CS 4342: Class 2

Jacob Whitehill

How old are these people?

Guess how old each person is based on their face image.

66

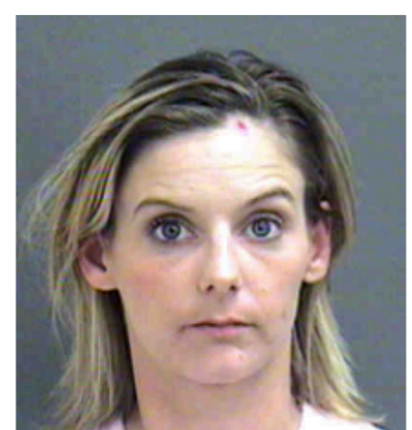
18

38

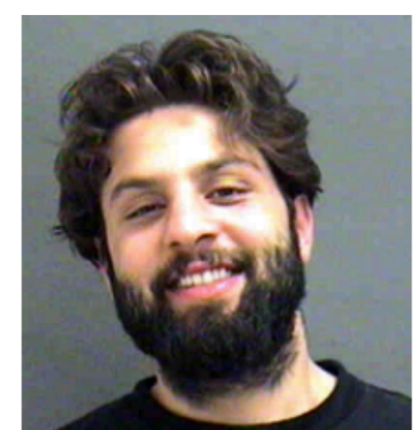
61

57

53



29



23

https://www.vision.ee.ethz.ch/en/publications/papers/articles/eth_biwi_01299.pdf

Ensemble of estimators

- What if we compute the **average** of the different predictions?

$$\overline{\mathbf{y}} = \frac{1}{m} \sum_{j=1}^{m} \hat{\mathbf{y}}^{(j)}$$

- What is the MSE of the **average predictor**?

$$f_{\mathrm{MSE}}(\overline{\mathbf{y}})$$

Ensemble of estimators

- What if we compute the **average** of the different predictions?

$$\overline{\mathbf{y}} = \frac{1}{m}\sum_{j=1}^{m}\hat{\mathbf{y}}^{(j)}$$

- What is the MSE of the **average predictor**?

$$f_{\mathrm{MSE}}(\overline{\mathbf{y}})$$

- How does this compare with the **average MSE** of all the predictors?

$$\frac{1}{m}\sum_{j=1}^{m} f_{\mathrm{MSE}}(\hat{\mathbf{y}}^{(j)})$$

Age estimation accuracy

- Show **`wisdom.py`**

Age estimation accuracy

- Show **`wisdom.py`**
- The MSE of the average predictor tends to be lower (better) than the average MSE over all predictors.
- This is an instance of the "wisdom of the crowd".
- Averaging together multiple predictor is sometimes called an **ensemble**.

Who is smiling?

Which of these people are smiling?







https://www.vision.ee.ethz.ch/en/publications/papers/articles/eth_biwi_01299.pdf

Who is smiling?

Which of these people are smiling?

1

1

0

?

0

0

0

1

https://www.vision.ee.ethz.ch/en/publications/papers/articles/eth_biwi_01299.pdf

Defining ground-truth

- A fundamental question in every machine learning problem is how to define what ground-truth means.

- In our example, we might define it as:

    - Does the person *look* like they're smiling?

    - Does the person *her/himself* report that they're smiling?

    - Is the person's lip-corner-puller *muscle* activated?

Quantifying uncertainty

- Sometimes the ground-truth value is unclear.
- To express a “soft” belief about the ground-truth, we can use **probabilities**.
- There are a couple of ways we could do this…

Quantifying uncertainty

- **Frequentist** probabilities:
  - Ask a large group of randomly selected people to **label** the face as smiling or not.
  - Count the number of labels for “smile” and divide by the total number of labels.
  - The ratio is the probability of “smile" for that face image.

