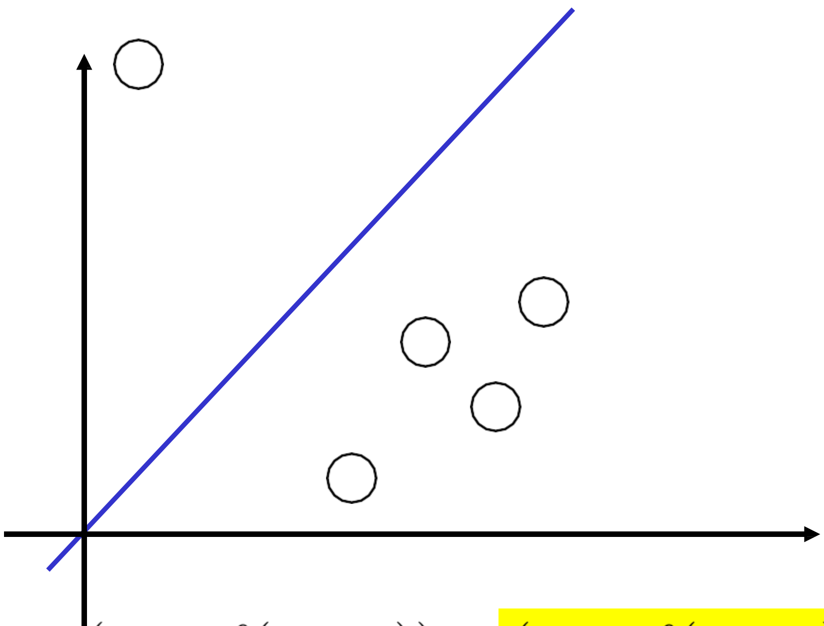
$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..n\}$$

$$cost(y_i - f(\mathbf{x}_i;\theta)) = (y_i - f(\mathbf{x}_i;\theta))^2$$

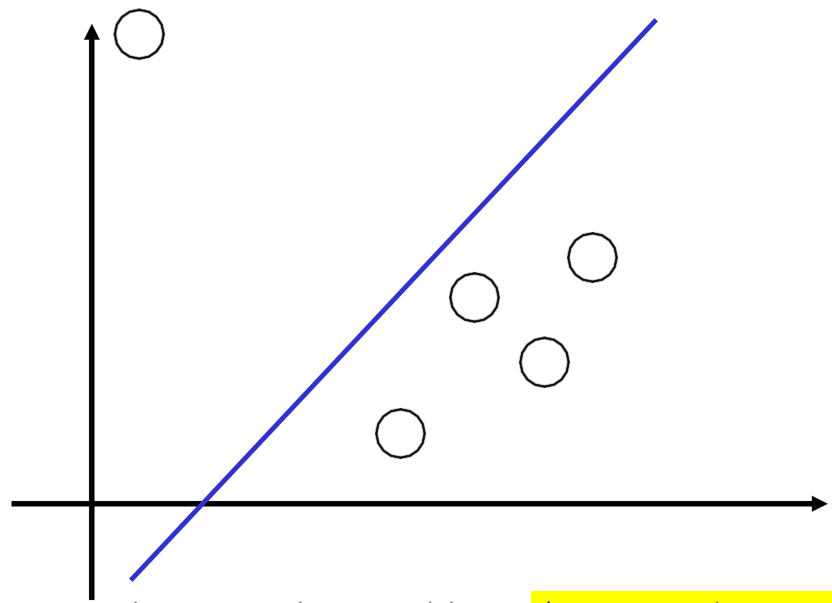



A general form:

$$\mathbf{w} = \arg\min_\theta \sum_{i=1}^{n} cost(y_i - f(\mathbf{x}_i;\mathbf{w}))$$



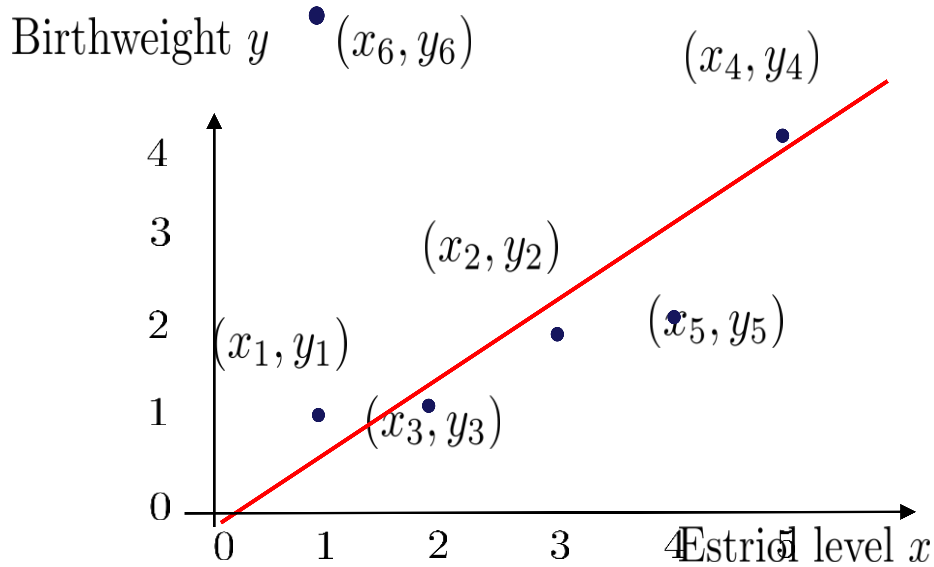
$$cost(y_i - f(\mathbf{x}_i;\mathbf{w})) = (y_i - f(\mathbf{x}_i;\mathbf{w}))^2$$



$$cost(y_i - f(\mathbf{x}_i;\mathbf{w})) = |y_i - f(\mathbf{x}_i;\mathbf{w})|$$

# Robust estimation

$$
\begin{array}{cccc}
Y & = & X & W \\
\begin{pmatrix} 1 \\ 1.9 \\ 1.05 \\ 4.1 \\ 2.1 \\ \color{red}{6.0} \end{pmatrix} & & \begin{pmatrix} 1, 1 \\ 1, 3 \\ 1, 2 \\ 1, 5 \\ 1, 4 \\ \color{red}{1, 1.1} \end{pmatrix} & \begin{pmatrix} w_0 \\ w_1 \end{pmatrix}
\end{array}
$$



