MACHINE LEARNING

MODEL SELECTION II

AGENDA

01 Cross validation

Hold out cross validation, k-fold cross validation

02 Feature Selection

Wrapper feature selection, filter feature selection

03 Bayesian statistics and Regularization

MAP estimates, Bias and variance Decomposition

04 Practical advices

Online Learning

CROSS VALIDATION

MODEL SELECTION II

CROSS VALIDATION

Model selection as posed in previous lectures, **consists** in **finding** the **best model** from a **set** of **models** $\mathcal{M} = \{M_1, \dots, M_d\}$.

A **simple approach** is **hold out cross validation** using a training set S:

1. **Randomly split S** into S_{train} (**70%** of the data) and S_{CV} (**30%** of data). Here, S_{CV} is called the **hold-out cross validation set**.

2. **Train** each **model** M_i on S_{train} only, to **get** some **hypothesis** h_i.

3. **Select** and **output** the **hypothesis** h_i that had the **smallest error** $\hat{\varepsilon}_{S_{CV}}(h_i)$ on the **hold out cross validation set**.

(Recall, $\hat{\varepsilon}_{S_{CV}}(h_i)$ denotes the **empirical error** of h on the **set** of **examples** in S_{CV}.)

MODEL SELECTION II

CROSS VALIDATION

The **disadvantage** of using **hold out cross validation** is that it "**wastes**" about **30%** of the **data**.

Thus, we could **use k-fold cross validation**:

1. **Randomly split S** into **k disjoint subsets $S1,\ldots,Sk$** of **m/k training examples** each.

2. For **each model M_i**, we **evaluate** it as **follows**:
   For $j = 1,\ldots,k$
   a) Train the **model M_i** on $S_1 \cup \cdots \cup S_{j-1} \cup S_{j+1} \cup \cdots S_k$ **(except S_j)** to get some **hypothesis h_{ij}**.
   b) Test the **hypothesis h_{ij}** on S_j, to **get** $\hat{\varepsilon}_{Sj}(h_{ij})$.
   c) The **estimated generalization error** of model M_i is then calculated as the **average** of the $\hat{\varepsilon}_{Sj}(h_{ij})$'s (averaged over j).

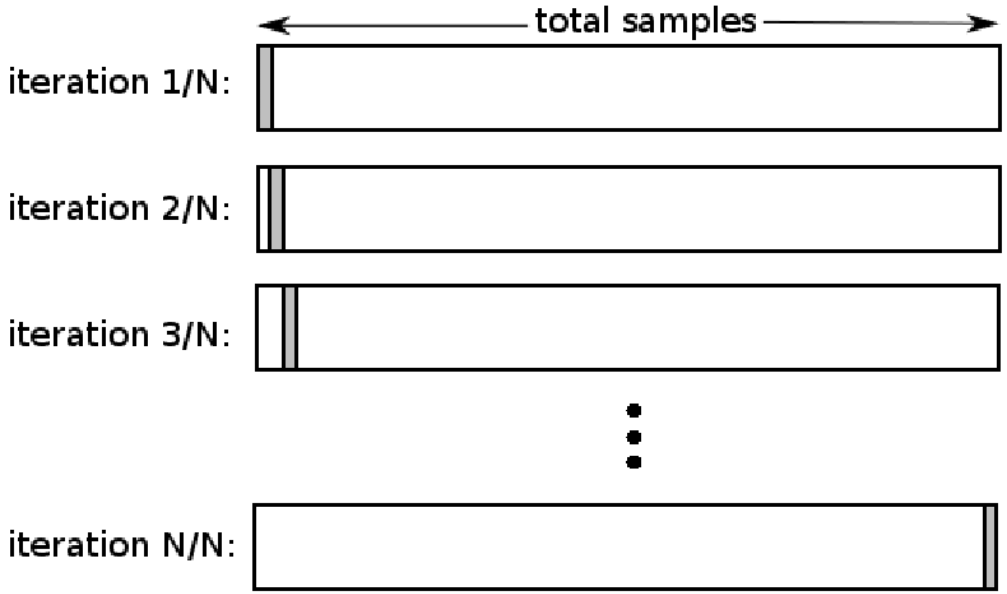
3. Pick the model M_i with the **lowest estimated generalization error** and **retrain** that **model** on the **entire training set S**. The resulting hypothesis is then output as our final answer.

MODEL SELECTION II

CROSS VALIDATION

The **typical** of **choice** for k in k-fold cross validation will be $k = 10$**. Even though,** there would be times when we will need to **train** each **model** $k = m$ **times**, a method called **leave-one-out cross validation**.

Thus, this **method** may become **very computationally expensive**.



FEATURE SELECTION

MODEL SELECTION II

FEATURE SELECTION

Suppose that we have a **supervised learning problem** where the **number** of **features** $\boldsymbol{n}$ is **very large**, but we **suspect** that there is only a **small number** of **features** that are "**relevant**".

Thus, the **VC dimension** of the **hypothesis class** would still be **O(n),** and thus **overfitting** would be a **potential problem unless** the **training set** is **large**.

Given $\boldsymbol{n}$ **features**, there are $\boldsymbol{2^n}$ **possible feature subsets**(since each of the $\boldsymbol{n}$ features can either be included or excluded from the subset).