Quantifying uncertainty

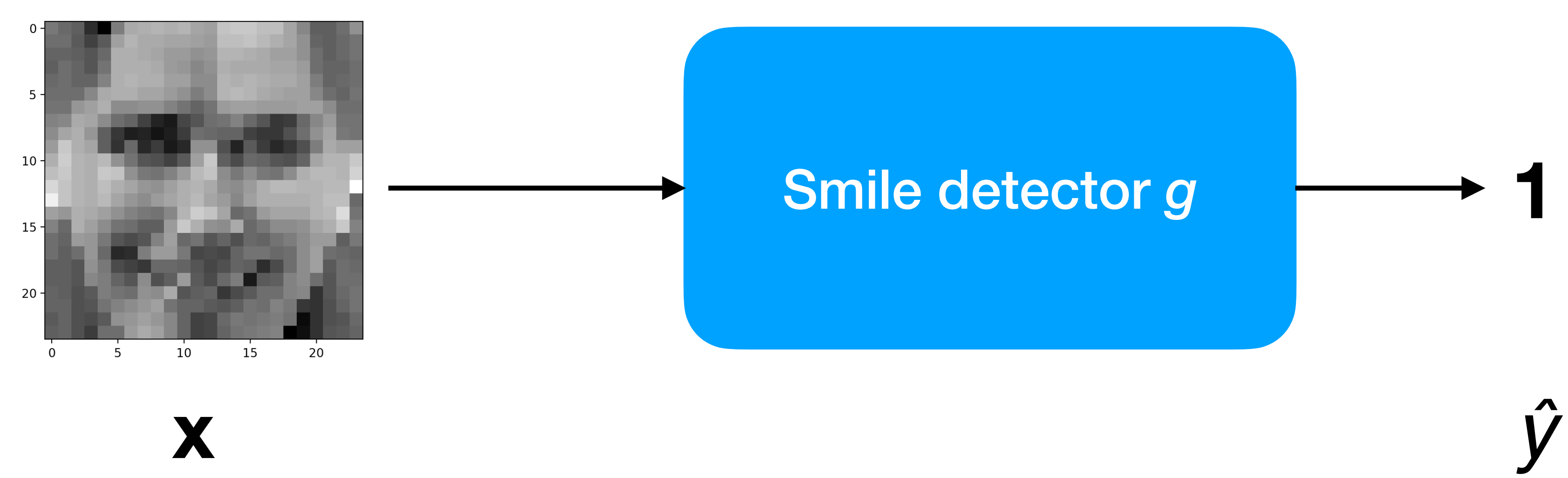
- **Bayesian** probabilities ("beliefs"):
  - Ask one person how much she/he believes the image is smiling, quantified as a number between 0 and 1.
  - The "belief" score is the probability of "smile" for that face image.

Automatic smile detection

- Suppose we want to build an automatic smile detector that analyzes a grayscale face image (24x24 pixels) and reports whether the face is smiling.

- We can represent the detector as a function *g* that takes an image **x** as an input and produces a guess $\hat{y}$ as output, where $\mathbf{x} \in \mathbb{R}^{24 \times 24}, \hat{y} \in \{0, 1\}$.

- Abstractly, *g* can be considered a "machine":

