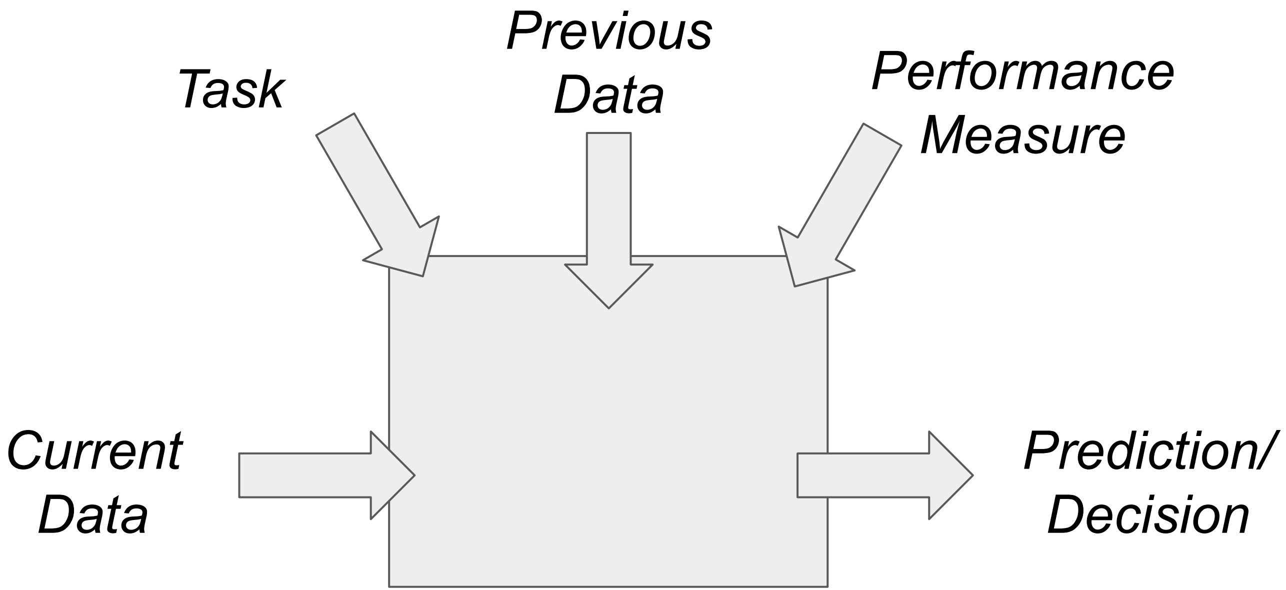
5
Classification

Amal Aboulhassan

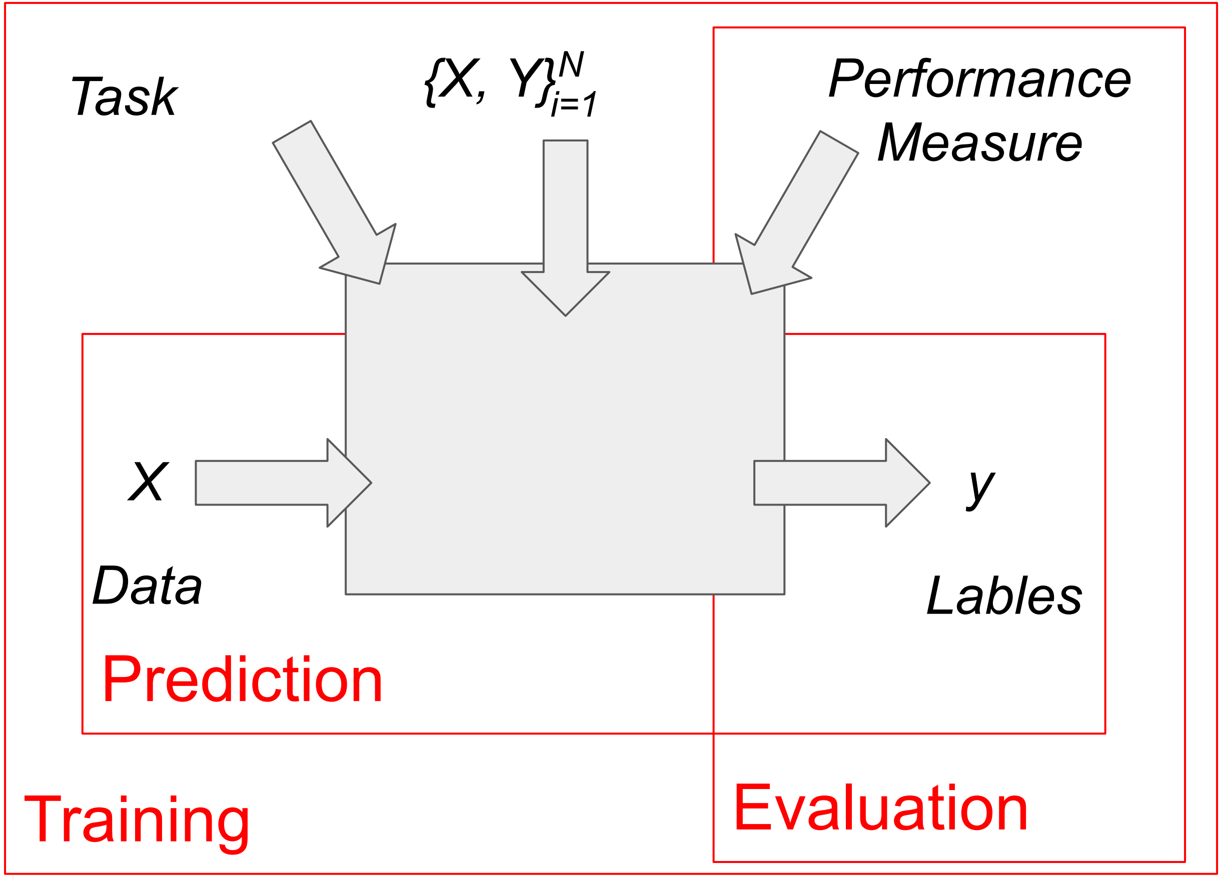
Machine Learning Taxonomy

- Supervised Learning
  - Regression
  - Classification
- Unsupervised Learning
  - Clustering
  - Others
- Reinforcement Learning

Machine Learning Process



Supervised Learning Process



Logistic Regression

- Logistic Regression: estimates the probability that a point belongs to a certain class
- In binary classification, we use two terms:
  - Positive class: the class under question. For example (heart attack class). Labeled as “1”
  - Negative class: the other class. For example (no-heart attack class). Labeled as “0”
- If estimated probability is >0.5, then the model suggests that the the point belongs to the positive class “1”. If it is <0.5, then it suggests the point belongs to the negative class “0

Types of Binary Classification

Binary Prediction

Goal: Predict label (0 or 1) given features x

- Input: $x_i \triangleq [x_{i1}, x_{i2}, \dots x_{if} \dots x_{iF}]$

"features"
"covariates"
"attributes"

