# Logistic Regression

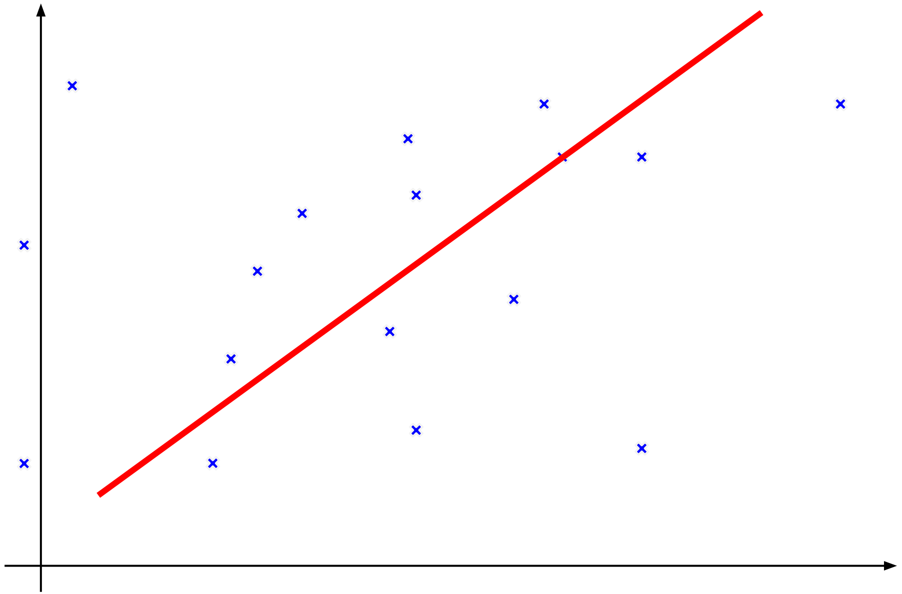
Doru Arfire

*Papers we love X, June 23rd 2016*

# Contents

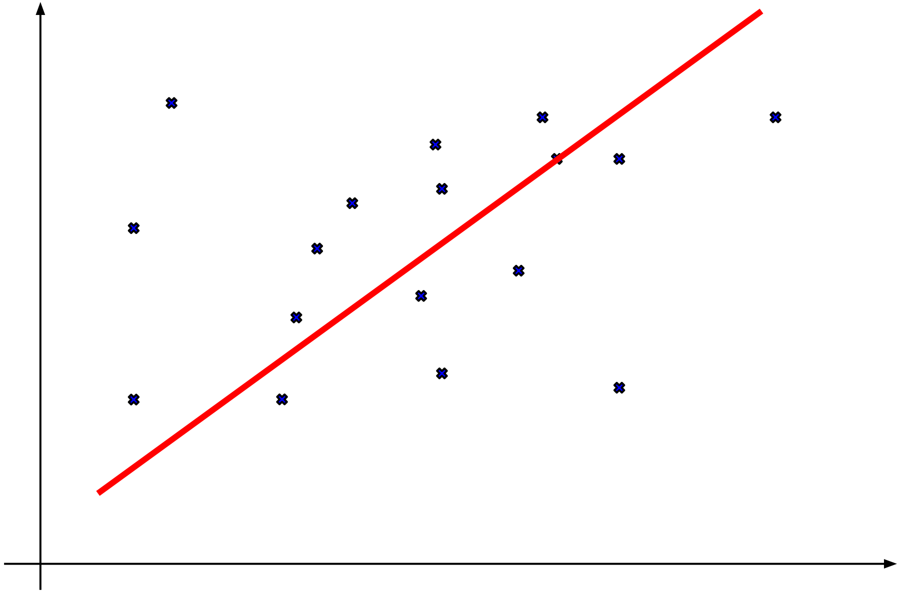
# Ordinary Least Squares

- Given real values $\mathbf{X}_{NxK}$ and $\mathbf{y}_{N}$, train model that can predict $\mathbf{y'}$ from $\mathbf{X'}$
- **X** - independent variables
- **y** - dependent variable
- Assumes $\mathbf{y}_i = \mathbf{w}^T\mathbf{x}_i + \mathbf{w}_0 + \boldsymbol{\varepsilon}_i$
- **ε** - error terms, assumed to have E [**ε**] = 0
- Columns (regressors) of **X** linearly independent