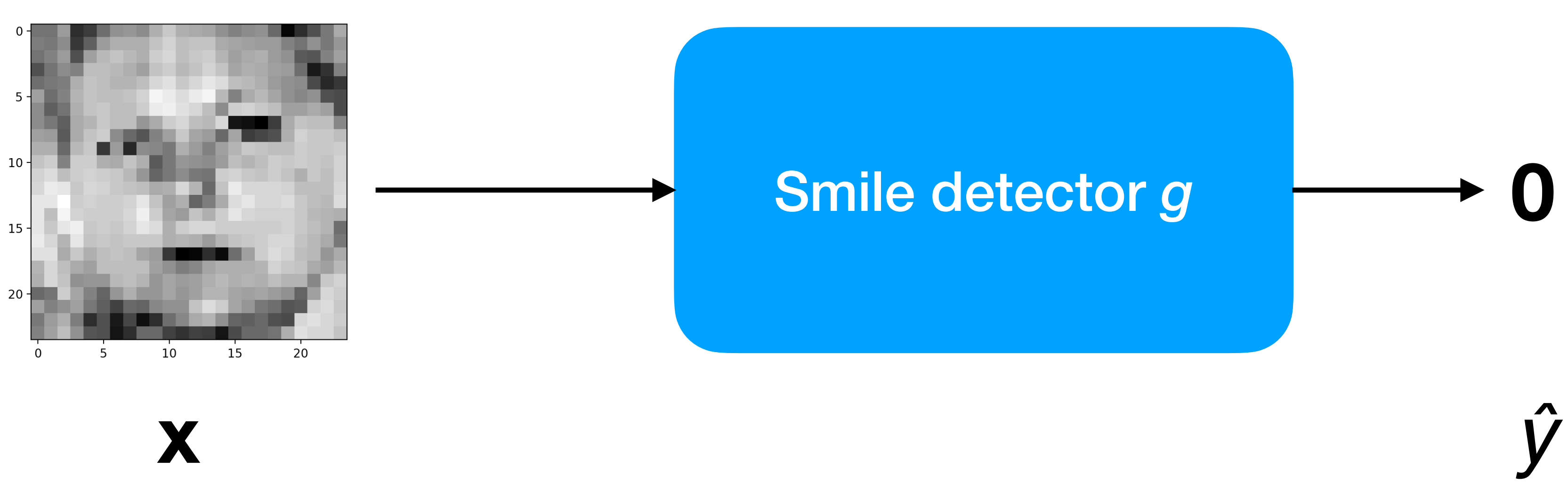


Automatic smile detection

- Suppose we want to build an automatic smile detector that analyzes a grayscale face image (24x24 pixels) and reports whether the face is smiling.

- We can represent the detector as a function *g* that takes an image **x** as an input and produces a guess $\hat{y}$ as output, where $\mathbf{x} \in \mathbb{R}^{24 \times 24}, \hat{y} \in \{0, 1\}$.

- Abstractly, *g* can be considered a “machine”:

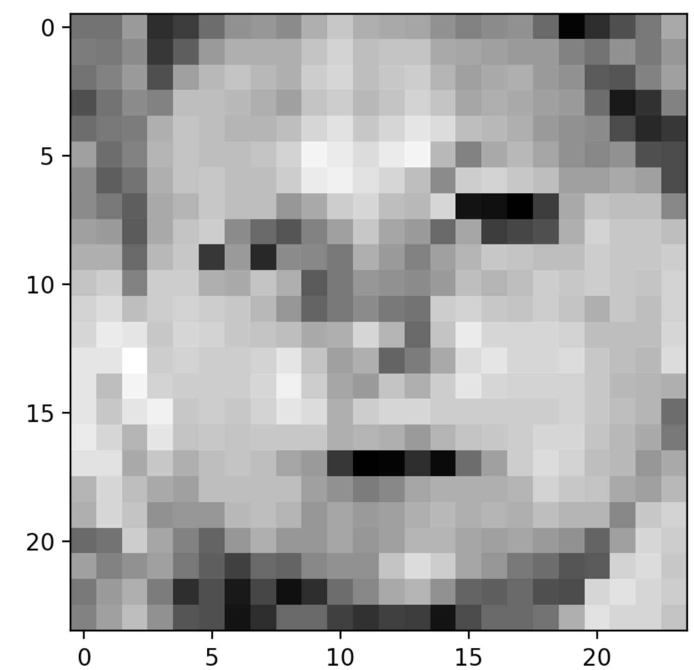


Smile classifier

- In Python, we can represent a face image as a 24x24 numpy array called `face`.
    - We can access the pixel at location (r,c) as `face[r,c]`.
- Suppose we have a dataset of several thousand face images { $\mathbf{x}^{(i)}$ } along with their associated labels { $y^{(i)}$ }.