Entries can be real-valued, or other numeric types (e.g. integer, binary)

- Output: $y_i \in \{0, 1\}$

"responses" or "labels"

Binary label (0 or 1)

```
>>> yhat_N = model.predict(x_NF)
>>> yhat_N[:5]
[0, 0, 1, 0, 1]
```

Heart attack example in the previous lecture

Types of Binary Classification

Probability Prediction

Goal: Predict probability of label given features

- Input: $x_i \triangleq [x_{i1}, x_{i2}, \ldots x_{if} \ldots x_{iF}]$

"features"
"covariates"
"attributes"

Entries can be real-valued, or other numeric types (e.g. integer, binary)

- Output: $\hat{p}_i \triangleq p(Y_i = 1 | x_i)$

"probability"

Value between 0 and 1
e.g. 0.001, 0.513, 0.987

```
>>> yproba_N2 = model.predict_proba(x_NF)
>>> yproba1_N = model.predict_proba(x_NF)[:,1]
>>> yproba1_N[:5]
[0.143, 0.432, 0.523, 0.003, 0.994]
```

Cancer data in the homework

Logistic Regression Classifier

Parameters:

weight vector $w = [w_1, w_2, \ldots w_f \ldots w_F]$

bias scalar b

Prediction:

$$\hat{p}(x_i, w, b) = p(y_i = 1 | x_i) \triangleq \text{sigmoid}\left(\sum_{f=1}^{F} w_f x_{if} + b\right)$$