# Ordinary Least Squares

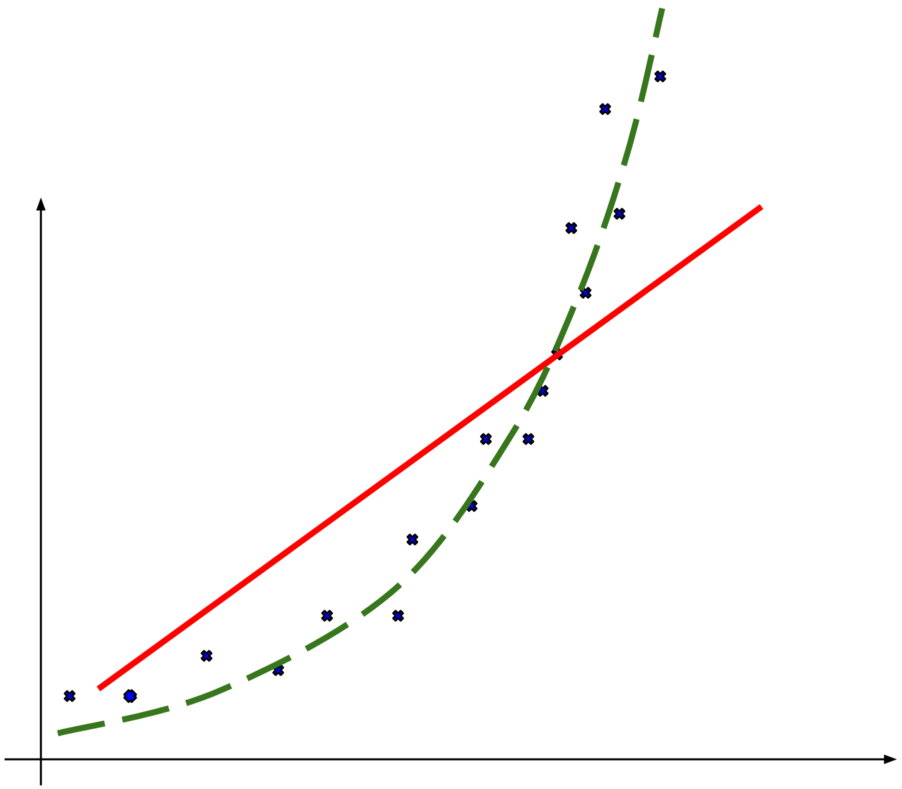
- We're learning model given by **$f(x) = w^T x$**
- **$x_i = [1, x_{i1}, x_{i2}, \ldots, x_{iK}]^T$**
- **$w = [w_0, w_1, w_2, \ldots, w_K]^T$**
- **$w_0$** is called the *intercept*
- We find **w** by minimizing **$||f(x) - y||^2$**