Smile classifier

- How might we write a Python function called `classifySmile` that takes a face and returns whether the face is smiling (`True`) or not (`False`)?



- Example:

```
def classifySmile (face):
    return ... (some function of face)
```

- What accuracy (f_{PC}) does our function achieve?

Accuracy measurement

- Let’s try this by hand in smile.py
- What accuracy can we achieve?
- Is this “good”?

Selecting a baseline

- In addition to defining an accuracy function, it's important to choose a "baseline" to which to compare your machine.
- The baseline is often the "leading brand" — the best machine that anyone has ever created before for the same problem.
- For a *new* ML problem, we might just compare to (1)
  ? or (2)
  ?
  .

Selecting a baseline

- In addition to defining an accuracy function, it’s important to choose a “baseline” to which to compare your machine.

- The baseline is often the “leading brand” — the best machine that anyone has ever created before for the same problem.

- For a *new* ML problem, we might just compare to (1) random guessing or (2) selecting the most probable class based on the statistics of the dataset.

Selecting a baseline

- What fraction of faces in $\mathcal{D}^{\text{test}}$ are smiling faces? 54.6%
- How accurate (f_{PC}) would a predictor be that just always output 1 no matter what the image looked like?

Selecting a baseline

- What fraction of faces in $\mathcal{D}^{\text{test}}$ are smiling faces? 54.6%
- How accurate (f_{PC}) would a predictor be that just always output 1 no matter what the image looked like?
  - 54.6%
- Note that there are other accuracy functions (e.g., f_{AUC}) that are invariant to the proportion of each class — aka the **prior probabilities** — of each class in the test set.