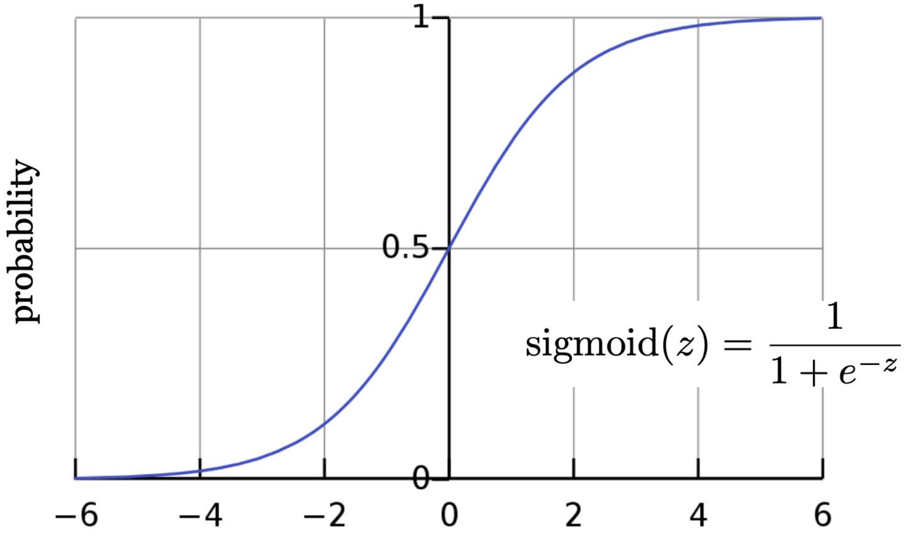
Training:

find weights and bias that minimize error

Logistic Regression Classifier

Logistic Sigmoid Function

Goal: Transform real values into probabilities



$$\hat{y} = \begin{cases} 0 & \text{if } \hat{p} < 0.5, \\ 1 & \text{if } \hat{p} \geq 0.5. \end{cases}$$

z_N = x_N * w

Exponential Function: $f(x) = e^x$
e (2.7182818…)

Logistic Regression: Training

Optimization: Minimize total log loss on train set

$$\min_{w,b} \sum_{n=1}^{N} \text{log\_loss}(y_n, \hat{p}(x_n, w, b))$$

Algorithm: Gradient descent

Avoid overfitting: Use L2 or L1 penalty on weights

The logarithmic Function

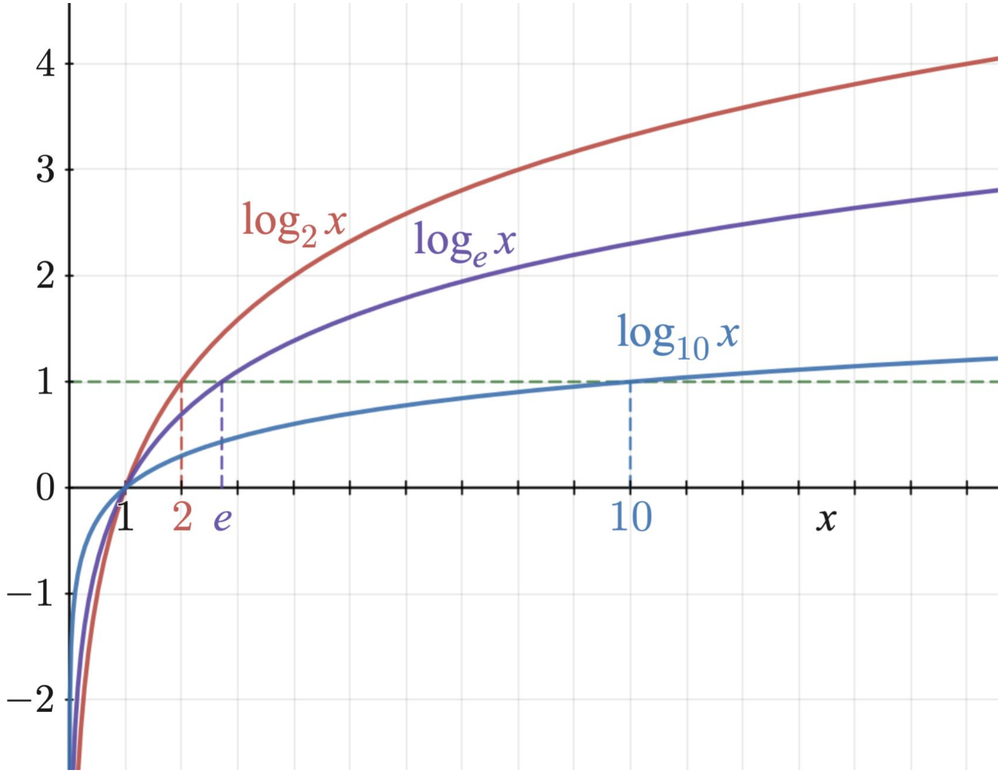
- The logarithm is the inverse function to exponentiation
- Example :
  - $1000 = 10^3$
  - What is the logarithm base 10 of 1000 ($\log_{10}(1000)$)?