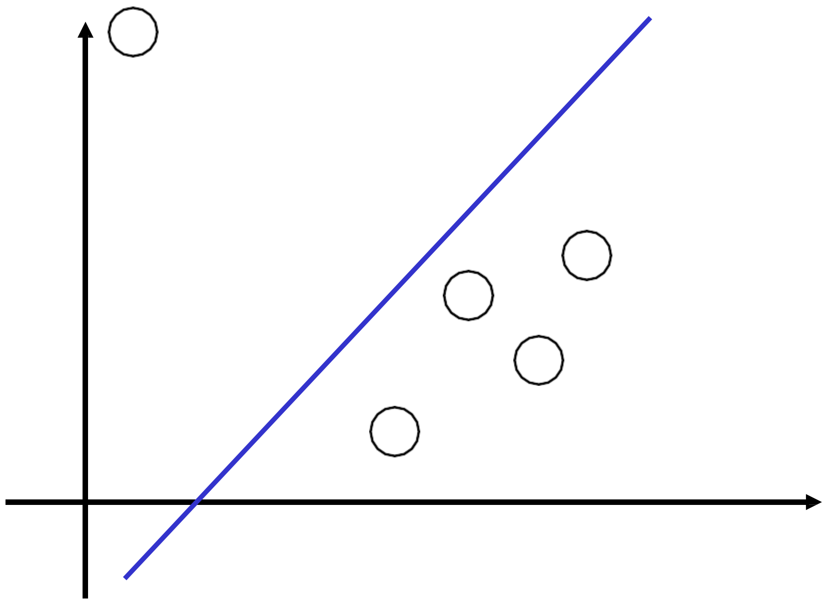

$$W^* = (X^T X)^{-1} X^T Y$$

$$\color{blue}{W^* = \arg min_W \sum_i (\mathbf{x}_i^T \cdot W - y_i)^2}$$

$$\color{red}{W^* = \arg min_W \sum_i |\mathbf{x}_i^T \cdot W - y_i|}$$

# Linear regression with L1 loss

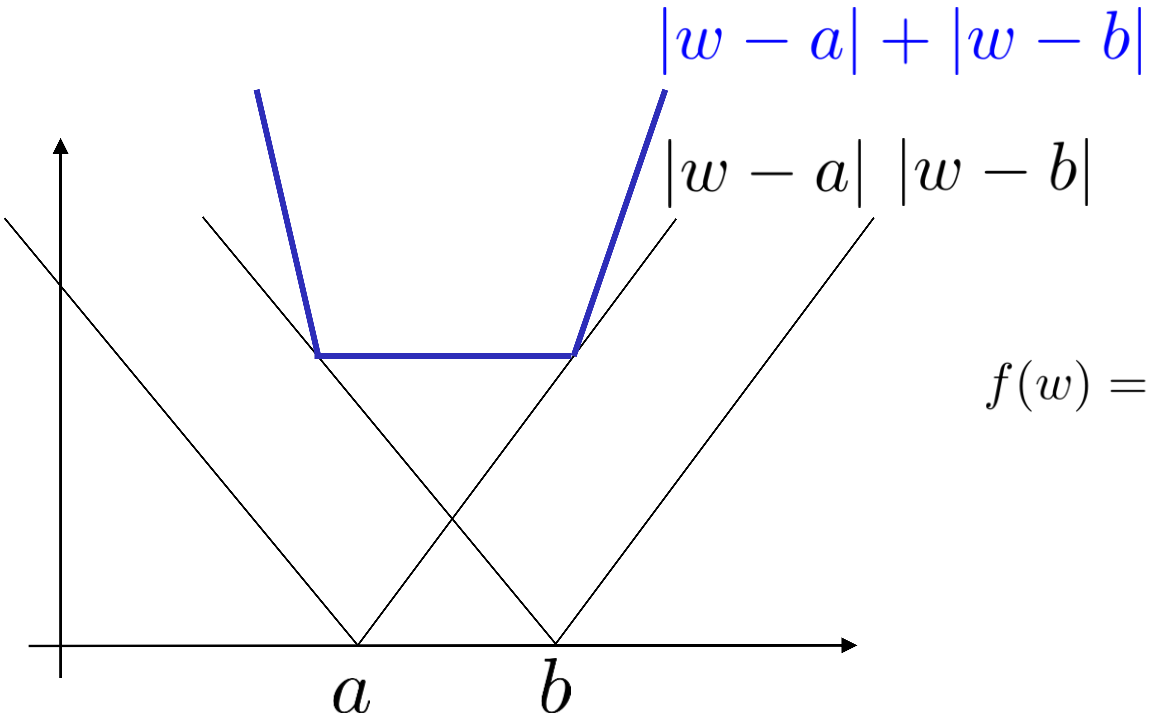
$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..n\}$$



$$\mathbf{w}^* = \arg\min_\theta \sum_{i=1}^n |y_i - f(\mathbf{x}_i; \mathbf{w})|$$

# Summation of two functions

$$f(w) = |w-a| + |w-b|,\ b > a$$




$$f(w) = \begin{cases} 2w - a - b & w \geq b \\ b - a & a \leq w \leq b \\ a + b - 2w & w < a \end{cases}$$

Question?

Is the summation of two convex functions convex?

$$\text{Convex: } f(x),\ g(x)$$

$$f(x) + g(x)?$$

A. Yes

B. No

C. It depends

Question?