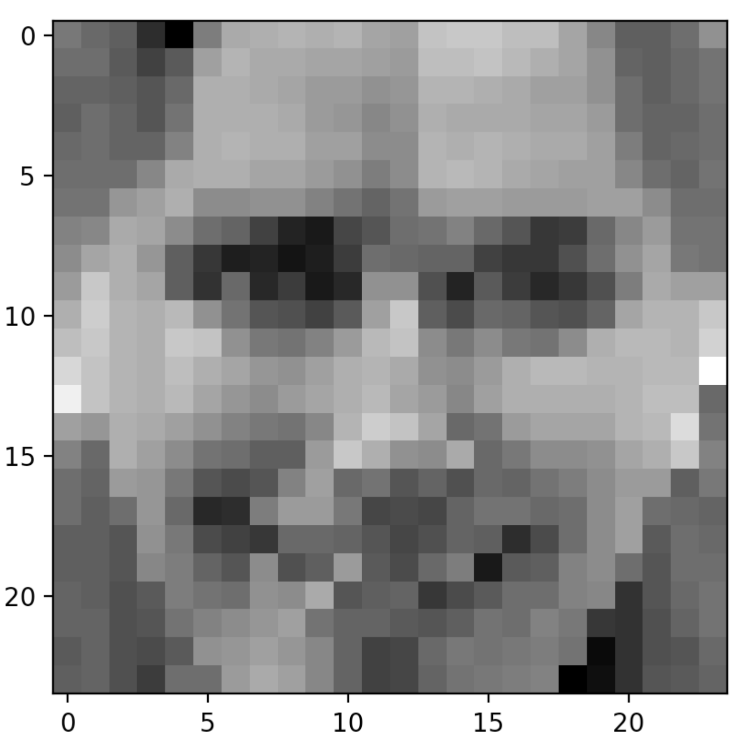
Automatic smile detection

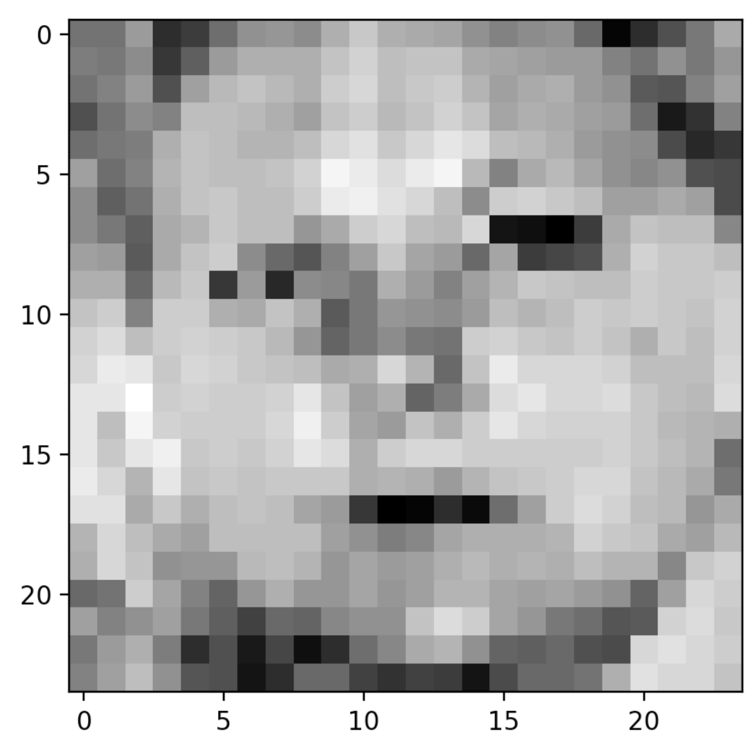
- Suppose we build g so that its output depends on only a *single* pair of pixels within the input face:

$$g(\mathbf{x}) = \mathbb{I}[\mathbf{x}_{r_1,c_1} > \mathbf{x}_{r_2,c_2}]$$

- Which pair (r_1, c_1), (r_2, c_2) would you choose?

- How good is it?





Jacob Whitehill, WPI

Combining multiple predictors

- Determining smile/non-smile based on a single comparison is very weak.

- What if we combined *multiple* pairs and took the majority-vote (choose non-smile if tied) across all *m* comparisons?

$$g^{(j)}(\mathbf{x}) = \mathbb{I}\left[\mathbf{x}_{r_1^{(j)},c_1^{(j)}} > \mathbf{x}_{r_2^{(j)},c_2^{(j)}}\right]$$

$$\hat{y} = g(\mathbf{x}) = \mathbb{I}\left[\left(\frac{1}{m}\sum_{j=1}^{m} g^{(j)}(\mathbf{x})\right) > 0.5\right]$$

Combining multiple predictors

- The accuracy of the “ensemble” can vary hugely depending on how the *m* “weak” predictors were selected.

- What would be a bad way of choosing the members of an ensemble?

Combining multiple predictors

- The accuracy of the "ensemble" can vary hugely depending on how the *m* "weak" predictors were selected.

- If the *m* weak predictors tend to give the same answer for the same inputs — i.e., they are **correlated** — then the ensemble predictor may not be much better than any of the weak predictors.

- It is important to choose the *m* weak predictors to work well in *cooperation*.

Combining multiple predictors

- Let's change notation slightly:

$$g^{(j)}\mathbf{x} = \mathbb{I}[\phi^{(j)}(\mathbf{x}) > 0]$$

$$\phi^{(j)}(\mathbf{x}) = \mathbf{x}_{r_1^{(j)},c_1^{(j)}} - \mathbf{x}_{r_2^{(j)},c_2^{(j)}}$$

- Each $\phi^{(j)}$ is called a **feature** of the input **x.**

- In machine learning, the features serve as the basis of the machine's predictions.