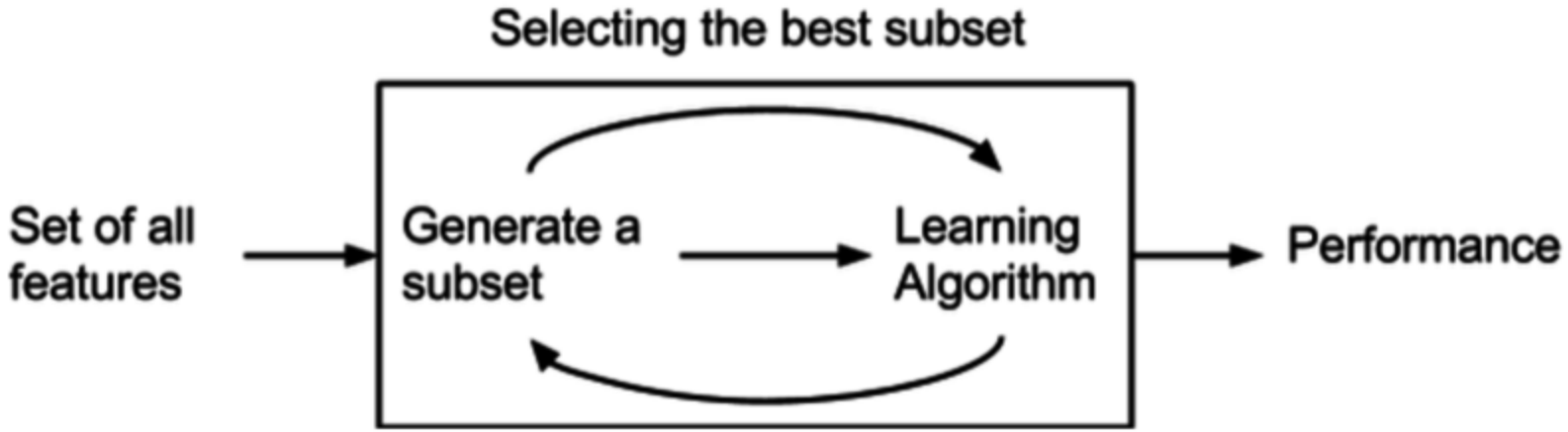
Therefore, **feature selection** can be posed as a **model selection problem** over $\boldsymbol{2^n}$ **possible models.** Because there are too **many possible models**, **heuristic** approaches are **designed** to **tackle** the **problem**.

MODEL SELECTION II

FEATURE SELECTION

WRAPPER FEATURE SELECTION:

A **class** of **feature selection algorithms** that "**wraps**" **around** your **learning algorithm**, and repeatedly **makes calls** to the learning algorithm to **evaluate** how well it does **using different feature subsets**.



MODEL SELECTION II

FEATURE SELECTION

WRAPPER FEATURE SELECTION:
Forward search

1. **Initialize** $F = \emptyset$.

2. **Repeat** {
   a) For $i = 1, \ldots, n$ if $i \neg\in F$, let $F_i = F \cup \{i\}$, and **use some version** of **cross validation** to **evaluate features** F_i. (I.e., train your learning algorithm using only the features in F_i, and estimate its generalization error.)
   b) Set F to be the **best feature subset** found on **step (a)**.
   }

3. **Select** and **output** the **best feature subset** that was **evaluated** during the **entire search procedure**.

You **end** when you have **added all features** to F or by **specifying** some **threshold k**.

MODEL SELECTION II

FEATURE SELECTION

WRAPPER FEATURE SELECTION:
Backward search

Starts off **with** $\boldsymbol{F = \{1, \ldots, n\}}$ as the set of **all features**, and **repeatedly deletes features one** at a **time** until $\boldsymbol{F = \emptyset}$**.**

The **problem** with **wrapper methods** reside in its **computational cost.** It so that **running** a **complete forward search algorithm** would take $\boldsymbol{O(n^2)}$ **calls.**

Even though, **wrapper models tend** to **work better** than other feature selection methods.

MODEL SELECTION II

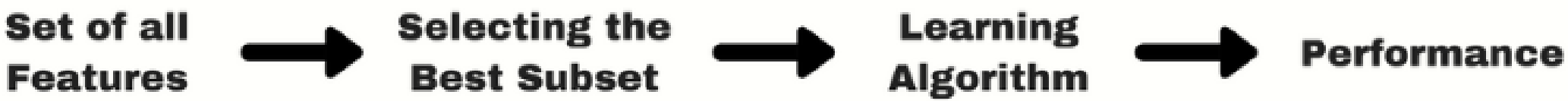
FEATURE SELECTION

FILTER FEATURE SELECTION:

Give a **heuristic**, but they are **computationally cheaper** than **wrapper methods**.

The **objective** resides in **computing** a **simple score** $S(j)$ that **measures** how **informative** each **feature** x_j is **about** the **class labels** y.

Finally, we **choose** the k **features** with **largest scores** $S(j)$**.**



MODEL SELECTION II

FEATURE SELECTION

FILTER FEATURE SELECTION:

For example, we could **compute** the **correlation** between x_j and y. Thus, you will **choose** the **features** that are the **most strongly correlated** with **class labels**.

It is **more common** to choose $S(j)$ (**particularly** for **discrete-valued features**) as the **mutual information** $MI(x_j, y)$ between x_j and y.

$$MI(x_j, y) = \sum_{x_j \in \{0,1\}} \sum_{y \in \{0,1\}} p(x_j, y) \frac{p(x_j, y)}{p(x_j)p(y)}$$

Probabilities, $p(x_j, y)$, $p(x_j)$ and $p(y)$ **can** all be **estimated** according to their **empirical distributions** on their **training set**.