$$w^* = \arg\max_{w} \sum_{i=1}^{N} (y_i - w^T x_i)^T (y_i - w^T x_i)$$

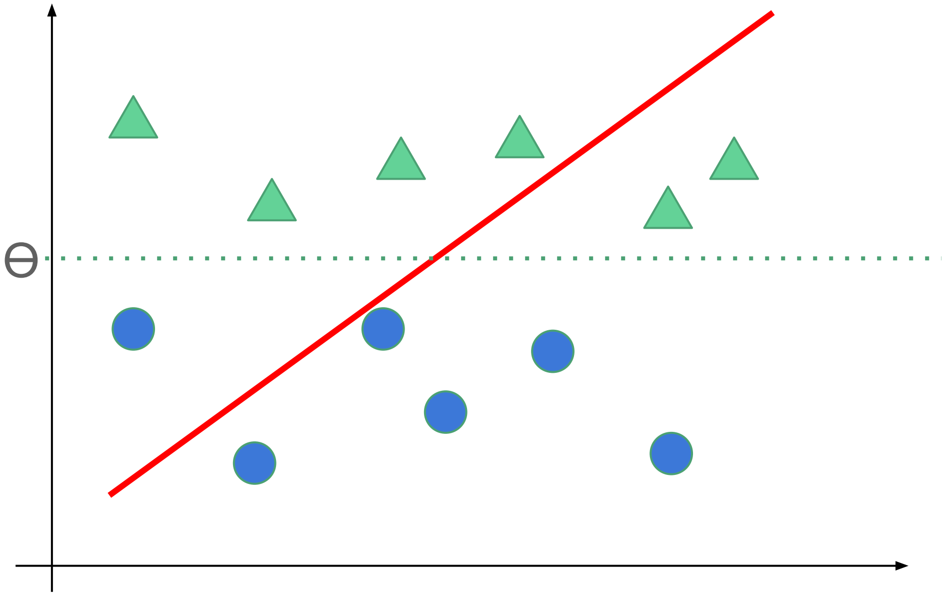
# OLS: beyond linearity

- Sometimes we suspect our input variables to have non-linear interactions
- E.g., y depends not on $x_1$, $x_2$, etc., but on $x_1^2$, $x_1x_2$, etc.
- Take advantage of linear relationship between y and $x_1^2$, $x_1x_2$
- Add extra variables $z_1 = x_1^2$, $z_2 = x_1x_2$, etc.

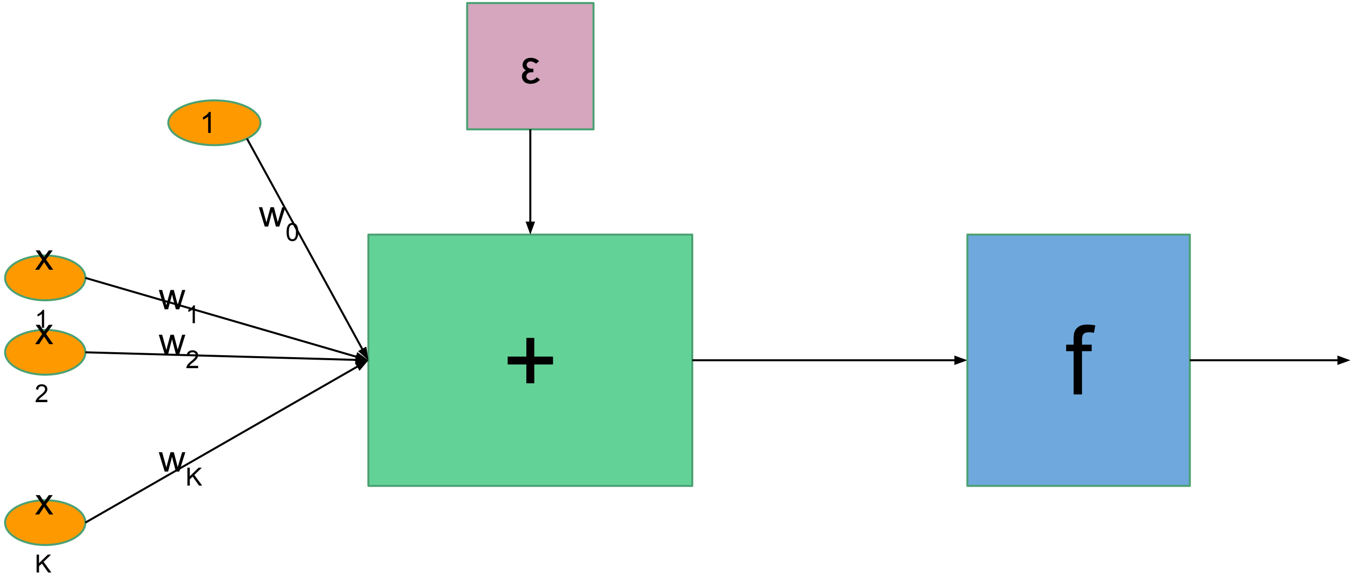


# Logistic Regression: Introduction

- Linear regression is good for, well, *regression*
- Unnatural fit for *classification*: given **X**, predict **y** ∈ {0, 1}
- You could have a threshold ϴ; anything above is 1, anything below is -1
- Difficult to interpret **f(x)** as degree of certainty (what is f(x) = 10, or -200)
- How do we find ϴ