Is the summation of two convex functions convex?

$$\text{Convex: } f(x),\ g(x)$$

$$f(x) + g(x)?$$

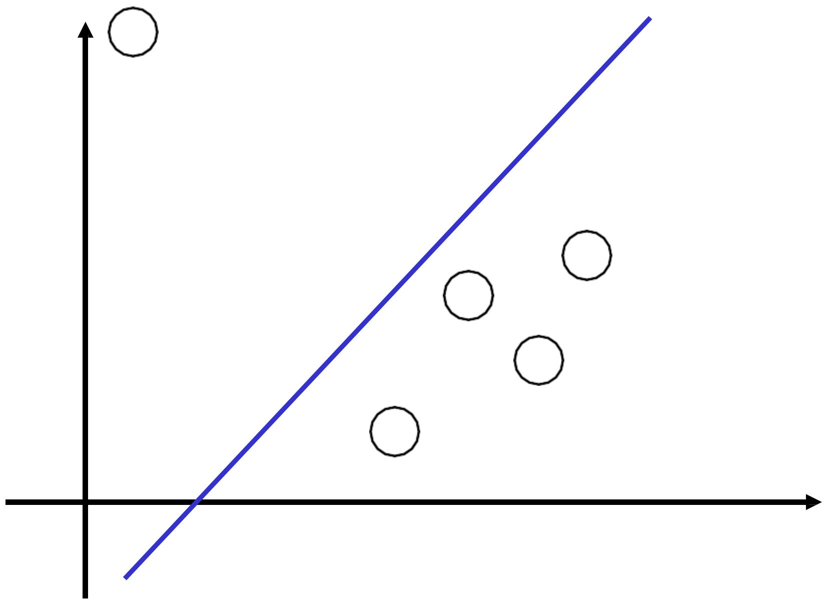
☆ A. Yes

B. No

C. It depends

# Linear regression with L1 loss

$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..n\}$$



$$\mathbf{w}^* = \arg\min_\theta \sum_{i=1}^n |y_i - f(\mathbf{x}_i; \mathbf{w})|$$

Function $\sum_{i=1}^n |y_i - f(\mathbf{x}_i; \mathbf{w})|$ is convex!

But a closed-form solution for $w^*$ doesn't exist.

# L1 Loss

$$S_{training} = \{(x_i, y_i), i = 1..n\} \qquad y_i \in \mathcal{R}$$

Obtain/train: $f(x, \mathbf{w}) = w_0 + w_1 x$

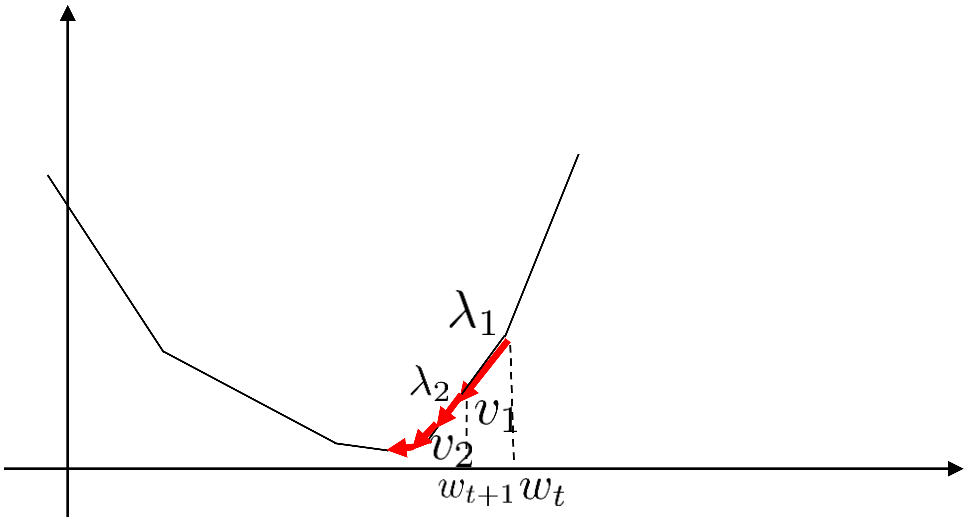
$$W^* = \arg min_W \sum_{i=1}^{n} |\mathbf{x}_i^T \cdot W - y_i| \qquad W = \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} \quad \mathbf{x}_i = \begin{pmatrix} 1 \\ x_i \end{pmatrix}$$

$$\frac{\partial |f(w)|}{\partial w} = \begin{cases} \frac{\partial f(w)}{\partial w} & f(w) > 0 \\ 0 & f(w) = 0 \\ -\frac{\partial f(w)}{\partial w} & otherwise \end{cases}$$

$$= sign(f(w)) \cdot \frac{\partial f(w)}{\partial w} \qquad sign(z) = \begin{cases} +1 & z > 0 \\ 0 & z = 0 \\ -1 & otherwise \end{cases}$$

1. Loss (Cost) Function $\quad L(W) = \sum_{i=1}^{n} |\mathbf{x}_i^T \cdot W - y_i|$
2. Obtain the gradient $\quad \frac{\partial L(W)}{\partial W} = \sum_{i=1}^{n} sign(\mathbf{x}_i^T W - y_i) \cdot \mathbf{x}_i$
3. Update parameter W $\quad W_{t+1} = W_t - \lambda_t \frac{\partial L(W)}{\partial W}$

# Gradient descent (ascent)




Gradient Method as a Line Search Method $\rightarrow$ Descent Direction

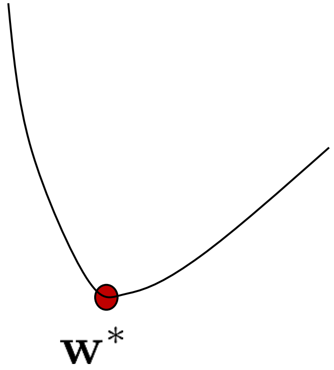
(a) Pick a direction $v_t$

(b) Pick a step size $\lambda_t$

(c) $w_{t+1} = w_t - \lambda_t \times v_t$ such that function decreases

(d) Repeat

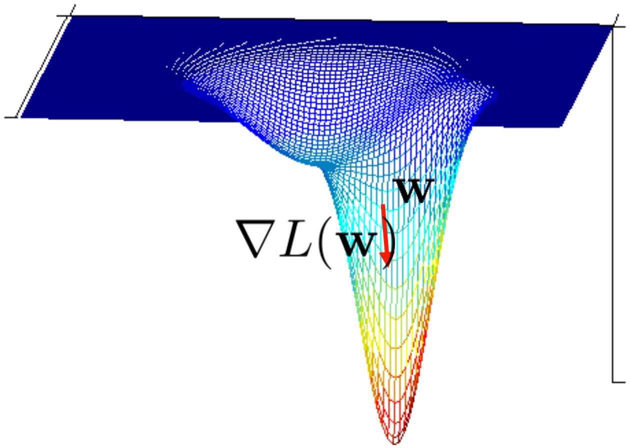
# Convex and differentiable: gradient descent