MODEL SELECTION II

FEATURE SELECTION

FILTER FEATURE SELECTION:

The **mutual information** score can **also** be **represented** as the **Kullback-Leibler** (**KL**) **divergence**:

$$\boldsymbol{MI(x_j, y) = KL\left(p(x_j, y)||p(x_j)p(y)\right)}$$

This gives a **measure** of how **different** the **two probability distributions** are.

If **both**, $\boldsymbol{p(x_j)p(y)}$ and $\boldsymbol{p(x_j)p(y)}$, are **independent** we have that $\boldsymbol{p(x_j, y) = p(x_j)\,p(y)}$. Thus, the **KL divergence** will be **zero**.

In other words, if $\boldsymbol{x_j}$ and $\boldsymbol{y}$ **are independent**, then $\boldsymbol{x_j}$ is **very** "**non-informative**" about $\boldsymbol{y}$, and thus **score** $\boldsymbol{S(j)}$ should be **small.**

MODEL SELECTION II

FEATURE SELECTION

FILTER FEATURE SELECTION:

Once you have **filtered** all k **features** based on **score** $S(j)$. You **select** k using **cross validation** by **iterating** over each **combination** of **best features**..

Other **scores** $S(j)$ that are **used:**

- **Pearson's Correlation.**
- **LDA.**
- **ANOVA.**
- **Chi-Square.**

BAYESIAN STATISTICS REGULARIZATION

MODEL SELECTION II

BAYESIAN STATISTICS

At the beginning of the past course we talked about **parameter fitting** using **Maximum Likelihood Estimation (MLE)**:

$$w_{MLE} = \underset{w}{\text{argmax}} \prod_{i=1}^{m} p(y^{(i)} / x^{(i)}; w)$$

FREQUENTIST: w is **NOT random**, but an **unknown constant-valued parameter**.

BAYESIAN: w **IS random** and an **unknown parameter**.

MODEL SELECTION II

BAYESIAN STATISTICS

In the **Bayesian approach**, we will **specify** a **prior distribution** $\boldsymbol{p(w)}$ (may be a normal distribution $\boldsymbol{N(0, \tau^2 I)}$) that captures our **"prior beliefs"** about the **parameters without** any **data**.

Thus, **given** a **training set** $\boldsymbol{S = \{(x^{(i)}, y^{(i)})\}_{i=1}^{m}}$ when we are asked to **make** a **prediction** on a **new value** of $\boldsymbol{x}$, we can then **compute** the **posterior distribution** on the **parameters**.

$$\boldsymbol{p(w/S) = \frac{p(S/w)p(w)}{p(S)}}$$

$$\boldsymbol{p(w/S) = \frac{\left(\prod_{i=1}^{m} p(y^{(i)}/x^{(i)}, w)\right)p(w)}{\int_w \left(\prod_{i=1}^{m} p(y^{(i)}/x^{(i)}, w)\, p(w)\right)dw}}$$

Posterior ⇩ $P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$ Likelihood ⇩ Prior ⇩ Evidence ⇧

EXAMPLE:

For **Bayesian logistic regression** you will have: $\boldsymbol{p(y^{(i)}/x^{(i)}; w) = h_w(x^{(i)})^{y^{(i)}} (1 - h_w(x^{(i)}))^{1-y^{(i)}}}$

MODEL SELECTION II

BAYESIAN STATISTICS

By **knowing** the **posterior distribution** $p(w/S)$ of the **parameters** w, we can **compute** the **posterior distribution** $p(y/x,S)$ of our **class labels** y after seeing the data:

$$p(y/x,S) = \int_w (p(y/x,w)p(w/S))\,dw$$

Therefore, the **expected value** of y/x will be:

$$E[y/x,S] = \int_y y\,p(y/x,S) = dy$$

MODEL SELECTION II

BAYESIAN STATISTICS

The **problem** with **Bayesian approach** is that it is **very difficult** to **compute** the **posterior distribution** because it requires **taking high dimensional integrals** over $\boldsymbol{w}$.

Thus, we will instead **approximate** the **posterior distribution** for **w.** To do so, we are going to **replace** the **posterior distribution** of $\boldsymbol{w}$ with a **single point estimate**.

The **MAP** (**Maximum a Posteriori**) **estimate** for $\boldsymbol{w}$ is given by:

$$\boldsymbol{p(w/S) = \frac{p(S/w)p(w)}{p(S)} = \frac{\left(\prod_{i=1}^{m} p(y^{(i)}/x^{(i)}, w)\right)p(w)}{\int_w \left(\prod_{i=1}^{m} p(y^{(i)}/x^{(i)}, w)\, p(w)\right)dw}}$$

estimate

$$\boldsymbol{\hat{w}_{MAP} = \underset{w}{\mathbf{argmax}} \prod_{i=1}^{m} p\left(y^{(i)}/x^{(i)}, w\right) p(w)}$$

MODEL SELECTION II

BAYESIAN STATISTICS

To make a **new prediction** with the **MAP estimate**, we will have:

$$h_{\hat{w}_{MAP}}\left(\hat{w}_{MAP}{}^{T}X\right)$$

The **difference between MAP** and **MLE** is that we **maximize considering** the **prior** $p(w)$.

$$\hat{w}_{MAP} = \underset{w}{\operatorname{argmax}} \prod_{i=1}^{m} p\left(y^{(i)}/x^{(i)}, w\right) p(w)$$

$$w_{MLE} = \underset{w}{\operatorname{argmax}} \prod_{i=1}^{m} p(y^{(i)}/x^{(i)}; w)$$

WHY BAYESIAN REGRESSION PREVENTS OVERFITTING?

