Overfitting

Jiaying Gong
Ph.D. candidate in Machine Learning Lab
Department of Computer Science

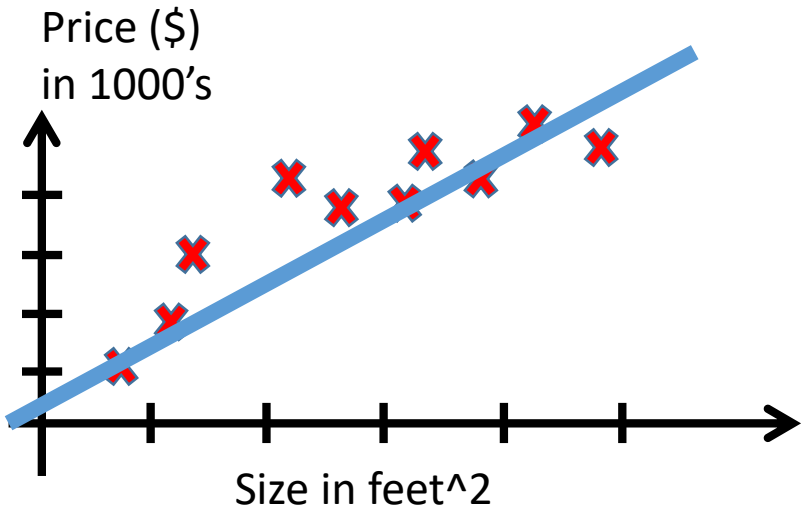
Objectives

- Broad knowledge of overfitting.
- Get a general understanding of different techniques to prevent overfitting.
- Mathematically understanding of PCA and L1 Regularization.

Overfitting

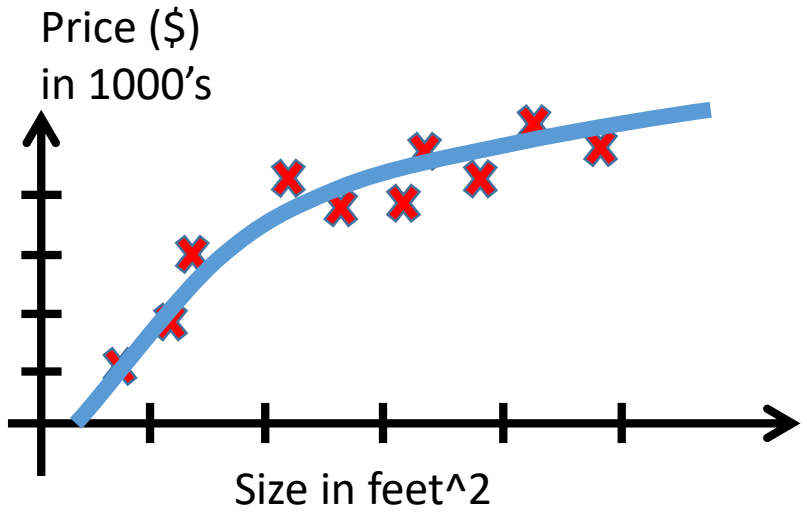
- If we have **too many features** (i.e. complex model), the learned hypothesis may fit the training set very well but fail to generalize to new examples (predict results on new examples).
- The **long training time** of the model or its **architectural complexity** may cause the model to overfit. If the model trains for too long on the training data or is too complex, it learns the noise or irrelevant information within the dataset.

Example: Linear regression



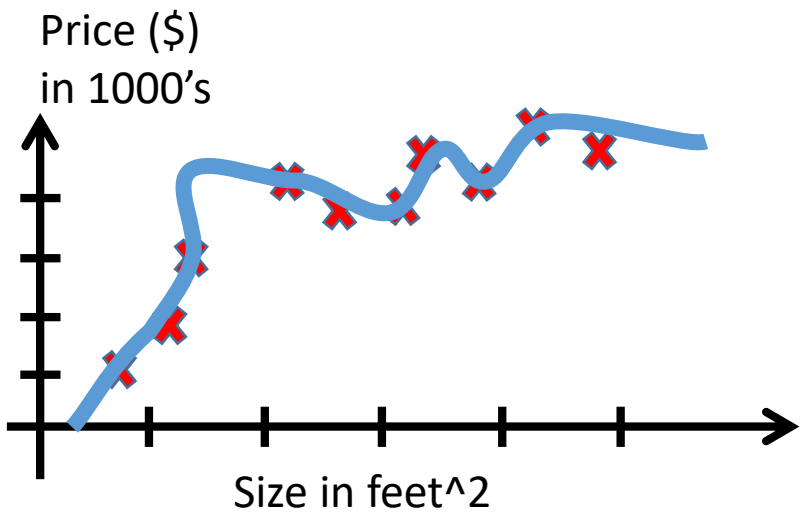
$h_\theta(x) = \theta_0 + \theta_1 x$

Underfitting



$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$

Just right



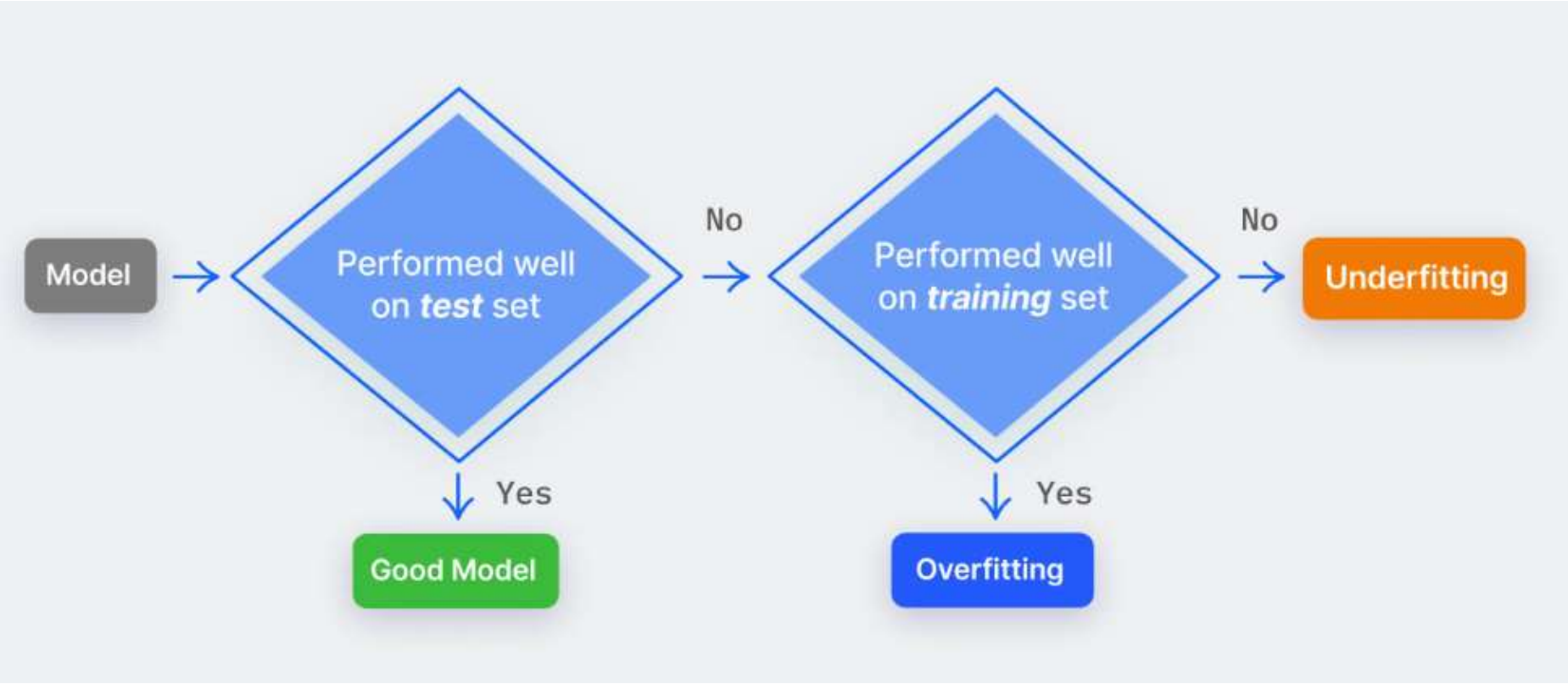
$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \cdots$

Overfitting

Overfitting vs Underfitting

- Underfitting occurs when we have a high bias in our data, i.e., we are oversimplifying the problem, and as a result, the model does not work correctly in the training data.
- Overfitting occurs when the model has a high variance, i.e., the model performs well on the training data but does not perform accurately in the evaluation set. The model memorizes the data patterns in the training dataset but fails to generalize to unseen examples.