Feature set

- Since each $g^{(j)}$ examines only a single feature, choosing a predictor $g^{(j)}$ is equivalent to choosing a feature $\phi^{(j)}$.
- Let the set of all possible features be called $\mathcal{F}$.
- Note that each prediction $\hat{y}$ implicitly depends on $\phi^{(1)}, \ldots, \phi^{(m)}$.

$$\hat{y} = g(\mathbf{x}) = \mathbb{I}\left[\left(\frac{1}{m}\sum_{j=1}^{m} g^{(j)}(\mathbf{x})\right) > 0.5\right]$$

- Our goal is to find the **best** combination of *m* features, i.e., the one whose accuracy is:

$$\max_{(\phi^{(1)},\ldots,\phi^{(m)})\in\mathcal{F}^m} f_{\mathrm{PC}}(\mathbf{y}, \hat{\mathbf{y}})$$

Theme of the course

- Machine learning is about creating intelligent machines by solving an **optimization problem**:
  - The **objective function** is a loss/cost/accuracy function (that we either minimize or maximize) based on a set of training data.
  - The **optimization parameters** define how our machine makes its predictions/decisions/estimations.

Theme of the course

- Many classical AI methods (e.g., A*) are often based on **discrete optimization**.

- In contrast, most modern ML methods (e.g., neural networks, support vector machines) are based on **continuous optimization**.

Feature set

- What is the size of the feature set:

$$\mathcal{F} = \{(r_1, c_1, r_2, c_2) \in \{0, \ldots, 23\}^4 : (r_1, c_1) \neq (r_2, c_2)\}$$

1. 317952
2. 331200
3. 304704
4. 255024

Feature set

- What is the size of the feature set:

$$\mathcal{F} = \{(r_1, c_1, r_2, c_2) \in \{0, \ldots, 23\}^4 : (r_1, c_1) \neq (r_2, c_2)\}$$

1. 317952
2. 331200
3. 304704
4. 255024 = 24*23*22*21

Feature set

- What is the size of the feature set:

$$\mathcal{F} = \{(r_1, c_1, r_2, c_2) \in \{0, \ldots, 23\}^4 : (r_1, c_1) \neq (r_2, c_2)\}$$

1. 317952
2. 331200
3. 304704 = (24*23)*(24*23)
4. 255024 = 24*23*22*21

Feature set

- What is the size of the feature set:

$$\mathcal{F} = \{(r_1, c_1, r_2, c_2) \in \{0, \ldots, 23\}^4 : (r_1, c_1) \neq (r_2, c_2)\}$$

1. 317952 = (24*23)*(24*24)
2. 331200
3. 304704 = (24*23)*(24*23)
4. 255024 = 24*23*22*21

Feature set

- What is the size of the feature set:

$$\mathcal{F} = \{(r_1, c_1, r_2, c_2) \in \{0, \ldots, 23\}^4 : (r_1, c_1) \neq (r_2, c_2)\}$$

1. 317952 = (24*23)*(24*24)
2. **331200 = (24*24)*(24*24-1)**
3. 304704 = (24*23)*(24*23)
4. 255024 = 24*23*22*21

Feature set

- The size of the feature set is rather large:

$$\mathcal{F} = \{(r_1, c_1, r_2, c_2) \in \{0, \ldots, 23\}^4 : (r_1, c_1) \neq (r_2, c_2)\}$$

- It contains (24*24)*(24*24-1)=331200 elements.

Feature set

- If $|\mathcal{F}| = 331200$, then even for *m*=5, we have

  $|\mathcal{F}^5|$ = 3985213938015928320000000000

- It is computationally intractable to enumerate over all of these combinations of features!

- Overcoming the exponential computational costs of **brute-force** ("try everything") optimization is one fo the chief goals of ML research.

Step-wise classification

- Step-wise regression/classification is a **greedy** algorithm for selecting features/predictors **myopically**, i.e., based on “what looks best right now”.

- Instead of optimizing *jointly* to find:

$$\max_{(\phi^{(1)},\ldots,\phi^{(m)})\in\mathcal{F}^m} f_{\mathrm{PC}}(\mathbf{y}, \hat{\mathbf{y}}; \phi^{(1)}, \ldots, \phi^{(m)})$$

We sometimes write the parameters that a function depends on after the ;

Jacob Whitehill, WPI

Step-wise classification

- Step-wise regression/classification is a **greedy** algorithm for selecting features/predictors **myopically**, i.e., based on "what looks best right now".