MODEL SELECTION II

BAYESIAN STATISTICS

If we **choose** a **prior** $p(w)$ to be a **normal distribution** $N(0, \tau^2 I)$ we are **"believing"** that **most** of the **weights** are **centered** around **zero.**

Thus**,** you are saying that **many features** may **not** be **considered** → **REDUCE complexity.**

In conclusion**, Bayesian regression** may be a **very effective algorithm** when $n \gg m$**.**

MODEL SELECTION II

BIAS AND VARIANCE DECOMPOSITION

Given a **training set** $\boldsymbol{S} = \{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})\}_{\boldsymbol{i=1}}^{\boldsymbol{m}}$ we **make** the **assumption** that $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{\epsilon}$, where $\boldsymbol{E}[\boldsymbol{\epsilon}] = \boldsymbol{0}$ and $\boldsymbol{V}[\boldsymbol{\epsilon}] = \boldsymbol{\tau^2}$ (**not necessarily Gaussian**). We define the **true** $\boldsymbol{f}$ as

$$\boldsymbol{f}(\boldsymbol{x}') = \boldsymbol{E}[\boldsymbol{y}/\boldsymbol{x} = \boldsymbol{x}']$$

We want to **construct** a **hypothesis** $\hat{\boldsymbol{f}}_{\boldsymbol{m}}$ given a **fixed size** training set $\boldsymbol{S}$ that mimics $\boldsymbol{f}$ well on all future **unseen examples**. In other words, $\hat{\boldsymbol{f}}_{\boldsymbol{m}}$ needs to have **good generalization error**.

We will only consider the case where the **generalization error** is the **EXPECTED SQUARED ERROR LOSS** on an **unseen example**.

MODEL SELECTION II

BIAS AND VARIANCE DECOMPOSITION

Suppose that the **estimated function** $\hat{f}_m$ is **obtained** with some **training process** over S. We can see that the **estimated function** $\hat{f}_m$ is **random**, where the **randomness comes** from the **errors** $\epsilon^{(i)\prime}s$ **embedded** in the **training** set **examples**.

Consider a **new unseen** example **pair** (y_*, x_*) and the **corresponding generalization error**, where the **expectation** is **over** the **randomness** in ϵ embedded in the **test example**, and in $\hat{f}_m$.

$$MSE(\hat{f}_m) = E\left[\left(y_* - \hat{f}_m(x_*)\right)^2\right]$$

$$MSE(\hat{f}_m) = E\left[\left(\epsilon + f(x_*) - \hat{f}_m(x_*)\right)^2\right]$$

MODEL SELECTION II

BIAS AND VARIANCE DECOMPOSITION

We can now **develop** the **equation**:

$$MSE(\hat{f}_m) = E\left[\left(\epsilon + f(x_*) - \hat{f}_m(x_*)\right)^2\right]$$

$$MSE(\hat{f}_m) = E\left[\epsilon^2 + \left(f(x_*) - \hat{f}_m(x_*)\right)^2 + 2\epsilon\left(f(x_*) - \hat{f}_m(x_*)\right)\right]$$

↓ Property of linearity $E(A+B) = E(A) + E(B)$

$$MSE(\hat{f}_m) = E[\epsilon^2] + E\left[\left(f(x_*) - \hat{f}_m(x_*)\right)^2\right] + E\left[2\epsilon\left(f(x_*) - \hat{f}_m(x_*)\right)\right]$$

↓ Because the errors ϵ are i. i. d. $E(AB) = E(A)E(B)$

$$MSE(\hat{f}_m) = E[\epsilon^2] + E\left[\left(f(x_*) - \hat{f}_m(x_*)\right)^2\right] + E[\epsilon]E\left[2\left(f(x_*) - \hat{f}_m(x_*)\right)\right]$$

↓ Assumption $E(\epsilon) = 0$

$$MSE(\hat{f}_m) = E[\epsilon^2] + E\left[\left(f(x_*) - \hat{f}_m(x_*)\right)^2\right]$$

MODEL SELECTION II

BIAS AND VARIANCE DECOMPOSITION

We can now **develop** the **equation**:

$$MSE(\hat{f}_m) = E[\epsilon^2] + E\left[\left(f(x_*) - \hat{f}_m(x_*)\right)^2\right]$$

Property $E(A^2) = V(A) + E(A)^2$

$$MSE(\hat{f}_m) = E[\epsilon^2] + V[f(x_*) - \hat{f}_m(x_*)] + E[f(x_*) - \hat{f}_m(x_*)]^2$$

Property $\tau^2 = V(\varepsilon) = E[(\varepsilon - E[\varepsilon])^2] = E[(\varepsilon - 0)^2] = E[(\varepsilon)^2]$

$$MSE(\hat{f}_m) = \tau^2 + V[f(x_*) - \hat{f}_m(x_*)] + E[f(x_*) - \hat{f}_m(x_*)]^2$$

Property $V[a - X] = V[X]$

$$MSE(\hat{f}_m) = \tau^2 + V[\hat{f}_m(x_*)] + E[f(x_*) - \hat{f}_m(x_*)]^2$$

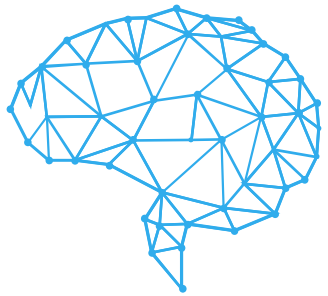
$$MSE(\hat{f}_m) = \underbrace{\tau^2}_{\text{IRREDUCIBLE ERROR}} + \underbrace{V[\hat{f}_m(x_*)]}_{\text{VARIANCE}} + \underbrace{E[\hat{f}_m(x_*) - f(x_*)]^2}_{\text{BIAS}^2}$$

NOTE: a clean decomposition into Bias and Variance terms exists only for the squared error loss.