Overfitting vs Underfitting



How to prevent overfitting?

How to avoid overfitting?

- Input Data
  - Train with more data (clean data)
  - Data augmentation
- Feature Engineering – (too many features):
  - Feature Extraction (Dimension Reduction)
  - Feature Selection
- Model – (architectural complexity):
  - Regularization
  - Dropout Layers
- Training Stage – (long training time):
  - Cross-Validation
  - Early Stopping

How to avoid overfitting?

- **Input Data**
  - **Train with more data (clean data)**
  - **Data augmentation**
- Feature Engineering:
  - Feature Extraction (Dimension Reduction)
  - Feature Selection
- Model:
  - Regularization
  - Dropout Layers
- Training Stage:
  - Cross-Validation
  - Early Stopping

What are data augmentation techniques?

Data Augmentation

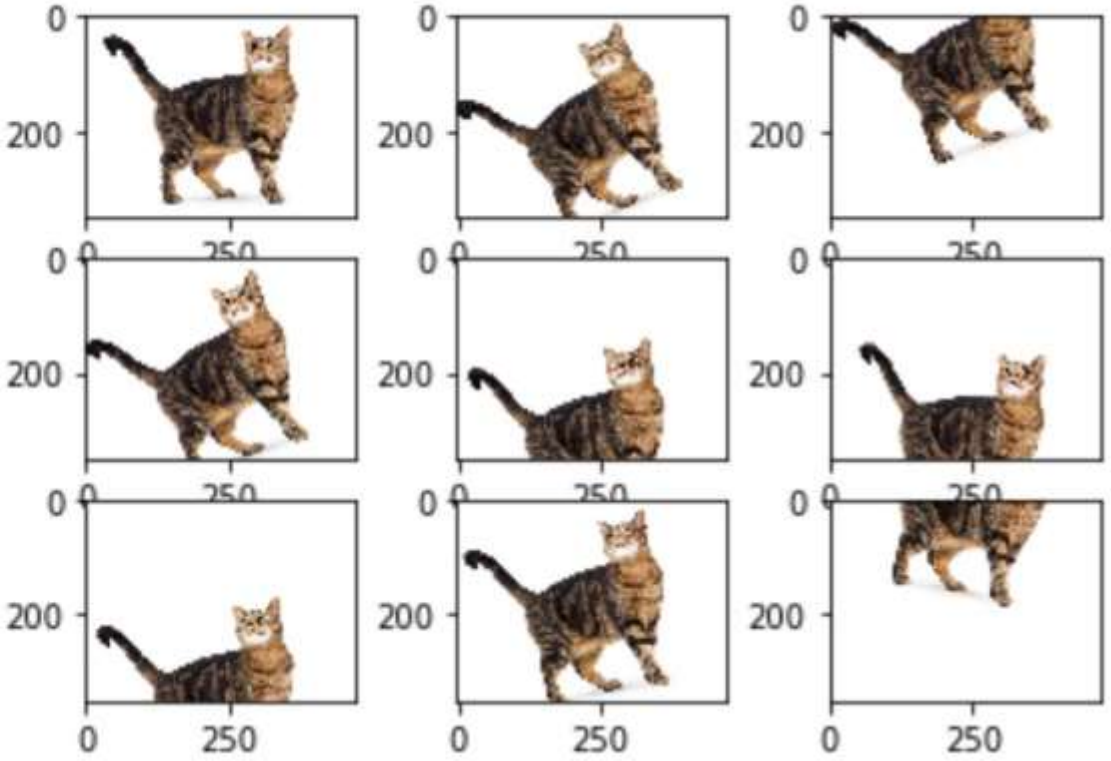
- Computer Vision (CV):
  - Image mirroring, scaling, rotation, flipping, zooming, etc.
  - GAN models (i.e. CycleGAN)
- Natural Language Processing (NLP):
  - Back translation
  - GPT-2 for generation
- Time-series Prediction:
  - Jitter, inject noise, shifting, scaling, etc.

Data Augmentation

- **Computer Vision (CV):**
  - **Image mirroring, scaling, rotation, flipping, zooming, etc.**
  - **GAN models (i.e. CycleGAN)**
- Natural Language Processing (NLP):
  - Back translation
  - GPT-2 for generation
- Time-series Prediction:
  - Jitter, inject noise, shifting, scaling, etc.