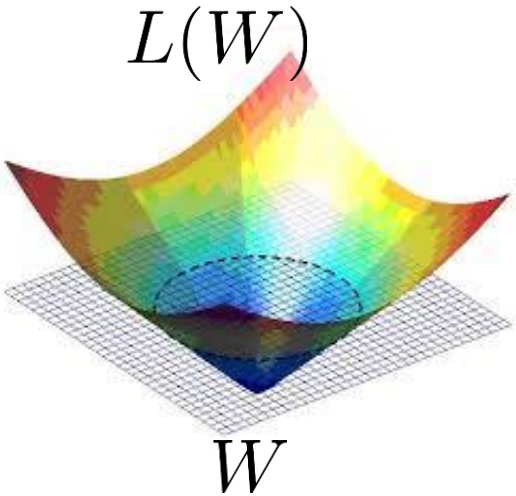

Definition:

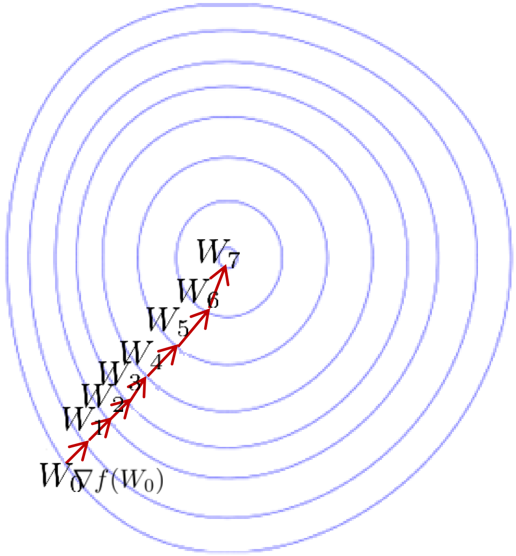
1. $\mathbf{w}^*$ is a globally optimal solution for $\mathbf{w}^* \in \Omega$ and $L(w^*) \leq L(\mathbf{w}) \forall \mathbf{w} \in \Omega$
2. $\mathbf{w}^\star$ is a locally optimal solution if there is a neighborhood $\mathcal{N}$ around $x$ such that $\mathbf{w}^\star \in \Omega$, $L(\mathbf{w}^\star) \leq L(w)$, $\forall \mathbf{w} \in \mathcal{N} \cap \Omega$.

Gradient: $\nabla L(\mathbf{w}) = [\frac{\partial L}{\partial w_i}]_{i=1,\ldots,m}$ (its a vector!)



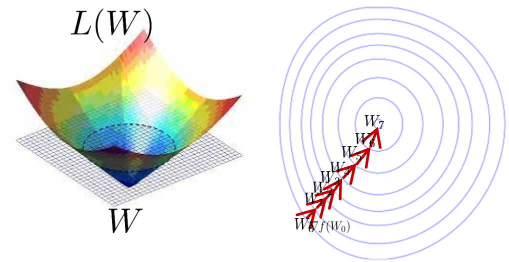
# Gradient descent




$$W_{t+1} \leftarrow W_t - \lambda_t \nabla L(W_t) \quad \lambda_t : stepsize$$

# The gradient decent algorithm




$$W_{t+1} \leftarrow W_t - \lambda_t \nabla L(W_t) \quad \lambda_t : stepsize$$

1. The gradient decent algorithm is one of the most widely used optimization methods in machine learning.
2. It can be applied to both convex and non-convex functions.
3. For non-convex functions, no guarantee to find the globally optimal solution but local optimums are ok in practice.
4. Finding the proper learning rates (not always fixed) is an important research topic for gradient decent.
5. Typically, you can start by using a small fixed learning rate when understanding the algorithm and your problem.

# L1 Loss

$$S_{training} = \{(x_i, y_i), i = 1..n\} \qquad y_i \in \mathcal{R}$$

Obtain/train: $f(x, \mathbf{w}) = w_0 + w_1 x$

$$W^* = \arg min_W \sum_{i=1}^{n} |\mathbf{x}_i^T \cdot W - y_i| \qquad W = \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} \quad \mathbf{x}_i = \begin{pmatrix} 1 \\ x_i \end{pmatrix}$$

$$\frac{\partial |f(w)|}{\partial w} = \begin{cases} \frac{\partial f(w)}{\partial w} & f(w) > 0 \\ 0 & f(w) = 0 \\ -\frac{\partial f(w)}{\partial w} & otherwise \end{cases}$$

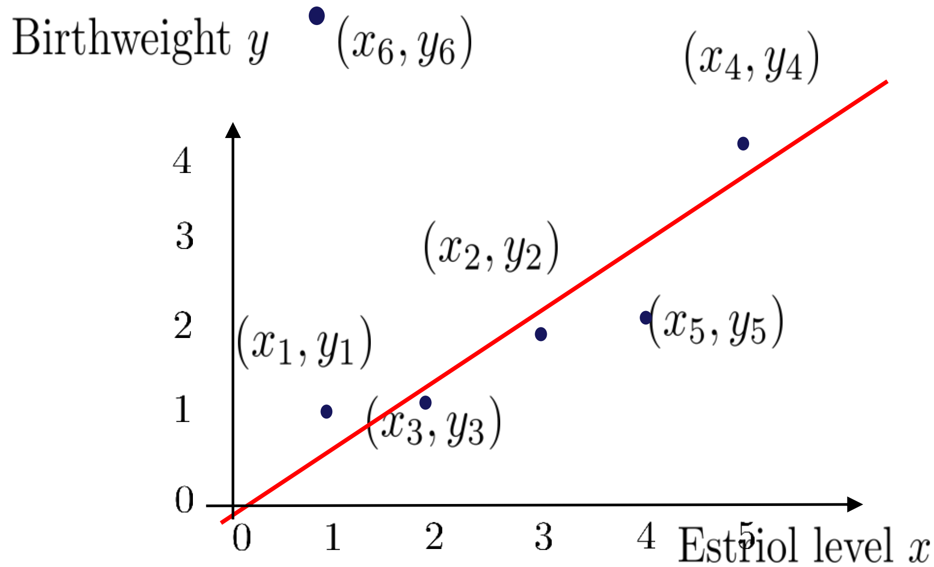
$$= sign(f(w)) \cdot \frac{\partial f(w)}{\partial w} \qquad sign(z) = \begin{cases} +1 & z > 0 \\ 0 & z = 0 \\ -1 & otherwise \end{cases}$$