The logarithmic Function

- The logarithm is the inverse function to exponentiation
- Example :
  - $1000 = 10^3$
  - What is the logarithm base 10 of 1000 ($\log_{10}(1000)$)? 3

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

The logarithmic Function



The logarithmic Function

- $\log_2 16 = 4$, since $2^4 = 2 \times 2 \times 2 \times 2 = 16$.
- Logarithms can also be negative: $\log_2 \frac{1}{2} = -1$ since $2^{-1} = \frac{1}{2^1} = \frac{1}{2}$.
- $\log_{10} 150$ is approximately 2.176, which lies between 2 and 3, just as 150 lies between $10^2 = 100$ and $10^3 = 1000$.
- For any base b, $\log_b b = 1$ and $\log_b 1 = 0$, since $b^1 = b$ and $b^0 = 1$, respectively.

Evaluation: Error Function

- Why (-ve) log?
- What are the values with high probabilities close to 1?
- What are the values with low probabilities closer to zero?

$$c(\theta) = \begin{cases} -\log(\hat{p}) & \text{if } y = 1, \\ -\log(1 - \hat{p}) & \text{if } y = 0. \end{cases}$$

Evaluation: Error Function

- Why (-ve) log?
- What are the values with high probabilities close to 1?
- What are the values with low probabilities closer to zero?
- If an instance belongs to the positive class, but the prediction generated small probability → big error → big log

$$c(\theta) = \begin{cases} -\log(\hat{p}) & \text{if } y = 1, \\ -\log(1-\hat{p}) & \text{if } y = 0. \end{cases}$$

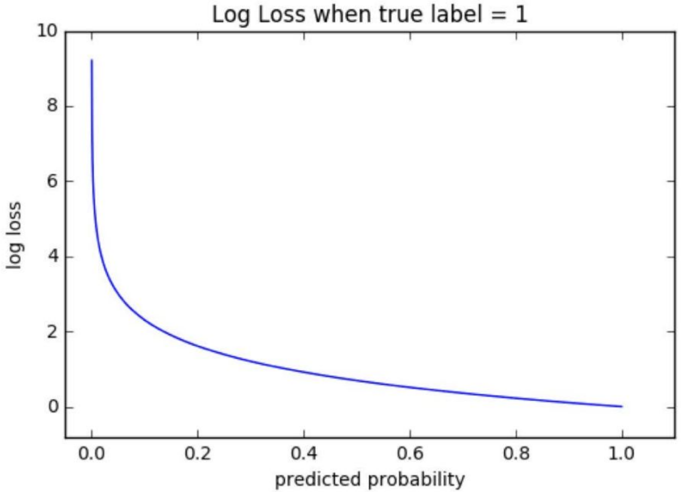
Evaluation: Error Function

Log loss (aka "binary cross entropy")

from sklearn.metrics import log_loss

$$\text{log\_loss}(y, \hat{p}) = -y \log \hat{p} - (1-y) \log(1-\hat{p})$$

- High probability, y = 1
- Low probability, y=1
- High probability, y=0
- Low probability, y=0



Lower is better!

Advantages:
- smooth
- easy to take derivatives!