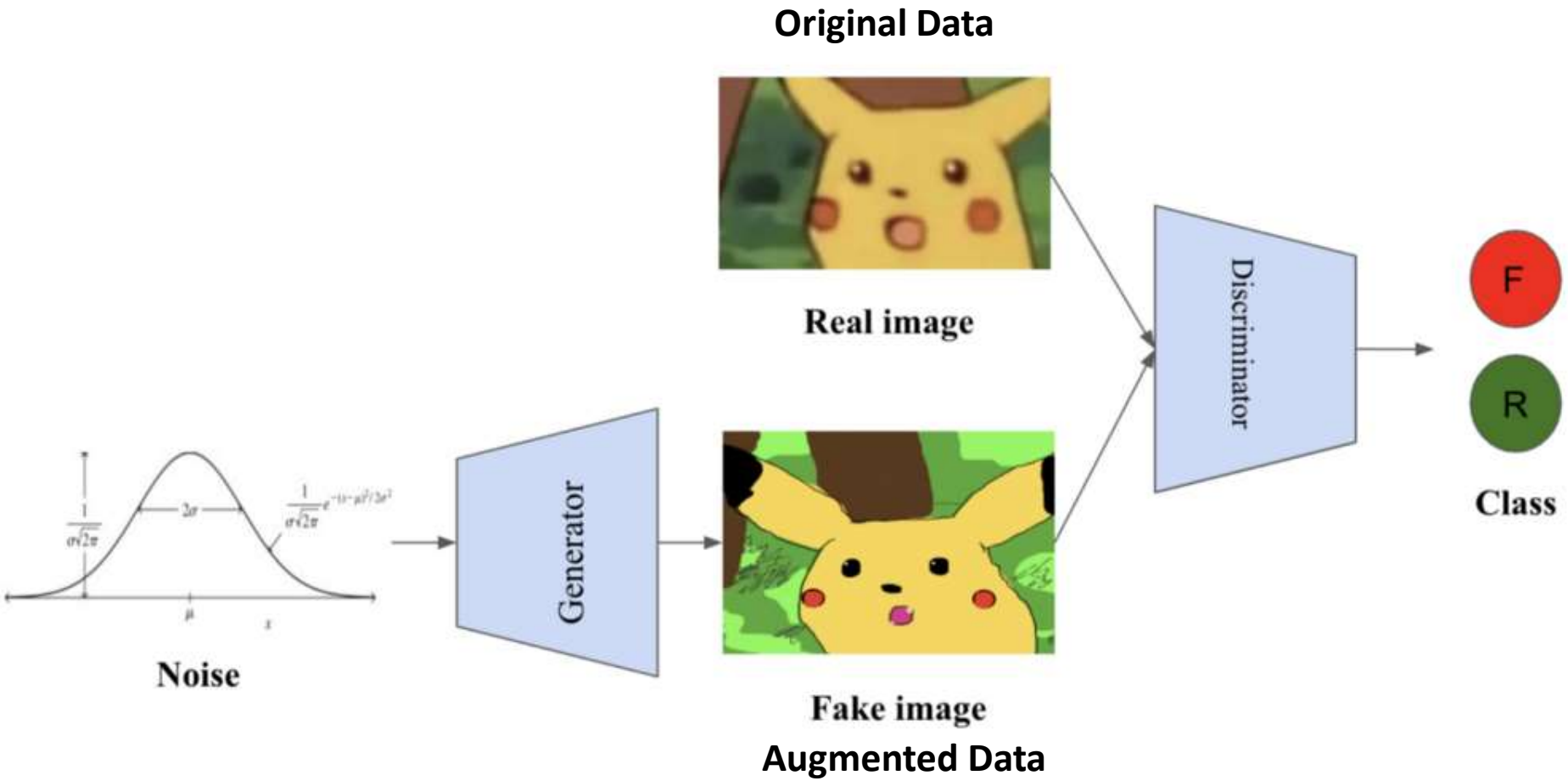
Data Augmentation - CV

Original Data



Augmented Data

Data Augmentation - CV



Data Augmentation

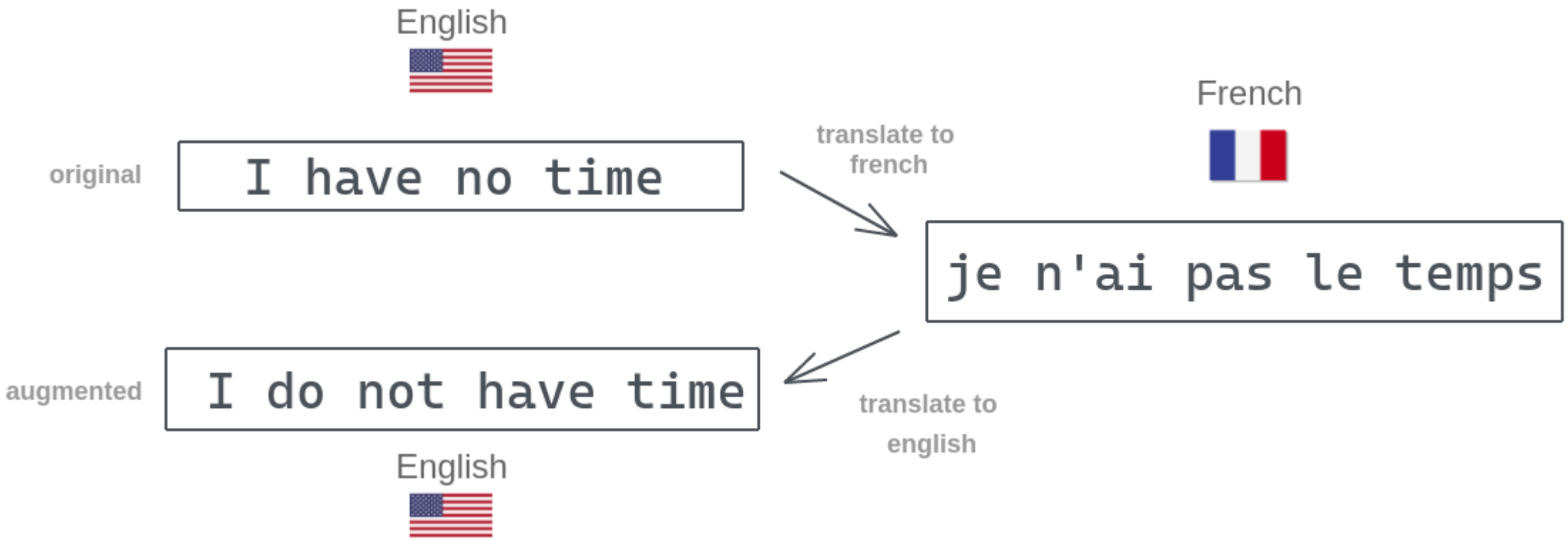
- Computer Vision (CV):
  - Image mirroring, scaling, rotation, flipping, zooming, etc.
  - GAN models (i.e. CycleGAN)
- **Natural Language Processing (NLP):**
  - **Back translation**
  - **GPT-2 for generation**
- Time-series Prediction:
  - Jitter, inject noise, shifting, scaling, etc.