PRACTICAL ADVICES

MODEL SELECTION II

PRACTICAL ADVICE



ONLINE LEARNING:

All **training algorithms** that we have **seen so far** are called **batch learning algorithms**, in which we **train** and **validate** over **fixed sets** of data.

In **online learning** we have **new data available** through time, so the **algorithm will learn repeatedly** from these **new sources** of data.

Therefore, **given** a **training algorithm**:

1. **Initialize parameters** of the algorithm
2. After receiving each i^{th} training example, **update parameters** using any optimization strategy (**stochastic gradient descent**) **using only** that **single example**.

The **main problem** with this approach is **CATASTROPHIC INTERFERENCE**

MODEL SELECTION II

PRACTICAL ADVICE

DEBUGGING LEARNING ALGORITHMS:

You have trained an algorithm, but the **performance** was **poor**. Some common **approaches** to **solve** this **include**:

- Try getting **more** training **examples**.
- Try a **smaller** set of **features**.
- Try a **larger** set of **features**.
- Try **changing** the **features**.
- Run **gradient descent** for **more iterations**.
- Try **Newton**' **method**.
- Use **different regularization** techniques.
- Try a **different algorithm**.

The **problem** with this is that it is **very time-consuming** and **may not** even **work**. You **need** to **analyze deeply** the **problem** to **know** what **approach** you **need** to **take**.

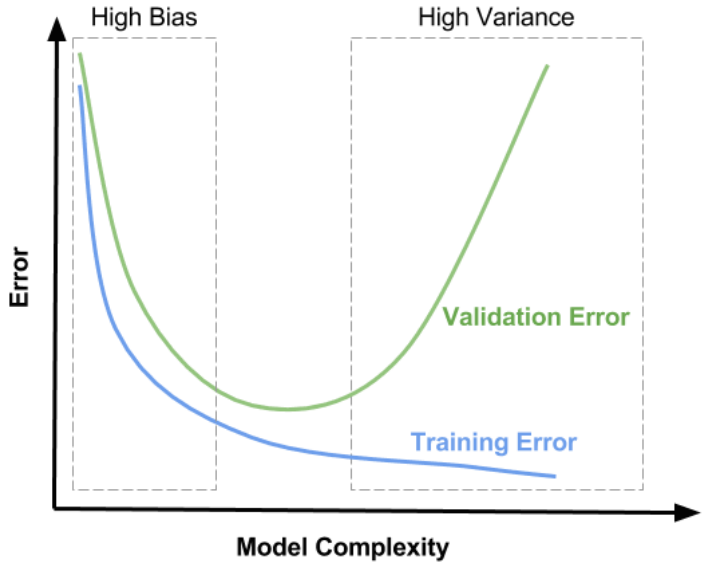
MODEL SELECTION II

PRACTICAL ADVICE

DEBUGGING LEARNING ALGORITHMS:

You will need to **check** if the **problem** is **related** with **high bias** (**underfitting**) or **high variance** (**overfitting**).

- **High variance**: **training error** will be much **lower than test error**.
- **High bias**: **training error** will **also** be **high**.