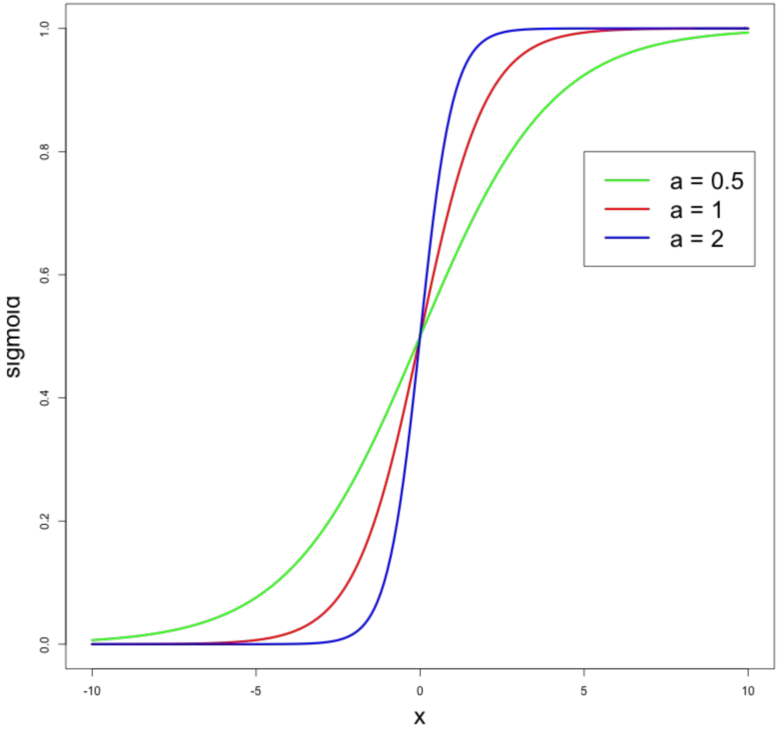
# Generalized Linear Model



# Logistic Function

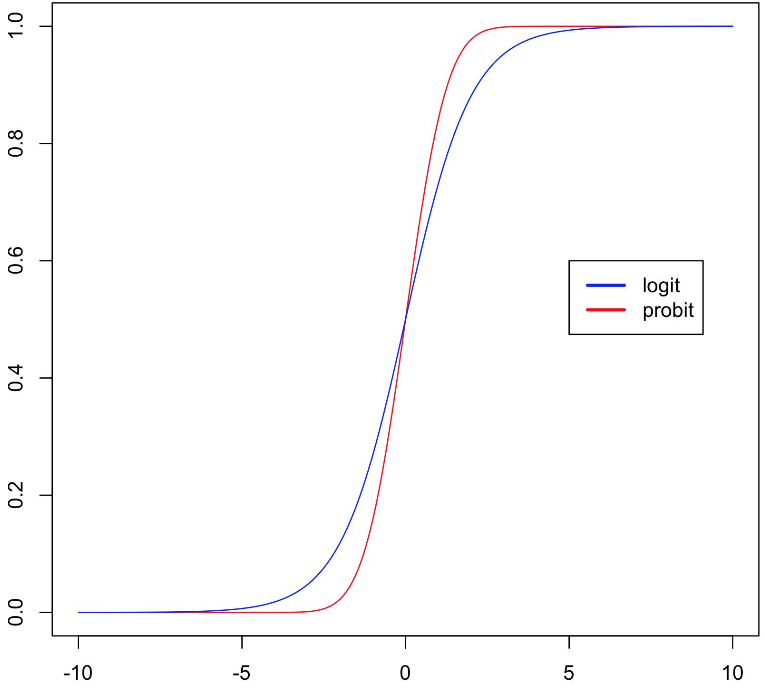
$$sigm_{\alpha}(x) = \frac{1}{1 + e^{-\alpha x}}$$

- Differentiable everywhere => we can do gradient descent
- Result is a probability-like value
- We can use it as a measure of certainty
- Parameterizable slope with $\alpha$