Data Augmentation - NLP

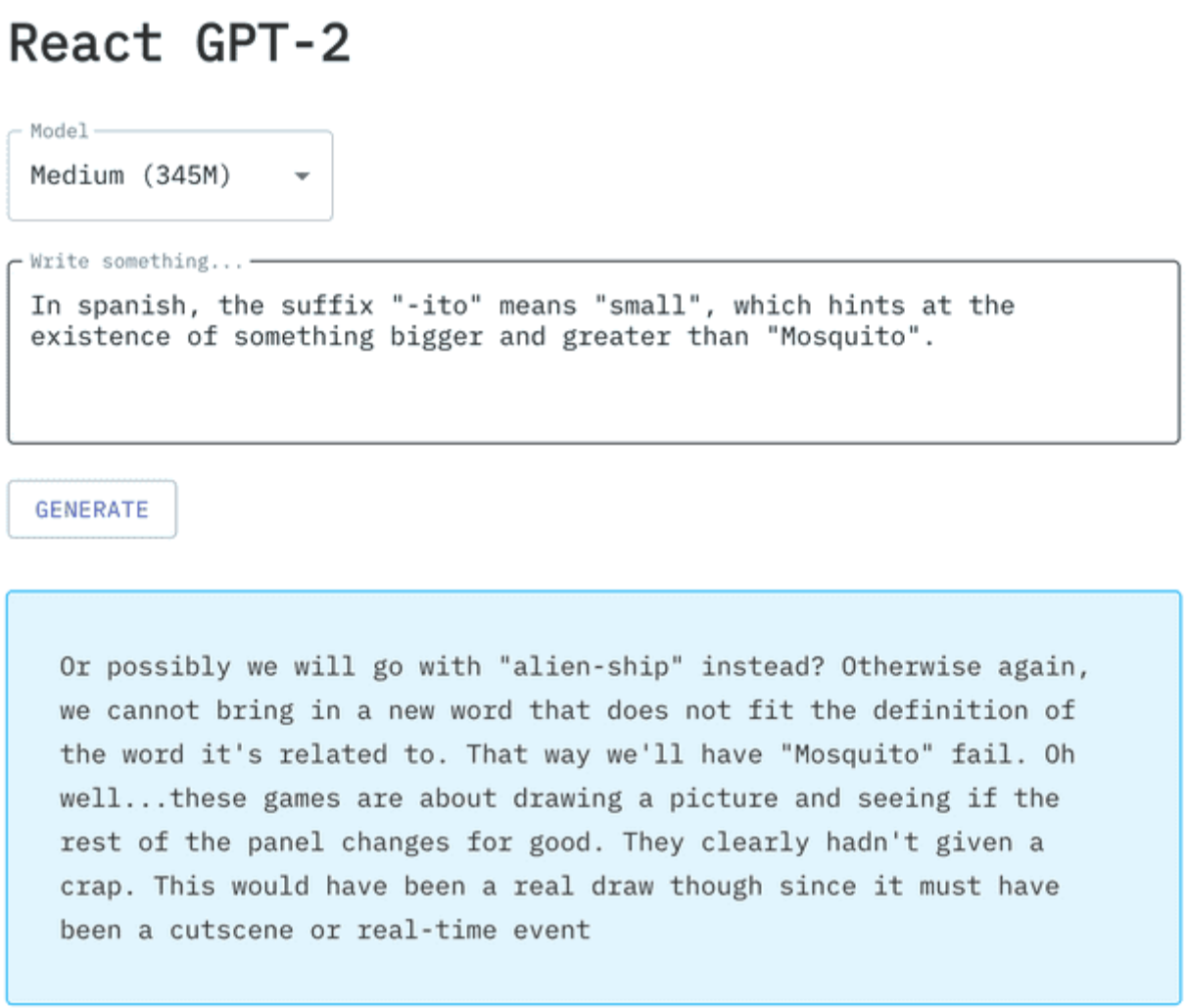
- Word Substitution

Data Augmentation - NLP

- Back Translation



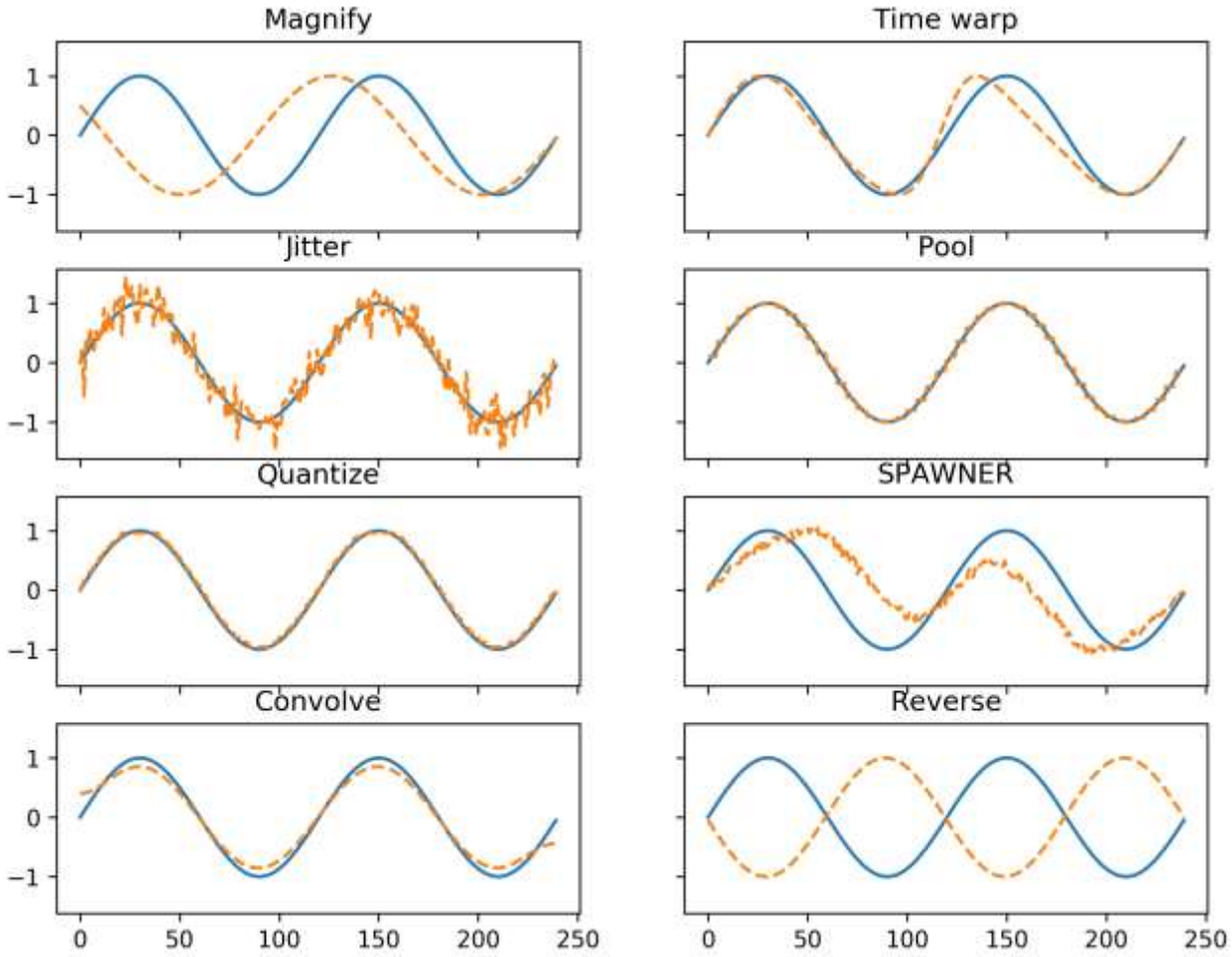
Data Augmentation - NLP



Data Augmentation

- Computer Vision (CV):
  - Image mirroring, scaling, rotation, flipping, zooming, etc.
  - GAN models (i.e. CycleGAN)
- Natural Language Processing (NLP):
  - Back translation
  - GPT-2 for generation
- **Time-series Prediction:**
  - **Jitter, inject noise, shifting, scaling, etc.**

Data Augmentation – Time-series



How to avoid overfitting?

- Input Data
  - Train with more data (clean data)
  - Data augmentation
- **Feature Engineering:**
  - **Feature Extraction (Dimension Reduction)**
  - Feature Selection
- Model:
  - Regularization
  - Dropout Layers
- Training Stage:
  - Cross-Validation
  - Early Stopping

What are features?

Features

Labels

Student_ID	Gender	Prev_Exam_Marks	Height (cm)	Weight Caregory (kgs)	Play Cricket
S001	M	65	178	61	1
S002	F	75	174	56	0
S003	M	45	163	62	1
S004	M	57	175	70	0
S005	F	59	162	67	0

Curse of Dimensionality

- *If 0.5*red + 0.3*green + 0.2*blue > 0.6 : return cat; else return dog;*

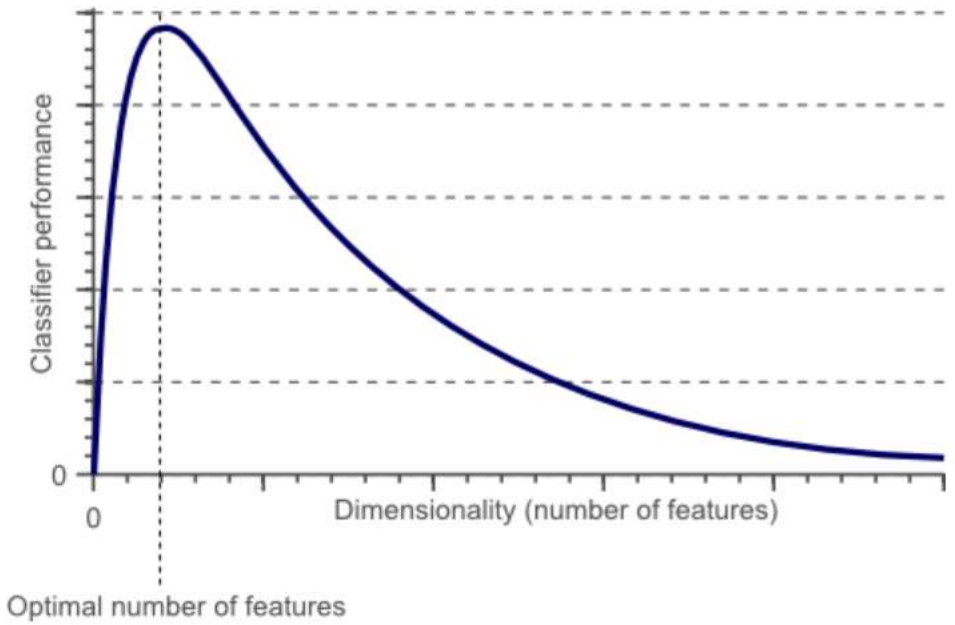


Figure 1. As the dimensionality increases, the classifier's performance increases until the optimal number of features is reached. Further increasing the dimensionality without increasing the number of training samples results in a decrease in classifier performance.

Curse of Dimensionality

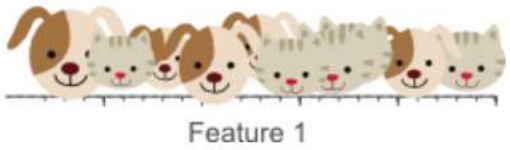


Figure 2. A single feature does not result in a perfect separation of our training data.

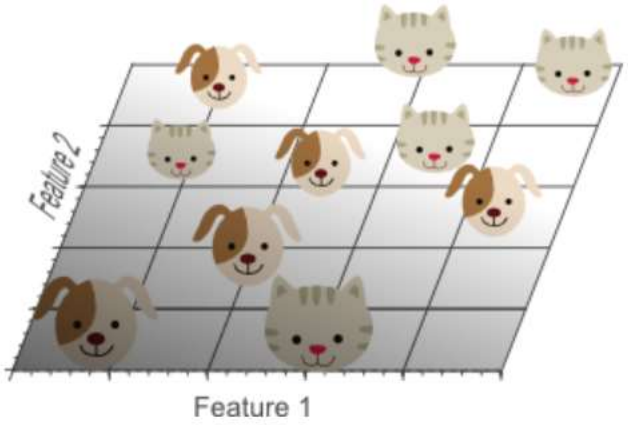


Figure 3. Adding a second feature still does not result in a linearly separable classification problem: No single line can separate all cats from all dogs in this example.