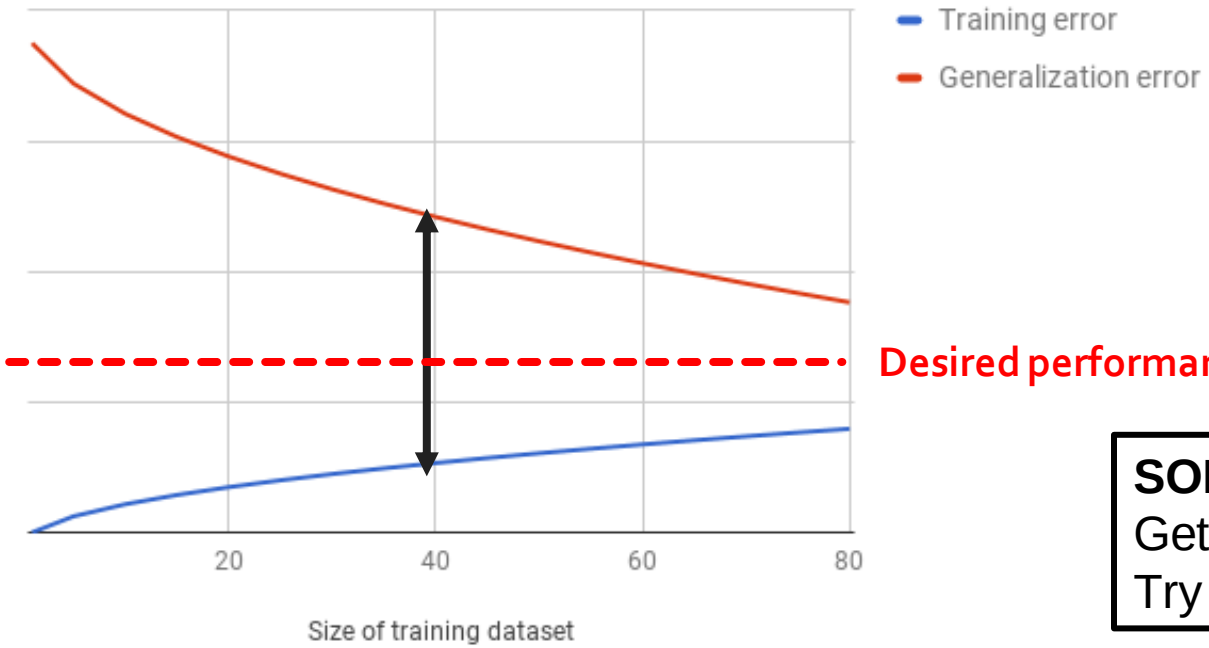
MODEL SELECTION II

PRACTICAL ADVICE

DEBUGGING LEARNING ALGORITHMS:

Typical **learning curve** for **high variance**:

- **Test error** still **decreases** as m **increases**. Thus a **larger training set** will **help**.
- **Training error increases** as m **increases**
- **High variance**: there is a **large gap** between the **training** and **test errors.**



SOLUTION:
Get more training examples.
Try a smaller set of features.

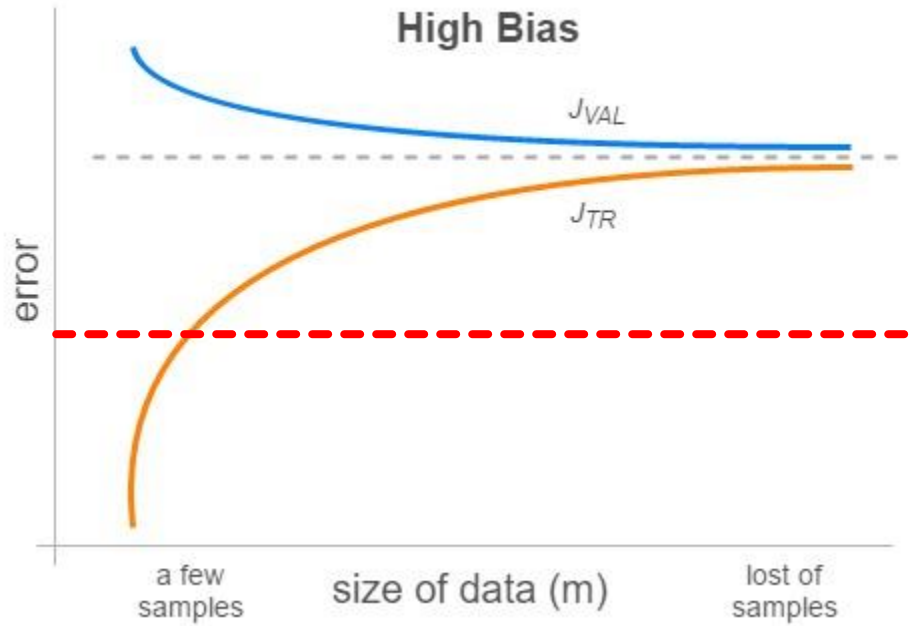
MODEL SELECTION II

PRACTICAL ADVICE

DEBUGGING LEARNING ALGORITHMS:

Typical **learning curve** for **high bias**:

- **Test error flattens** very **early** on (**getting more** training **data** is **not** the **solution**).
- **Training error is unacceptably high** as m **increases**
- **High bias**: there is a **small gap** between the **training** and **test errors.**



Desired performance

SOLUTION:
Try a larger set of features.
Try changing the features.

MODEL SELECTION II

PRACTICAL ADVICE

DEBUGGING LEARNING ALGORITHMS:

We need to deal with **two** other **problems**:

- **Optimization algorithm** (gradient descent).
- **Objective function**(cost function $\boldsymbol{J}$).

Suppose we try **two algorithms**: **SVM** and **logistic regression**. We will define the **parameters** obtained by **both algorithms** as $\boldsymbol{w_{SVM}}$ and $\boldsymbol{w_{LR}}$ respectively.

Therefore you will have **two measures** for **both algorithms**:

- $\boldsymbol{J(w)}$**:** cost function of LR or SVM.
- $\boldsymbol{F1(w)}$**:** F1 score of LR or SVM (target metric).

MODEL SELECTION II

PRACTICAL ADVICE

DEBUGGING LEARNING ALGORITHMS:

You have **two cases** supposing that the **optimization objective** was specified as a **maximization** problem (**NOT minimization**). :

Case 1:

- $F1(w_{SVM}) > F1(w_{LR})$
- $J(w_{SVM}) > J(w_{LR})$
- w_{LR} **failed** to **maximize** J, thus the **problem** is in the **optimization algorithm** (GD).
- **Solution**: more iterations of GD or change to Newton's method.

Case 2:

- $F1(w_{SVM}) > F1(w_{LR})$
- $J(w_{SVM}) \leq J(w_{LR})$
- w_{LR} **succeded** to **maximize** J, thus the **problem** is in the **objective function** (J).
- **Solution**: try different λ, change the algorithm to SVM.