# Logistic Function

- First discovered by Pierre François Verhulst (1845)
- Rediscovered by Raymond Pearl and Lowell Reed (1920s)
- Used to model population growth
- Introduced in statistical analysis by Berkson (*Application of the Logistic Function to Bio-Assay*, 1944)
- Initially, an alternative to ***probit*** - CDF of normal distribution



# Logistic regression: loss function

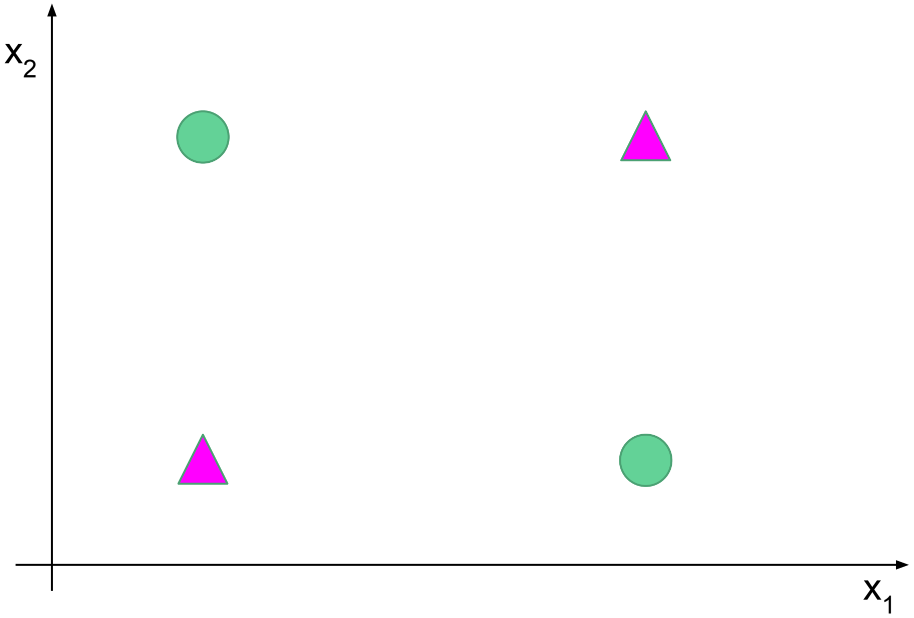
- Find **W** by minimizing the number of errors $\left\| y - \sigma(w^T x) \right\|$
- We interpret $P(y=1|x;w) = \sigma(w^T x)$
- Therefore $P(y=0|x;w) = 1 - \sigma(w^T x)$
- Equivalently $P(y|x;w) = \sigma(w^T x)^y (1 - \sigma(w^T x))^{(1-y)}$
- We define the likelihood: $L(w) = \prod_{x_i} P(y_i|x_i;w)$
- The cost function

$$J(w) = -\frac{1}{N}\sum_{i=1}^{N} y_i log(\sigma(w^T x_i)) + (1-y_i) log(1-\sigma(w^T x_i))$$

- Minimize it w.r.t. **w** (MLE)

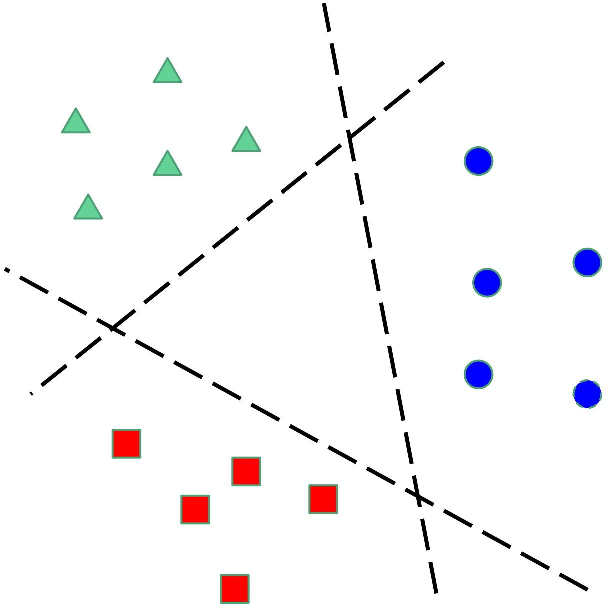
# Logistic Regression: discussion

- Bowl shaped error surface
- Gradient descent is guaranteed to find a global minimum
- Good results if data is linearly separable
- XOR problem, not linearly separable
- We need a more complex boundary