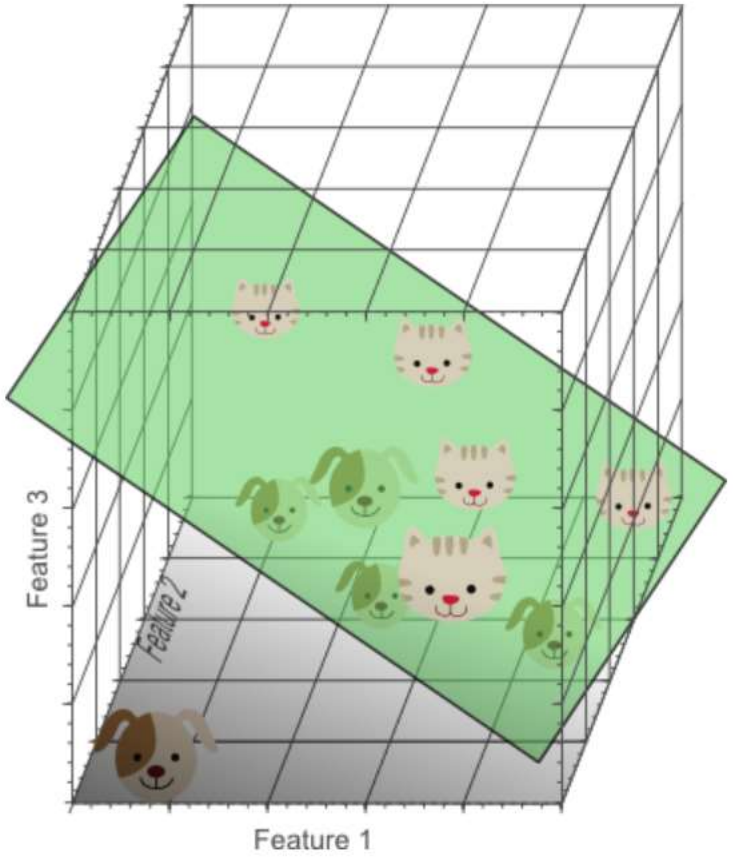


Figure 5. The more features we use, the higher the likelihood that we can successfully separate the classes perfectly.

Curse of Dimensionality

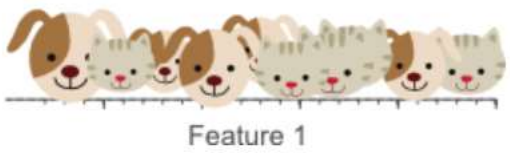


Figure 2. A single feature does not result in a perfect separation of our training data.

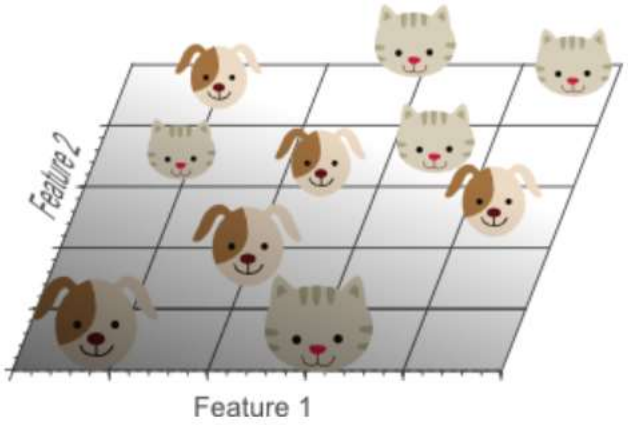


Figure 3. Adding a second feature still does not result in a linearly separable classification problem: No single line can separate all cats from all dogs in this example.

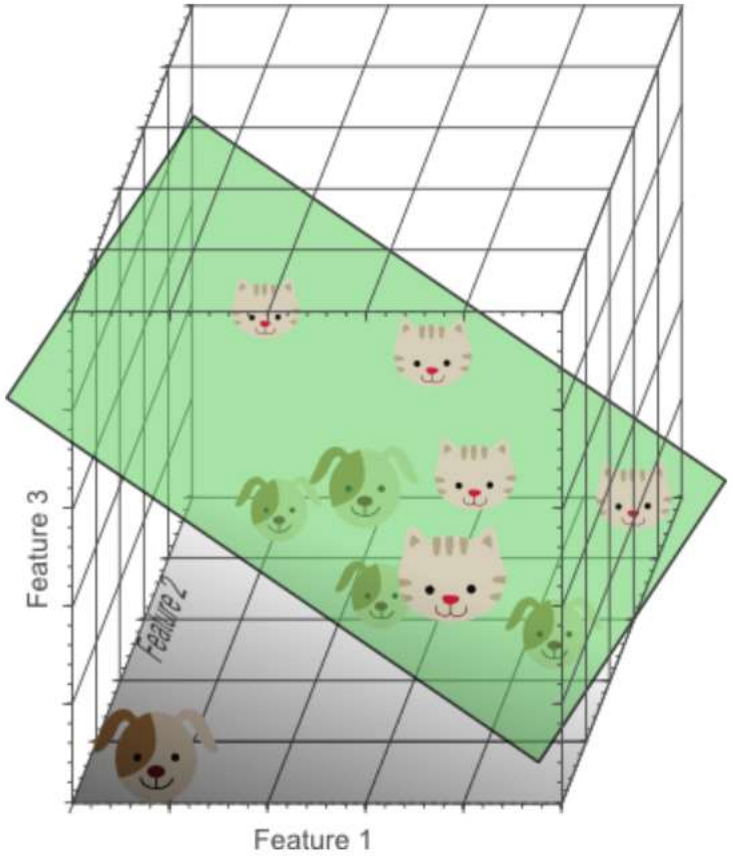


Figure 5. The more features we use, the higher the likelihood that we can successfully separate the classes perfectly.

Problem: The density of the training samples decreased exponentially when we increased the dimensionality of the problem.

Curse of Dimensionality

- In the **1D** case (figure 2), 10 training instances covered the complete 1D feature space, the width of which was 5 unit intervals. Therefore, in the 1D case, the sample density was **10/5=2 samples/interval**. In the 2D case however (figure 3), we still had 10 training instances at our disposal, which now cover a **2D** feature space with an area of 5×5=25 unit squares. Therefore, in the 2D case, the sample density was **10/25 = 0.4 samples/interval**. Finally, in the **3D** case, the 10 samples had to cover a feature space volume of 5x5x5=125 unit cubes. Therefore, in the 3D case, the sample density was **10/125 = 0.08 samples/interval.**

Curse of Dimensionality

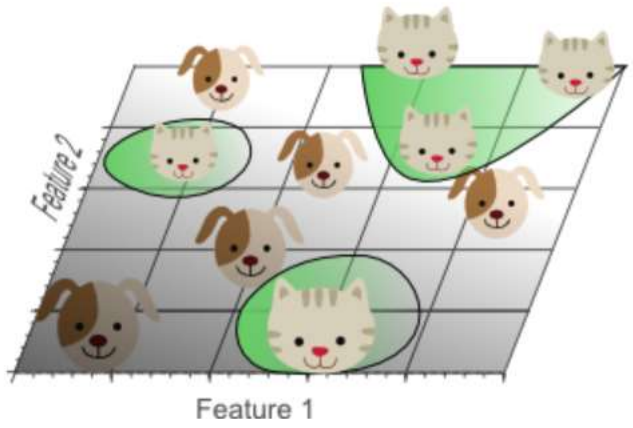


Figure 6. Using too many features results in overfitting. The classifier starts learning exceptions that are specific to the training data and do not generalize well when new data is encountered.