1. Loss (Cost) Function $\quad L(W) = \sum_{i=1}^{n} |\mathbf{x}_i^T \cdot W - y_i|$
2. Obtain the gradient $\quad \frac{\partial L(W)}{\partial W} = \sum_{i=1}^{n} sign(\mathbf{x}_i^T \cdot W - y_i) \times \mathbf{x}_i$
3. Update parameter W $\quad W_{t+1} = W_t - \lambda_t \frac{\partial L(W)}{\partial W}$

# Robust estimation

$$Y \quad = \quad X \quad W$$

$$\begin{pmatrix} 1 \\ 1.9 \\ 1.05 \\ 4.1 \\ 2.1 \\ \color{red}{6.0} \end{pmatrix} \begin{pmatrix} 1, 1 \\ 1, 3 \\ 1, 2 \\ 1, 5 \\ 1, 4 \\ \color{red}{1, 1.1} \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \end{pmatrix}$$




1. Loss (Cost) Function $\quad W^* = \arg min_W \sum_i |\mathbf{x}_i^T \cdot W - y_i|$
2. Obtain the gradient $\quad \frac{\partial L(W)}{\partial W} = \sum_{i=1}^n sign(\mathbf{x}_i^T \cdot W - y_i) \times \mathbf{x}_i$
3. Update parameter W $\quad W_{t+1} = W_t - \lambda_t \frac{\partial L(W)}{\partial W}$

# L1

$$S_{training} = \{(x_i, y_i), i = 1..n\}$$

E.g. $S_{training} = \{(x_i, y_i), i = 1..n\}$
$= \{(1,1), (3,1.9), (2,1.05), (5,4.1), (4,2.1)\}$

$$X = \begin{pmatrix} 1,1 \\ 1,3 \\ 1,2 \\ 1,5 \\ 1,4 \end{pmatrix} \qquad Y = \begin{pmatrix} 1 \\ 1.9 \\ 1.05 \\ 4.1 \\ 2.1 \end{pmatrix}$$

Obtain/train: $f(x, \mathbf{w}) = w_0 + w_1 x$

$$W^* = \arg min_W \sum_{i=1}^{n} |\mathbf{x}_i^T \cdot W - y_i|$$

$$W = \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} \qquad \mathbf{x}_i = \begin{pmatrix} 1 \\ x_i \end{pmatrix}$$

1. Loss (Cost) Function

$$L(W) = \sum_{i=1}^{n} |\mathbf{x}_i^T \cdot W - y_i|$$

2. Obtain the gradient

$$\frac{\partial L(W)}{\partial W} = \sum_{i=1}^{n} sign(\mathbf{x}_i^T \cdot W - y_i) \times \mathbf{x}_i$$

3. Update parameter W

$$W_{t+1} = W_t - \lambda_t \frac{\partial L(W)}{\partial W}$$

What if we want both
L1 Loss and Squared Loss

$$W^* = \arg\min_W \sum_{i=1}^{n} (\mathbf{x}_i^T \cdot W - y_i)^2$$

$$W^* = \arg min_W \sum_{i=1}^{n} |\mathbf{x}_i^T \cdot W - y_i|$$

## Squared Loss
# Learning Linear Regression Using Gradient Descent Instead

$$S_{training} = \{(x_i, y_i), i = 1..n\}$$

E.g. $S_{training} = \{(x_i, y_i), i = 1..n\}$
$= \{(1,1), (3,1.9), (2,1.05), (5,4.1), (4,2.1)\}$

$$X = \begin{pmatrix} 1,1 \\ 1,3 \\ 1,2 \\ 1,5 \\ 1,4 \end{pmatrix} \qquad Y = \begin{pmatrix} 1 \\ 1.9 \\ 1.05 \\ 4.1 \\ 2.1 \end{pmatrix}$$

Obtain/train: $f(x, \mathbf{w}) = w_0 + w_1 x$

$$W = \arg\min_W \sum_{i=1}^{n} (\mathbf{x}_i^T \cdot W - y_i)^2$$

$$W = \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} \qquad \mathbf{x}_i = \begin{pmatrix} 1 \\ x_i \end{pmatrix}$$

1. Loss (Cost) Function

$$L(W) = \sum_{i=1}^{n} (\mathbf{x}_i^T \cdot W - y_i)^2$$

2. Obtain the gradient

$$\frac{\partial L(W)}{\partial W} = \sum_{i=1}^{n} 2 \times (\mathbf{x}_i^T \cdot W - y_i) \times \mathbf{x}_i$$

3. Update parameter W

$$W_{t+1} = W_t - \lambda_t \frac{\partial L(W)}{\partial W}$$

# L1+Square Loss

$$S_{training} = \{(x_i, y_i), i = 1..n\}$$

E.g. $S_{training} = \{(x_i, y_i), i = 1..n\}$

$= \{(1,1), (3,1.9), (2,1.05), (5,4.1), (4,2.1)\}$

$$X = \begin{pmatrix} 1,1 \\ 1,3 \\ 1,2 \\ 1,5 \\ 1,4 \end{pmatrix} \qquad Y = \begin{pmatrix} 1 \\ 1.9 \\ 1.05 \\ 4.1 \\ 2.1 \end{pmatrix}$$