- Instead of optimizing *jointly* to find:

$$\max_{(\phi^{(1)},\ldots,\phi^{(m)})\in\mathcal{F}^m} f_{\mathrm{PC}}(\mathbf{y},\hat{\mathbf{y}};\phi^{(1)},\ldots,\phi^{(m)})$$

…we optimize *iteratively*:

$$\max_{\phi^{(1)}\in\mathcal{F}} f_{\mathrm{PC}}(\mathbf{y},\hat{\mathbf{y}};\phi^{(1)})$$

Find the single best feature.

Step-wise classification

- Step-wise regression/classification is a **greedy** algorithm for selecting features/predictors **myopically**, i.e., based on “what looks best right now”.

- Instead of optimizing *jointly* to find:

$$\max_{(\phi^{(1)},\ldots,\phi^{(m)})\in\mathcal{F}^m} f_{\mathrm{PC}}(\mathbf{y},\hat{\mathbf{y}};\phi^{(1)},\ldots,\phi^{(m)})$$

…we optimize *iteratively*:

$$\max_{\phi^{(1)}\in\mathcal{F}} f_{\mathrm{PC}}(\mathbf{y},\hat{\mathbf{y}};\phi^{(1)})$$

$$\max_{\phi^{(2)}\in\mathcal{F}} f_{\mathrm{PC}}(\mathbf{y},\hat{\mathbf{y}};\phi^{(1)},\phi^{(2)})$$

Given we have already committed to the first feature, which single next feature is best in combination?

Step-wise classification

- Step-wise regression/classification is a **greedy** algorithm for selecting features/predictors **myopically**, i.e., based on “what looks best right now”.

- Instead of optimizing *jointly* to find:

$$\max_{(\phi^{(1)},\ldots,\phi^{(m)})\in\mathcal{F}^m} f_{\mathrm{PC}}(\mathbf{y}, \hat{\mathbf{y}}; \phi^{(1)}, \ldots, \phi^{(m)})$$

…we optimize *iteratively*:

$$\max_{\phi^{(1)}\in\mathcal{F}} f_{\mathrm{PC}}(\mathbf{y}, \hat{\mathbf{y}}; \phi^{(1)})$$

$$\max_{\phi^{(2)}\in\mathcal{F}} f_{\mathrm{PC}}(\mathbf{y}, \hat{\mathbf{y}}; \phi^{(1)}, \phi^{(2)})$$

$$\max_{\phi^{(3)}\in\mathcal{F}} f_{\mathrm{PC}}(\mathbf{y}, \hat{\mathbf{y}}; \phi^{(1)}, \phi^{(2)}, \phi^{(3)})$$

Repeat.

…

Step-wise classification

- Instead of $|\mathcal{F}|^m$ possible choices, we only have $m \times |\mathcal{F}|$.

- This is doable!

- We have reduced the exponential growth into linear growth — big difference!

- Note, however, that there is no guarantee that the solution is optimal. Step-wise classification is an **approximate solution** to selecting the *m* best features/predictors.

$$\max_{\phi^{(1)} \in \mathcal{F}} f_{\mathrm{PC}}(\mathbf{y}, \hat{\mathbf{y}}; \phi^{(1)})$$

$$\max_{\phi^{(2)} \in \mathcal{F}} f_{\mathrm{PC}}(\mathbf{y}, \hat{\mathbf{y}}; \phi^{(1)}, \phi^{(2)})$$

$$\max_{\phi^{(3)} \in \mathcal{F}} f_{\mathrm{PC}}(\mathbf{y}, \hat{\mathbf{y}}; \phi^{(1)}, \phi^{(2)}, \phi^{(3)})$$

...

Step-wise classification

- Pseudocode:

```
predictors = []  # Empty list
For j = 1, ..., m:
  1. Find next best predictor given what's already in predictors
  2. Add it to predictors
```

- Run smile_demo.py and optimize on 10 images.