Evaluation: Cost Function

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}log(\hat{p}^{(i)}) + (1-y^{(i)})log(1-\hat{p}^{(i)})]$$

$$\min_{w,b} \sum_{n=1}^{N} \text{log\_loss}(y_n, \hat{p}(x_n, w, b))$$

Gradient descent for L2 penalized LR

$$\min_{\boldsymbol{w},w_0} \boxed{\underbrace{-\sum_i \log\ p(y_i\,|\boldsymbol{x}_i;\boldsymbol{w},w_0)}_{J(w,\,w_0)}} + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$$

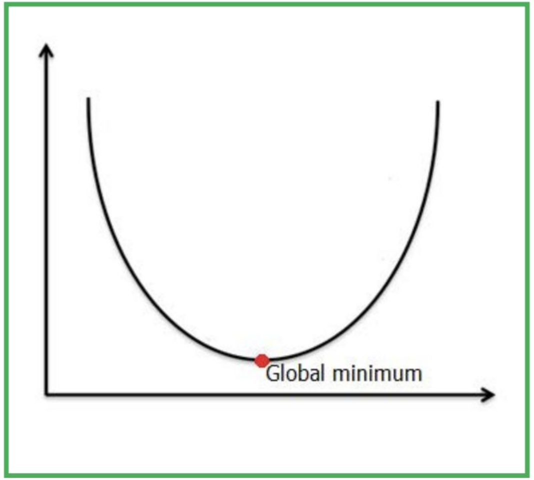
Start with $\boldsymbol{w}^0 = 0, w_0^0 = 0$, step size s
for $t = 0, \dots, (T-1)$
$\quad \boldsymbol{w}^{t+1} = \boldsymbol{w}^t - s\,\nabla J(\boldsymbol{w}^t, w_0^t) - \lambda\boldsymbol{w}^t$
$\quad w_0^{t+1} = w_0^t - s\,\nabla J(\boldsymbol{w}^t, w_0^t)$

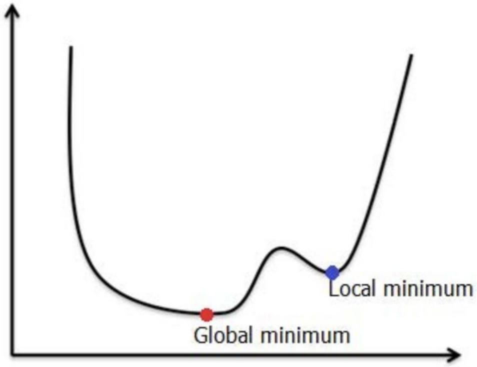
$\quad$ if $L(\boldsymbol{w}^{t+1}, w_0^{t+1}) - L(\boldsymbol{w}^t, w_0^t) < \delta$
$\qquad$ break
return $\boldsymbol{w}^T, w_0^T$

Mike Hughes - Tufts COMP 135 - Spring 2019

Evaluation: Cost Function

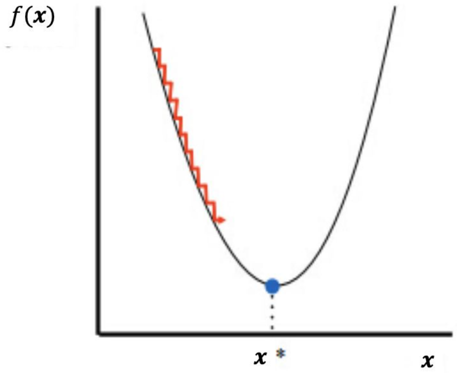
Log loss is convex!



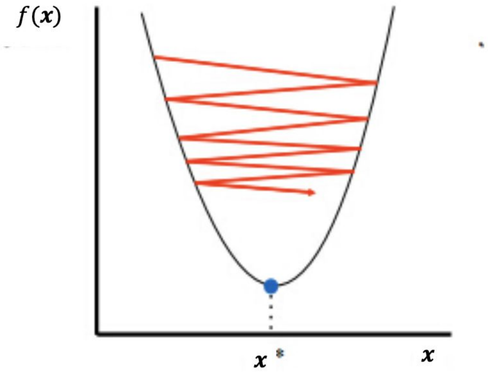


Evaluation: Cost Function

Intuition: 1D minimization



Too small: converge very slowly



Too big: overshoot and even diverge

Evaluation: Cost Function

Rule for picking step sizes

- Never try just one!
- Try several values (exponentially spaced) until
  - Find one clearly too small
  - Find one clearly too large (oscillation / divergence)
- Always make trace plots!
  - Show the loss, norm of gradient, and parameters