Obtain/train: $f(x, \mathbf{w}) = w_0 + w_1 x$

$$W = \arg\min_W \sum_{i=1}^{n} [(\mathbf{x}_i^T \cdot W - y_i)^2 + \alpha |\mathbf{x}_i^T \cdot W - y_i|]$$

$$W = \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} \qquad \mathbf{x}_i = \begin{pmatrix} 1 \\ x_i \end{pmatrix}$$

1. Loss (Cost) Function

$$L(W) = \sum_{i=1}^{n} [(\mathbf{x}_i^T \cdot W - y_i)^2 + \alpha |\mathbf{x}_i^T \cdot W - y_i|]$$

2. Obtain the gradient

$$\frac{\partial L(W)}{\partial W} = \sum_{i=1}^{n} [2 \times (\mathbf{x}_i^T \cdot W - y_i) + \alpha \times sign(\mathbf{x}_i^T \cdot W - y_i)] \times \mathbf{x}_i$$

3. Update parameter W

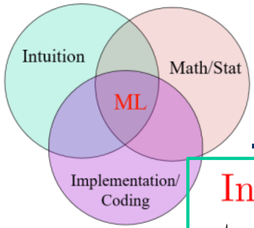
$$W_{t+1} = W_t - \lambda_t \frac{\partial L(W)}{\partial W}$$

Take home message

- The formulation/problem of regression (e.g. linear) is separate from the solution (e.g. least-square solution itself).

- You can solve a least square estimation problem using a closed form solution (least square solution/fitting) or the gradient descent algorithm.

- Using robust statistics e.g. L1 using the gradient descent algorithm opens up a wide range of direction for the regression problem.




# Recap: Robust Estimation

**Intuition**: Linear (polynomial) regression can be learned robustly when the training data include outliers that mislead the training algorithm. This can be achieved by adopting the L1 for computing the loss between the ground-true value and the prediction. There is no closed (analytical) solution when learning a robust linear regression model and the training is typically done using the gradient descent algorithm.

**Math**:

$$\begin{array}{ccccc} Y & & X & W \\ \begin{pmatrix} 1 \\ 1.9 \\ 1.05 \\ 4.1 \\ 2.1 \end{pmatrix} & = & \begin{pmatrix} 1, 1, 0.5 \\ 1, 3, 0.9 \\ 1, 2, 1.0 \\ 1, 5, 6.7 \\ 1, 4, 2.5 \end{pmatrix} & \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix} \end{array}$$

$$W^* = \arg min_W \sum_{i=1}^{n} |\mathbf{x}_i^T \cdot W - y_i|$$

$$\frac{\partial L(W)}{\partial W} = \sum_{i=1}^{n} sign(\mathbf{x}_i^T W - y_i) \cdot \mathbf{x}_i$$

$$W_{t+1} = W_t - \lambda_t \frac{\partial L(W)}{\partial W}$$

**Implementation**:

1. Initilize $W$ at random.
2. Compute the gradient $\frac{\partial L(W)}{\partial W} = \sum_{i=1}^{n} sign(\mathbf{x}_i^T W - y_i) \cdot \mathbf{x}_i$.
3. Update $W$ by: $W \leftarrow W - \lambda \frac{\partial L(W)}{\partial W}$.
4. Go to step 2 if $\frac{\partial L(W)}{\partial W}$ is large, otherwise stop.