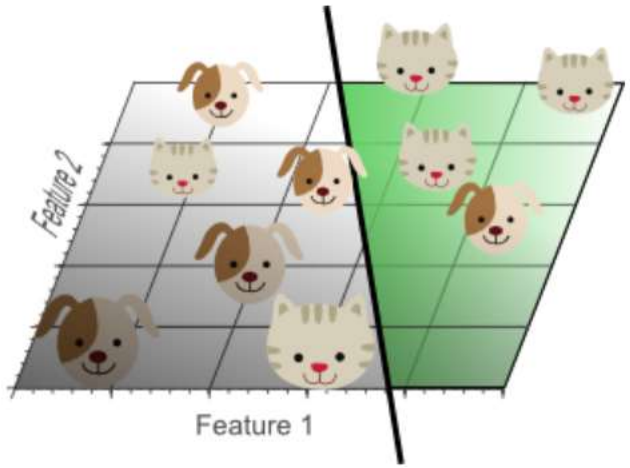


Figure 7. Although the training data is not classified perfectly, this classifier achieves better results on unseen data than the one from figure 5.

Dimension Reduction

- Dimensionality reduction is a method of converting the high dimensional variables into lower dimensional variables without changing the specific information of the variables.
- Dimensionality Reduction is used to reduce the feature space with consideration by a set of principal features.

Dimension Reduction - PCA

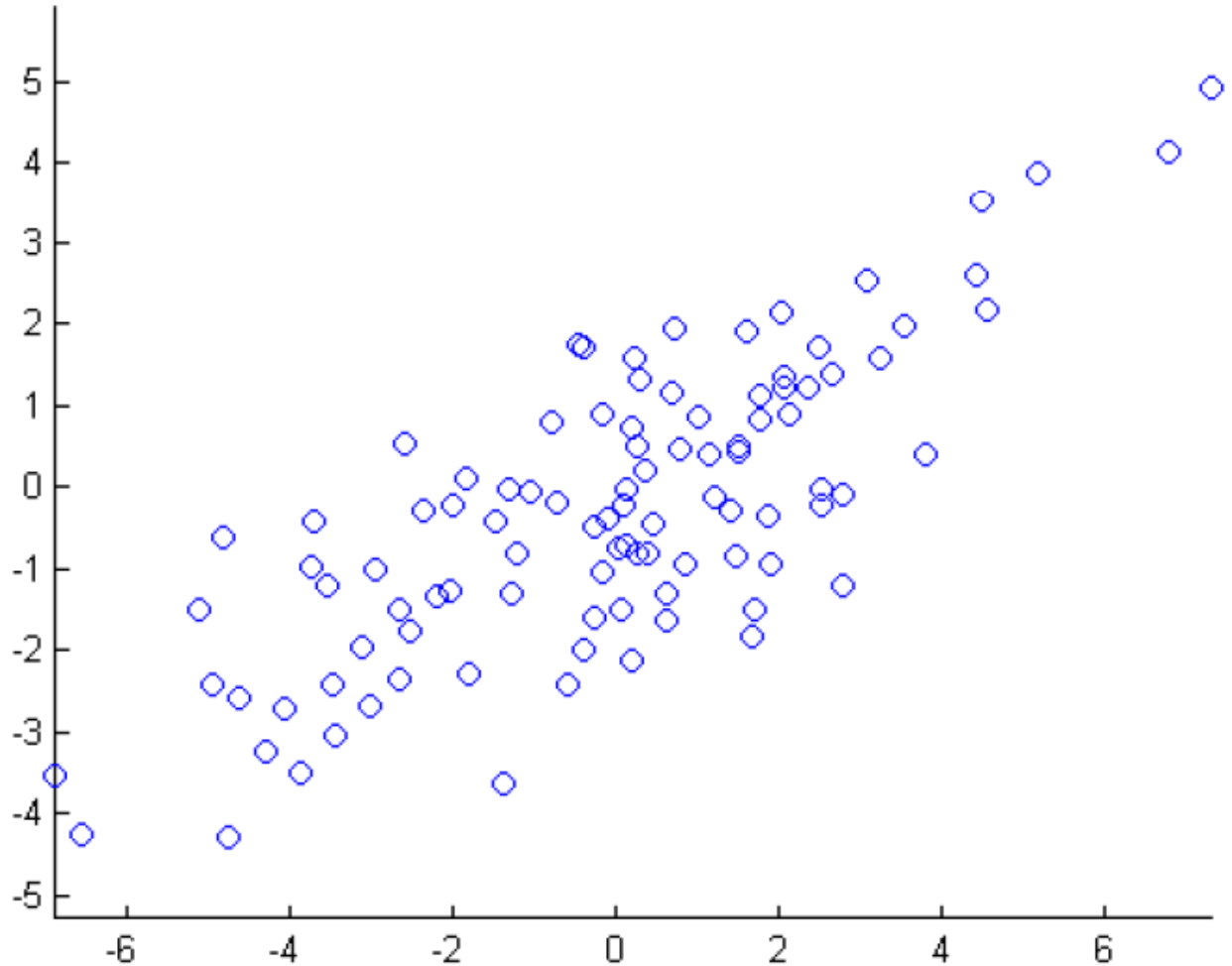
- Principal component analysis (PCA) is a technique that transforms high-dimensions data into lower-dimensions while retaining as much information as possible.

Example

- Set of 2-D data

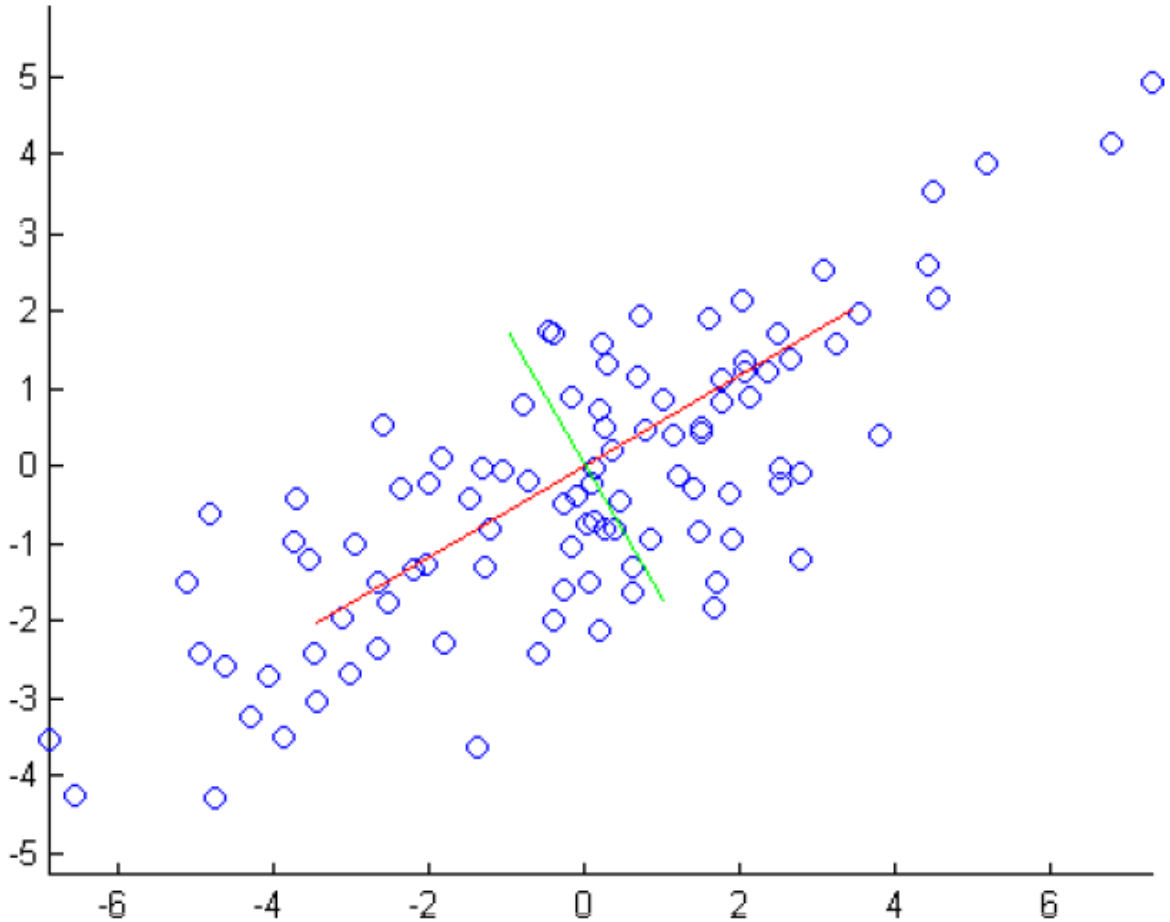
x=

-3.0139	-2.6748
0.1810	-2.1227
-6.5559	-4.2424
1.5960	1.9161
2.7825	-1.2017
2.0292	2.1421
-2.5200	-1.7787
0.2340	-0.8122
0.8263	-0.9407
-4.6227	-2.5955
-4.3052	-3.2307
-2.6013	0.5435
3.5409	1.9755
5.1704	3.8629
.	.
.	.
.	.