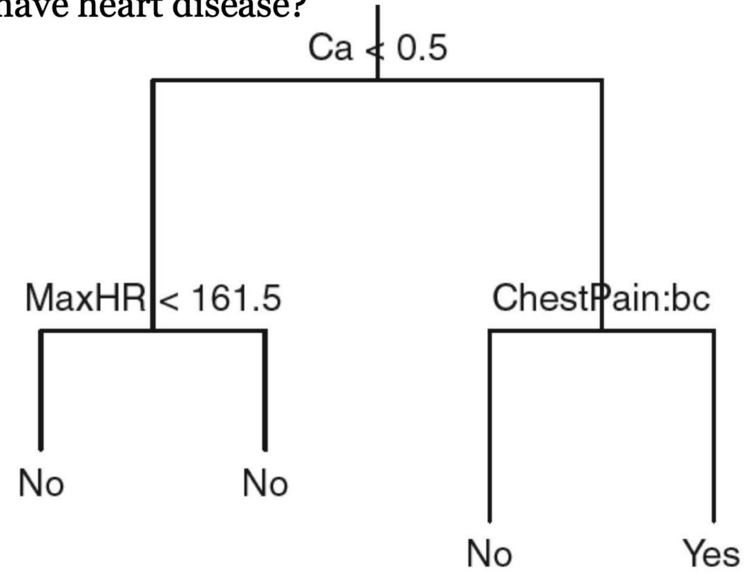
- Smarter choices for step size:
  - Decaying methods
  - Search methods
  - Adaptive methods

Evaluation: Cost Function

- Common methods
  - Decay over iterations
  - Line Search scipy.optimize.line_search

Decision Tree

Goal: Does patient have heart disease?



Leaves make binary predictions! *(but can be made probabilistic)*

Decision Tree

Parameters:

- at each internal node: x variable id and threshold
- at each leaf: probability of positive y label

Prediction:

- identify rectangular region for input x
- predict: most common y value in region
- predict_proba: report fraction of each label in region

Training:

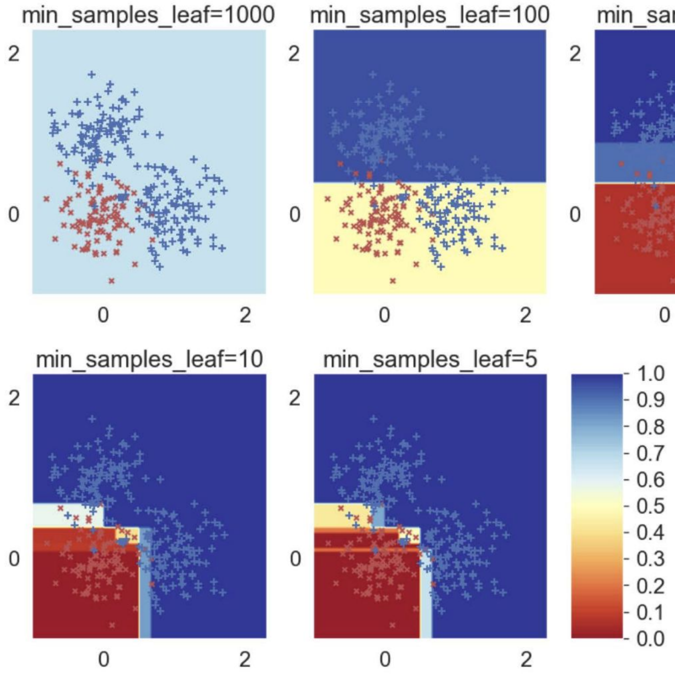
- minimize error on training set
- often, use greedy heuristics

Decision Tree Example

https://www.youtube.com/watch?v=_L39rN6gz7Y

Decision Tree

Decision Tree: Predicted Probas



Pretty flexible!

Function is piecewise constant and axis-aligned

Slide Credits

- Some slides credited to Mike Hughes - Tufts
- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
- https://scikit-learn.org/stable/modules/generated/sklearn.metrics.log_loss.html
-

Questions!