Metrics and evaluation: from intuition to a sound

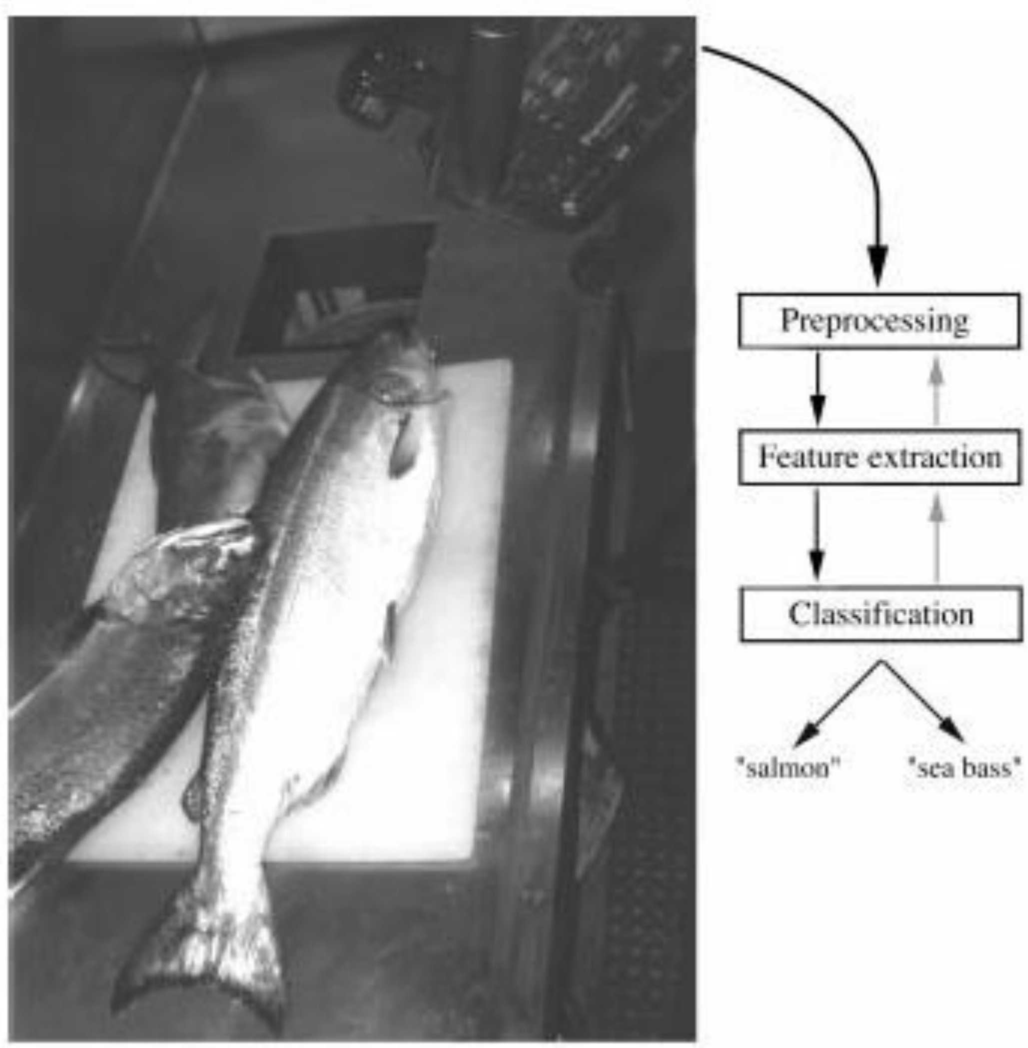
mathematical formulation

with a

clear objective.

A main difference between the early AI and the modern ML systems.

# An example


Preprocessing
Feature extraction
Classification
"salmon"
"sea bass"

# Error Metrics and Object Functions

- One thing that separates modern machine learning from the efforts in traditional AI is the establishment of benchmarks under widely accepted common evaluation metrics.

- Being able to faithfully compare the performances of different machine learning algorithms/systems significantly propel the advancement of machine learning field.

- In addition, establishing a clear objective function (errors + regularization) to optimize when training machine learning algorithms is a key reason for the success of modern machine/deep learning.

# Summary of the problem

$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..n\}$$

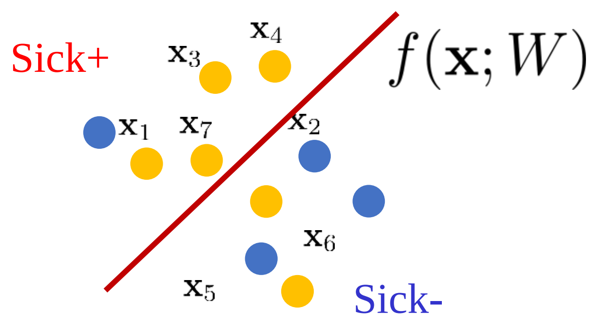
$\mathbf{x} = (x_1, ..., x_m), x_i \in \mathbb{R}, \quad \mathbf{x} \in \mathbb{R}^m$

$y \in \{-1, +1\}$ $\quad$ $y = -1$: sick-

$y = +1$: sick+

Classifier: $Classify = f(\mathbf{x}; W) \in \{-1, +1\}$

Model parameter to be learned: $W$




Training error:

$$e_{training} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(y_i \neq f(\mathbf{x}_i; W))$$

$$e_{training} = \frac{3}{10} = 0.3$$

# Summary of the problem

$$S_{testing} = \{(\mathbf{x}_i, y_i), i = 1..q\}$$

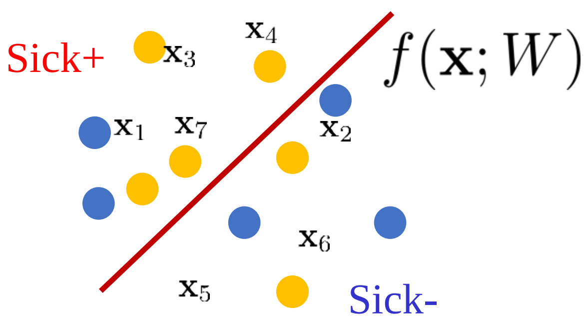
$\mathbf{x} = (x_1, ..., x_m), x_i \in \mathbb{R}, \quad \mathbf{x} \in \mathbb{R}^m$

$y \in \{-1, +1\}$ $\quad y = -1$: sick-

$y = +1$: sick+

Classifier: $Classify = f(\mathbf{x}; W) \in \{-1, +1\}$

Learned model parameter: $W$




Testing error:

$$e_{testing} = \frac{1}{q} \sum_{i=1}^{q} \mathbf{1}(y_i \neq f(\mathbf{x}_i; W))$$

$$e_{testing} = \frac{4}{11} = 0.3636$$

# Some Learned Lessons (Pedro Domingos)

**Table 1. The three components of learning algorithms.**

| Representation | Evaluation | Optimization |
|---|---|---|
| Instances | Accuracy/Error rate | Combinatorial optimization |
| K-nearest neighbor | Precision and recall | Greedy search |
| Support vector machines | Squared error | Beam search |
| Hyperplanes | Likelihood | Branch-and-bound |
| Naive Bayes | Posterior probability | Continuous optimization |
| Logistic regression | Information gain | Unconstrained |
| Decision trees | K-L divergence | Gradient descent |
| Sets of rules | Cost/Utility | Conjugate gradient |
| Propositional rules | Margin | Quasi-Newton methods |
| Logic programs | | Constrained |
| Neural networks | | Linear programming |
| Graphical models | | Quadratic programming |
| Bayesian networks | | |
| Conditional random fields | | |

# Why not to use the classification "error" all the time.

Classification error: $e = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(y_i \neq f(\mathbf{x}_i; W))$

When your training dataset is balanced, e.g. the same number of positives and negatives, then the classification error seems to be a sound metric.

In practice, the given datasets are often unbalanced.

$$S = \{(\mathbf{x}_i, y_i), i = 1..n\}$$

much greater

Often: $\underbrace{\boxed{\sum_{i=1}^{n} \mathbf{1}(y_i = 0))}}_{\#\text{ of negatives}} >> \underbrace{\boxed{\sum_{i=1}^{n} \mathbf{1}(y_i = 1))}}_{\#\text{ of positives}}$

For example, if we have **1,000** negative samples and 1 positive sample, we can blindly classify (a trivial solution) every input sample as negative:

Classification error: $e = \frac{1}{1{,}001}$ Seeming low, but misleading!