MODEL SELECTION II

PRACTICAL ADVICE

DEBUGGING LEARNING ALGORITHMS:

Even though, there will be times when **you need** to **come up** with your **own diagnostics**.

Also, **even if** an **algorithm** is **working well**, you might **run diagnostics** to **make sure** you **understand** what is **going on**.

- If you **spend** a **lot** of **time** in the **problem**, it is very **valuable** to **get** an **intuitive sense** of **what works** and **what doesn't work** in your problem.

- **Justify why** the **algorithm** is **working**.

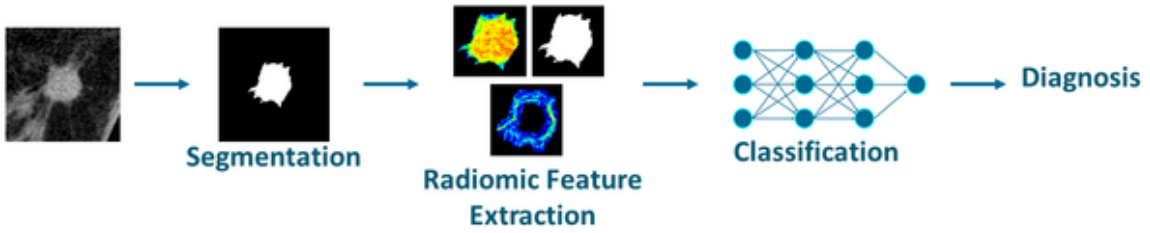
MODEL SELECTION II

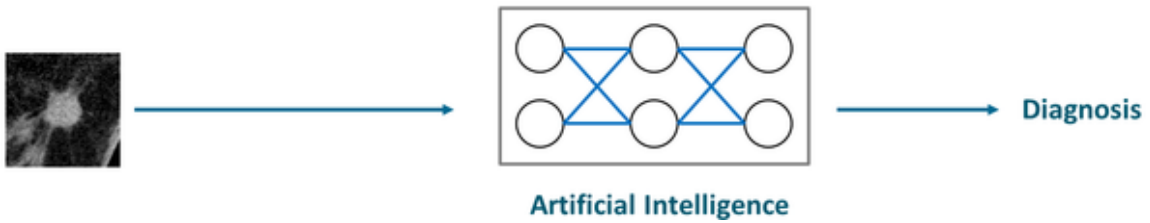
PRACTICAL ADVICE

ERROR ANALYSIS:

Many **applications** combine **different components** into a “**pipeline**”.

Explain **difference** between **current** and **perfect performance**.





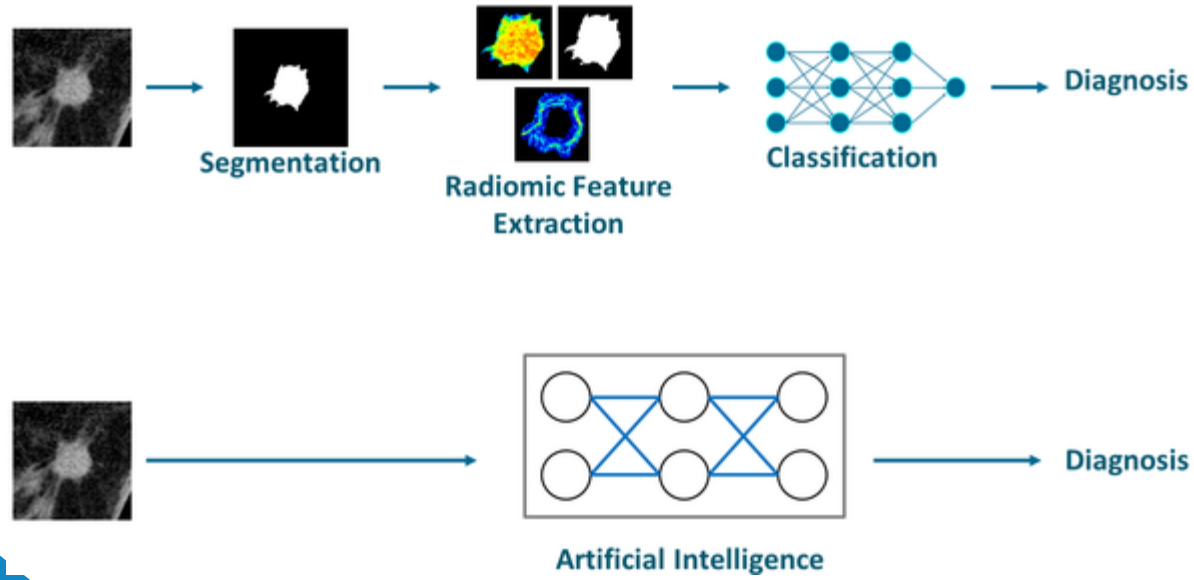
How many error is attributable to each of the components?

MODEL SELECTION II

PRACTICAL ADVICE

ERROR ANALYSIS:

Plug in the **ground-truth** for each **component** and **see** how **accuracy** (F1 score) **changes**.