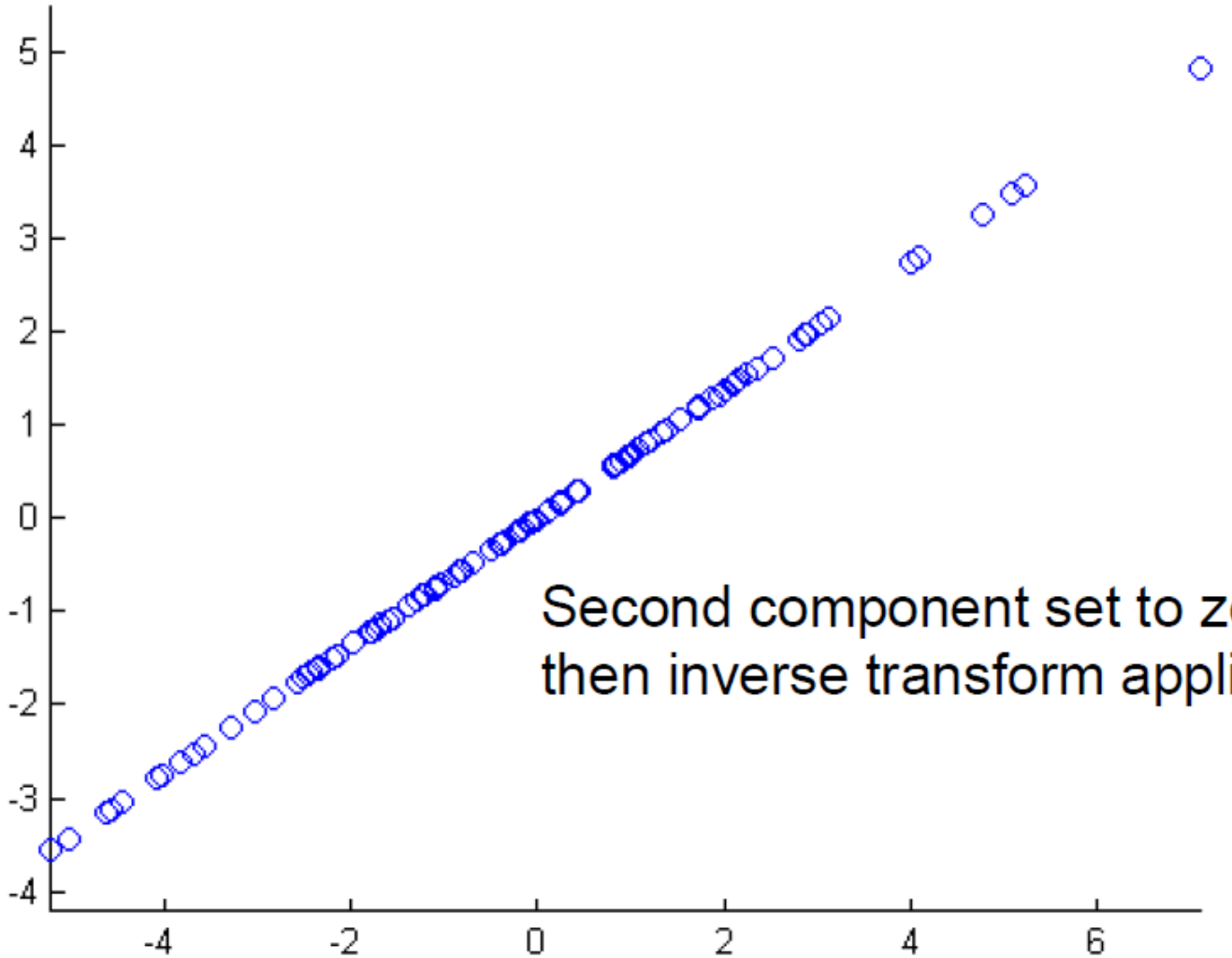


Principal components visualised

- red line represents direction of first principal component
  - line of greatest variation
- green line is direction of second principal component
  - perpendicular to red line.
- When there are more than 2 dimensions,
  - next component along line of next greatest variation



PCA is used to reduce data dimensionality while retaining the most information



Understand PCA mathematically

- Step 1: Calculate the covariance matrix for the features in the dataset.
- Step 2: Calculate the eigenvalues and eigenvectors for the covariance matrix.
- Step 3: Sort eigenvalues and their corresponding eigenvectors.
- Step 4: Pick k eigenvalues and form a matrix of eigenvectors.
- Step 5: Transform the original matrix.

PCA

- Step 1: Calculate the covariance matrix for the whole dataset

$$\mathrm{Cov}(x, y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n}$$

$$\text{Covariance Matrix} = \begin{bmatrix} COV(X,X) & COV(X,Y) \\ COV(Y,X) & COV(Y,Y) \end{bmatrix}$$

PCA

- Step 2: Calculate eigenvalues and eigen vectors.
- Let A be the covariance matrix, ν a vector and λ a scalar that satisfies $A\nu = \lambda\nu$, then λ is called eigenvalue associated with eigenvector ν of A.
- $A\nu = \lambda\nu$ -> $(A-\lambda E)\,\nu=0$
- Because ν is a non-zero vector, $|\ A-\lambda E\ |=0$

PCA

- Step 3: Sort eigenvalues and their corresponding eigenvectors.
- Step 4: Pick k eigenvalues and form a matrix of eigenvectors.
- Step 5: Transform the original matrix.

Exercise for PCA

	f1	f2
D1	-1	-2
D2	-1	0
D3	0	0
D4	2	1
D5	0	1

Find the eigenvalue and the transformed matrix

Question: What's the meaning if eigenvalues are similar?

Solution

$$\begin{bmatrix} -1 & -2 \\ -1 & 0 \\ 0 & 0 \\ 2 & 1 \\ 0 & 1 \end{bmatrix} \longrightarrow Cov = \begin{bmatrix} \frac{6}{5} & \frac{4}{5} \\ \frac{4}{5} & \frac{6}{5} \end{bmatrix} \longrightarrow \begin{vmatrix} \frac{6}{5} - \lambda & \frac{4}{5} \\ \frac{4}{5} & \frac{6}{5} - \lambda \end{vmatrix} = (\frac{6}{5} - \lambda)^2 - \frac{16}{25} = 0 \longrightarrow \lambda_1 = 2, \lambda_2 = \frac{2}{5}$$

$$\lambda_1 = 2: \quad \begin{bmatrix} -\frac{4}{5} & \frac{4}{5} \\ \frac{4}{5} & -\frac{4}{5} \end{bmatrix} \vec{v} = 0 \longrightarrow \begin{matrix} -\frac{4}{5}x_1 + \frac{4}{5}x_2 = 0 \\ \frac{4}{5}x_1 - \frac{4}{5}x_2 = 0 \end{matrix} \longrightarrow x_1 = x_2$$

$$\lambda_2 = \frac{2}{5} \quad \begin{bmatrix} \frac{4}{5} & \frac{4}{5} \\ \frac{4}{5} & \frac{4}{5} \end{bmatrix} \vec{v} = 0 \longrightarrow \begin{matrix} \frac{4}{5}x_1 + \frac{4}{5}x_2 = 0 \\ \frac{4}{5}x_1 + \frac{4}{5}x_2 = 0 \end{matrix} \longrightarrow x_1 = -x_2$$