# Error measures and metrics

| | classify+ | classify- |
|---|---|---|
| true label+ | true label+ and classify+ | true label+ and classify- |
| true label- | true label- and classify+ | true label- and classify- |

larger preferred

True positive rate: P(classify + | true label + )

$$= sensitivity = recall$$

smaller preferred

False positive rate: P(classify+ | true label -)

larger preferred

True negative rate: P(classify - | true label-)

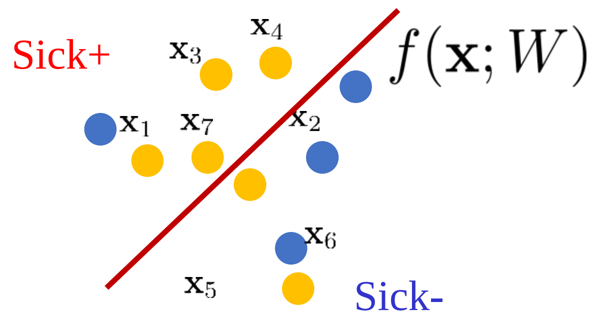
$$= specificity$$

smaller preferred

False negative rate: P(classify- | true label +)

# How to compute the errors

$\mathbf{x} = (x_1, ..., x_m), x_i \in \mathbb{R}, \quad \mathbf{x} \in \mathbb{R}^m \quad y \in \{-1, +1\} \quad y = -1\text{: sick-} \quad y = +1\text{: sick+}$




Given: $S_{training} = \{(\mathbf{x}_i, y_i), i = 1..100\}$

Classification (Classify): $f(\mathbf{x}; W) \in \{-1, +1\}$

| $\mathbf{x}$ (features) | y (sick- or sick+?) | $f(\mathbf{x}; W)$ (classify- or classify +?) |
|---|---|---|
| $\mathbf{x}_1$ | $-1$ | $-1$ |
| $\mathbf{x}_2$ | $-1$ | $+1$ |
| $\mathbf{x}_3$ | $+1$ | $+1$ |
| $\mathbf{x}_4$ | $-1$ | $-1$ |
| • | • | • |
| $\mathbf{x}_{100}$ | $+1$ | $-1$ |

## Summary

(confusion matrix)

| $f(\mathbf{x}; W)$ \ $y$ | **sick +** | **sick -** | **Total** |
|---|---|---|---|
| **Classify +** | 30 | 10 | 40 |
| **Classify -** | 10 | 50 | 60 |
| **Total** | 40 | 60 | 100 |

# Error measures

## Summary

(confusion matrix)

| $f(\mathbf{x}; W)$ \ $y$ | **sick +** | **sick -** | **Total** |
|---|---|---|---|
| **classify +** | 30 | 10 | 40 |
| **classify -** | 10 | 50 | 60 |
| **Total** | 40 | 60 | 100 |

$y$: ground truth labels

$f(\mathbf{x}; W)$: prediction

- Having faithful evaluation measures is critical in the success of machine learning.
- Computing the "error" is not unique.

# Error Metrics and Evaluation

| | | | | | |
|---|---|---|---|---|---|
| | | **Condition (as determined by "Gold standard")** | | | |
| | Total population | Condition positive | Condition negative | Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$ | |
| **Test outcome** | Test outcome positive | **True positive** | **False positive** (Type I error) | Positive predictive value (PPV, Precision) = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Test outcome positive}}$ | False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Test outcome positive}}$ |
| | Test outcome negative | **False negative** (Type II error) | **True negative** | False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Test outcome negative}}$ | Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Test outcome negative}}$ |
| | Positive likelihood ratio (**LR+**) = TPR/FPR | True positive rate (TPR, Sensitivity, Recall) = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$ | False positive rate (FPR, Fall-out) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$ | Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$ | |
| | Negative likelihood ratio (**LR−**) = FNR/TNR | False negative rate (FNR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$ | True negative rate (TNR, Specificity, SPC) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$ | | |
| | Diagnostic odds ratio (**DOR**) = LR+/LR− | | | | |

Classification result is denoted as “Test outcome” here.

https://en.wikipedia.org/wiki/Precision_and_recall

# Error measures and metrics

| | classify+ | classify- |
|---|---|---|
| sick+ | sick+ and classify+ | sick+ and classify- |
| sick- | sick- and classify+ | sick- and classify- |

larger preferred

True positive rate: P(classify + | sick + )

$$= sensitivity = recall$$

smaller preferred

False positive rate: P(classify+ | sick - )

larger preferred

True negative rate: P(classify - | sick-)

$$= specificity$$

smaller preferred

False negative rate: P(classify- | sick +)

# Intuition Test

Summary (confusion matrix)

| $f(\mathbf{x}; W)$ \ $y$ | **sick +** | **sick -** | **Total** |
|---|---|---|---|
| **classify +** | 30 | 10 | 40 |
| **classify -** | 10 | 50 | 60 |
| **Total** | 40 | 60 | 100 |

A. High sensitivity, low specificity

B. High sensitivity, high specificity

C. Low sensitivity, low specificity

D. Low sensitivity, high specificity

# Intuition Test

Summary (confusion matrix)

| $f(\mathbf{x};W)$ \ $y$ | **sick +** | **sick -** | **Total** |
|---|---|---|---|
| **classify +** | 30 | 10 | 40 |
| **classify -** | 10 | 50 | 60 |
| **Total** | 40 | 60 | 100 |

A. High sensitivity, low specificity

B. High sensitivity, high specificity

C. Low sensitivity, low specificity

D. Low sensitivity, high specificity

# Error measures

## Summary (confusion matrix)

| $f(\mathbf{x};W)$ \ $y$ | **sick +** | **sick -** | **Total** |
|---|---|---|---|
| **classify +** | 30 | 10 | 40 |
| **classify -** | 10 | 50 | 60 |
| **Total** | 40 | 60 | 100 |

True positive rate: P(classify+ | sick +) $= \frac{30}{40} = 0.75$

$= sensitivity = recall$

False positive rate: P(classify + | sick - ) $= \frac{10}{60} = 0.167$

True negative rate: P(classify - | sick-) $= \frac{50}{60} = 0.833$

$= specificity$

False negative rate: P(classify - | sick +) $= \frac{10}{40} = 0.25$