# Logistic regression: multinomial extensions

- How do we use LR for more than 2 classes?
- 1-vs-all approach:
- We can run K independent regressions
- Each will compute a different set of parameters
- Choose class with best result
- The probabilities need not sum to 1



# Logistic regression: Softmax

- A method to directly compute P(y = k | w; X), using a single model
- Softmax function generalizes the logistic function to K classes

$$P(y_i = j | x_i; w) = \frac{e^{w_j^T x_i}}{\sum_{k=1}^{K} e^{w_k^T x_i}}$$

- **X** is NxM; **w** is MxK
- The cost function becomes

$$J(w) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{K} I\{y_i = j\} log \frac{e^{w_j^T x_i}}{\sum_{l=1}^{K} e^{w_l^T x_i}}$$

# Logistic regression: Softmax

$$J(w) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K} I\{y_i = j\} log \frac{e^{w_j^T x_i}}{\sum_{l=1}^{K} e^{w_l^T x_i}}$$

- Generalizes logistic regression cost function
- No closed form solution, solved using numeric optimization (GD)
- Still convex, GD will find a global maximum
- Overparameterized: multiple **w** settings will optimize it
- Not equivalent to 1-vs-all approach
- Output probabilities necessarily sum to 1

# Softmax vs 1-vs-all

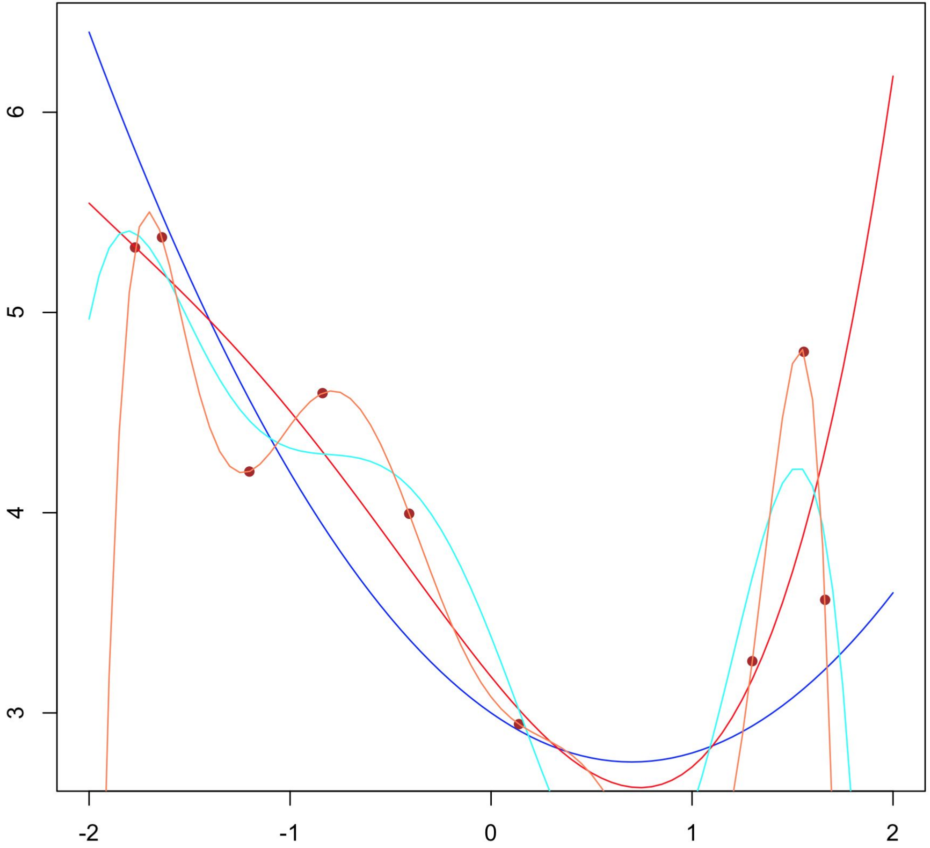
- Use softmax when we have mutually exclusive classes
  - Exclusive music genres: Pop, Rock, Jazz
- Use 1-vs-all when classes can overlap
  - A song can be Pop and Rock at the same time