COMPONENT	ACCURACY
Overall system	85 %
Segmentation	93.1 %
Radiomic Feature Extraction	95 %
Algorithm	100 %

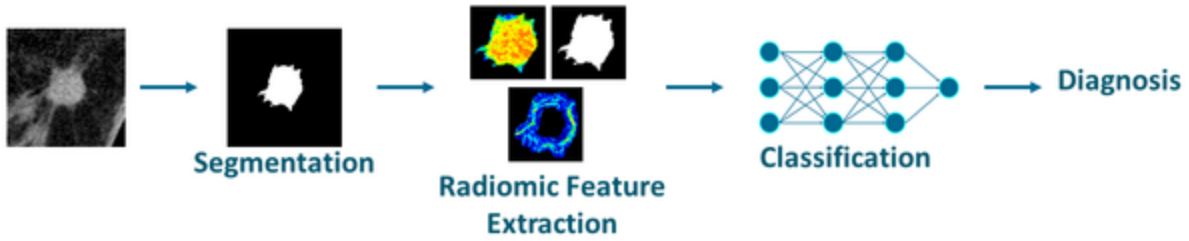
FOCUS ON SEGMENTATION

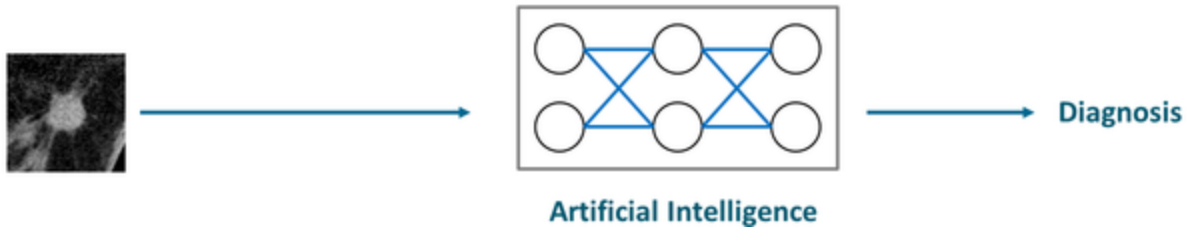
MODEL SELECTION II

PRACTICAL ADVICE

ABLATIVE ANALYSIS:

Explain **difference** between **baseline (poorer)** and **current performance**.





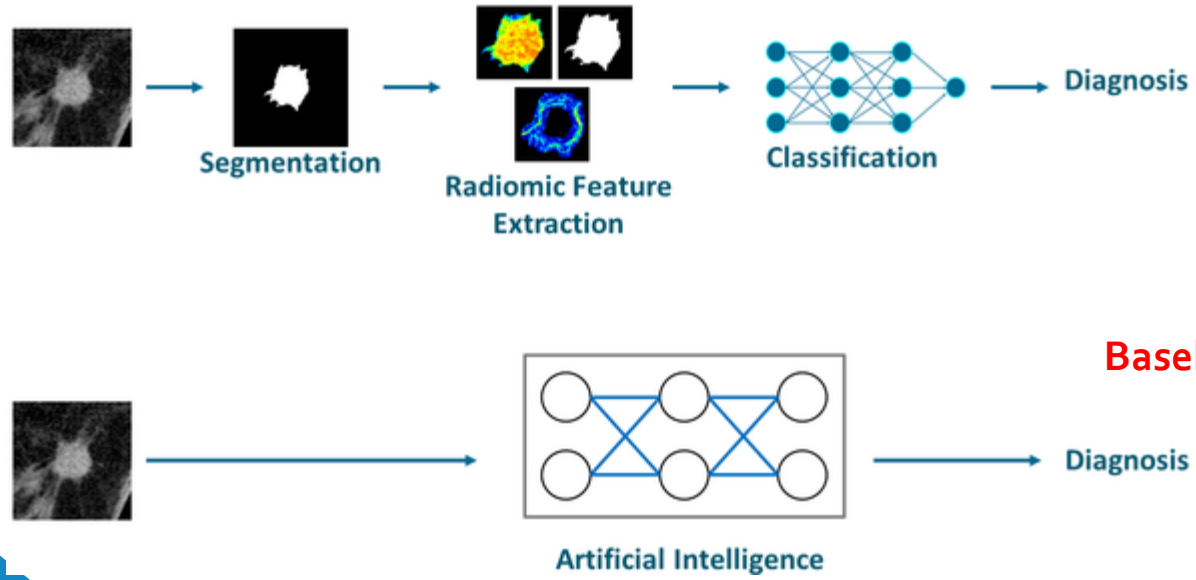
How much did each of these components really help?

MODEL SELECTION II

PRACTICAL ADVICE

ABLATIVE ANALYSIS:

Remove components from your system **one at** a **time**. **See** how it **breaks**.



COMPONENT	ACCURACY
Overall system	85 %
Segmentation	84 %
Radiomic Feature Extraction	80%

RADIOMIC FEATURE EXTRACTION CONTRIBUTED THE MOST

MODEL SELECTION II

PRACTICAL ADVICE

GETTING STARTED:

Approach 1: careful design

- Spend **long time** designing right features, collecting data and designing the algorithm.
- **Benefit**: more scalable algorithms, contribute to research, new algorithms.

Approach 2: build and fix.

- Build something **quick** and **dirty**.
- Run error analysis and diagnostics, then fix it.
- **Benefit**: application will may work more quickly. Faster time to market.

MODEL SELECTION II

PRACTICAL ADVICE

GETTING STARTED:

- The **first thing** you need to do is **PLOT** the **data**, **UNDERSTAND** it! What is **wrong with it**?
- You **don’t want** to **start** with **building very complex pipelines**. Start simple!
- If you **want** to **do research**, you will need to **do very deep analysis**. Just **don’t** go **too deep** expecting that it **will link naturally** to an **application**!
- Spend a **LOT** of time in **diagnostics**!