# Preparing the data

- The data needs to be scaled and centered
- All columns need to be in the same range
- Height: 1.5 - 2.0
- Yearly income: 30000 - 200000
- Can use categorical data: SUNNY, CLOUDY, RAINY
- Need to be numerically encoded, introducing extra columns
- One-Hot encoding: SUNNY = {0, 0}; CLOUDY = {0, 1}; RAINY = {1, 0}
- Ordinal variables encoding is trickier: COLD, WARM, HOT
- We need to preserve the idea of ordering, but don’t know the “distances” between COLD and WARM, WARM and HOT, etc.

# Overfitting

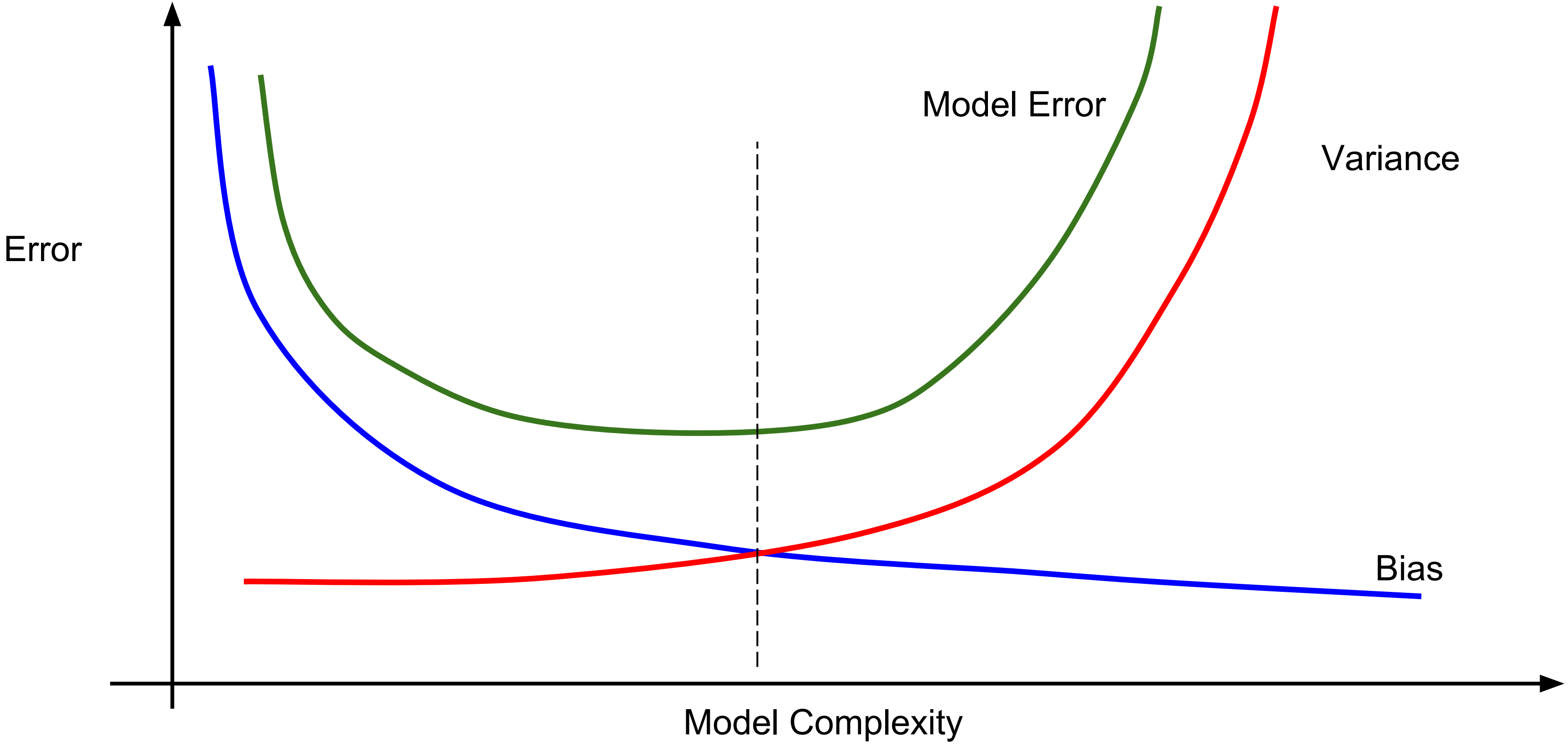
- Original $f(x) = -0.7x + 0.5x^2+3$
- Gaussian error: $\mathcal{N}(0, 0.7)$
- We fitted 4-degree polynomial
- And then 6-degree polynomial
- And then 9 degrees, which fits the data perfectly (9 points)
- However it will generalize poorly; it is ***overfitting***
- The model is too complex; it is learning noise



# Bias/Variance Tradeoff

- Suppose we train the same model M on different training sets from the same population
- Bias -- a measure of the mean error of M
- Variance -- a measure of how much the prediction of M differs from one training set to another
- High bias means underfitting; the model is too simple
- High variance means overfitting; the model is too complex
- In practice we can't eliminate both, hence the tradeoff

# Bias/Variance Tradeoff



# Bias/Variance Tradeoff

| | ***Bias*** | ***Variance*** |
|---|---|---|
| ***Low*** | - Linear regression / linear data<br>- 3rd degree poly / quadratic data<br>- ANN with many nodes trained to completion | - Constant function<br>- Linear regression / quadratic data |
| ***High*** | - Constant function<br>- Linear regression / quadratic data<br>- ANN with few nodes applied to nonlinear data | - High degree poly<br>- ANN with many nodes trained to completion |

# How to deal with overfitting

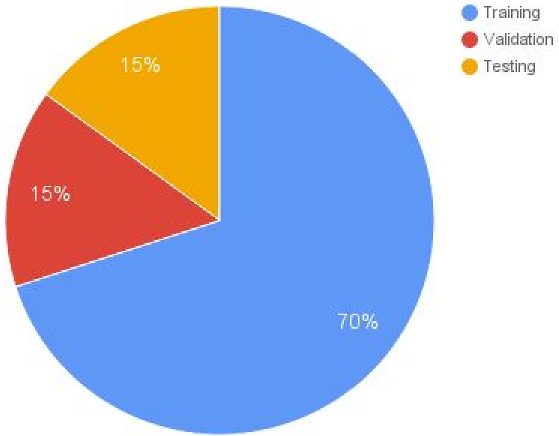
- More data; cancels the effects of noise
- Early stopping: stop before starting to overfit
- Model selection: select a model that overfits less
  - Cross-validation
- Dropout (NN): randomly drop unit contribution
  - Forces the model to learn with less input, hence less noise and co-adaptation
- Regularization: force a simpler model

# L2 Regularization

- Also known as Ridge or Tikhonov regularization
- Complex models have **w** with a higher norm
- Change the loss function to force a lower norm
- Minimize: $\mathcal{L}(w) = -logL(w) + \lambda * \|w\|^2$
- High values for **w** penalize the loss
- How do we choose λ?
- In general, how do we choose the parameters of a model?

# Parameter tuning

- In general, *training set* error is not a reliable estimation of a model's performance
- We need a *test set*, which is never seen during training; used to report a model's performance
- If we need to tune parameters (e.g. regularization $\lambda$), we set aside a *validation set*
- We train on training set, compare performance on different parameters on the validation set



# Cross-validation

- A more robust validation technique
- Instead of using a fixed training/validation split
- Perform multiple splits $(t_1, v_1), (t_2, v_2), (t_3, v_3)$, ... from the same training set
- Train using $t_i$ and validate using $v_i$
- The final error is the mean
- Multiple approaches:
  - Leave-one-out: validation set has size 1
  - K-fold: partition in K subsets; use K-1 for training and 1 for validation
    - Repeat K times
  - Repeated random split

# Demo

# Q&A