Solution

$$P_1^T = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

$$\begin{bmatrix} -1 & -2 \\ -1 & 0 \\ 0 & 0 \\ 2 & 1 \\ 0 & 1 \end{bmatrix}$$

$$Y = \begin{bmatrix} -\frac{3}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \\ 0 \\ \frac{3}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}$$

$$P_2^T = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}$$

Result

```python
import numpy as np
import pandas as pd
A = np.matrix([[-1, -2], [-1, 0], [0, 0], [2, 1], [0, 1]])
df = pd.DataFrame(A,columns  = ['f1', 'f2'])
df_cov = np.cov(df.T, bias = 1)
print('covariance matrix:\n', df_cov)

eigen_val, eigen_vectors = np.linalg.eig(df_cov)
print('eigen value:',eigen_val)
print('eigen vector:\n', eigen_vectors)

n_components=1
top_eigen_vectors = eigen_vectors[:,:n_components]
print('selected top eigen vector:\n', top_eigen_vectors)

transformed_data = top_eigen_vectors.T*A.T
print('Data after PCA:',transformed_data)
```

```
covariance matrix:
 [[1.2 0.8]
 [0.8 1.2]]
eigen value: [2.  0.4]
eigen vector:
 [[ 0.70710678 -0.70710678]
 [ 0.70710678  0.70710678]]
selected top eigen vector:
 [[0.70710678]
 [0.70710678]]
Data after PCA: [[-2.12132034 -0.70710678  0.          2.12132034  0.70710678]]
```

How to avoid overfitting?

- Input Data
  - Train with more data (clean data)
  - Data augmentation
- **Feature Engineering:**
  - Feature Extraction (Dimension Reduction)
  - **Feature Selection**
- Model:
  - Regularization
  - Dropout Layers
  - Ensemble
- Training Stage:
  - Cross-Validation
  - Early Stopping

Feature Selection

- Aim:
  - Simplify the Models
  - Shorten Training Time
  - Avoid the Curse of Dimensionality
  - Enhance Generalization by Reducing Overfitting

- There is no fixed rule that defines how many features should be used in a classification problem.
- In fact, this depends on the amount of training data available, the complexity of the decision boundaries, and the type of classifier used.

FEATURE SELECTION METHOD

- **Filter**
  - Removing features with low variance
  - Univariate feature selection
    - Chi2 (Y Discrete)
    - Pearson Correlation (Y Continuous)
    - Mutual Information (Y Discrete) and Maximal Information Coefficient (Y Continuous)
    - Distance Correlation
    - Model Based Ranking
- Wrapper
  - Recursive Feature Elimination
- Embedded
  - Feature selection using SelectFromModel
    - L1-based Feature Selection
    - Randomized Sparse Models
    - Tree-based Feature Selection
  - Feature selection as part of a pipeline

Chi-Square test (x^2 feature selection)

- In statistics, the test is applied to test the independence of two events, where two events A and B are defined to be independent if $P(AB) = P(A)P(B)$. In feature selection, the two events are occurrence of the term and occurrence of the class.

$$X^2(\mathbb{D}, t, c) = \sum_{e_t \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{(N_{e_t e_c} - E_{e_t e_c})^2}{E_{e_t e_c}}$$

EXAMPLE

$$X^2(\mathbb{D},t,c)=\sum_{e_t\in\{0,1\}}\sum_{e_c\in\{0,1\}}\frac{(N_{e_te_c}-E_{e_te_c})^2}{E_{e_te_c}}$$

	GENE1	GENE2	GENE3	GENE4
Disease	0.73	0.24	0.21	0.54
No Disease	0.71	0.26	0.87	0.55

Sample No.	GENE3	Disease
1	1	0
2	1	0
3	1	1
4	1	0
5	1	0
6	0	1
7	1	0
8	0	1
9	1	0
10	0	1

N_{11}=1
N_{01}=3
N_{10}=6
N_{00}=0

$$E_{11}=10\times P(t=1)\times P(c=1)=2.8$$

$$N_{01}=10\times P(t=0)\times P(c=1)=1.2$$

$$N_{10}=10\times P(t=1)\times P(c=0)=4.2$$

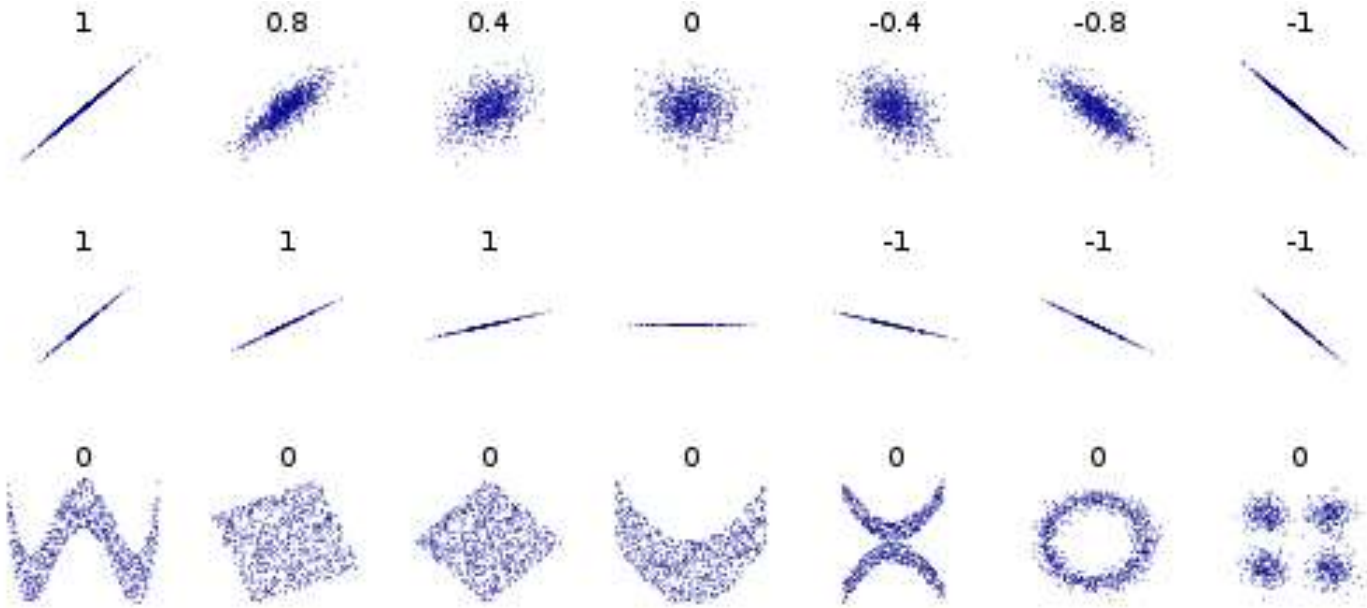
$$N_{00}=10\times P(t=0)\times P(c=0)=1.8$$

$$X^2(t,c)=\frac{(1-2.8)^2}{2.8}+\frac{(3-1.2)^2}{1.2}+\frac{(6-4.2)^2}{4.2}+\frac{(0-1.8)^2}{1.8}=6.4286$$

P-value	X^2
0.1	2.71
0.05	3.84
0.01	6.63
0.005	7.88
0.001	10.83

Pearson Correlation

- A Pearson correlation is a number between -1 and 1 that indicates the extent to which two variables are linearly related.



Pearson Correlation

$$\mathrm{COV}(X, Y) = \frac{1}{n-1} \sum_1^n (X_i - \overline{X})(Y_i - \overline{Y})$$

$$\mathrm{COR}(X, Y) = \frac{\sum_1^n (X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_1^n (X_i - \overline{X})^2 \sum_1^n (Y_i - \overline{Y})^2}}$$

Answer:

```
from scipy.stats import pearsonr

data1 = [-1,-1,0,2,0]
data2 = [-2,0,0,1,1]
corr, _ = pearsonr(data1, data2)
print(corr)
```

```
0.6666666666666669
```

Mutual Information and Maximal Information Coefficient

- Mutual Information of two random variables is a measure of the mutual dependence between the two variables. More specifically, it quantifies the amount of information obtained about one random variable through observing the other random variable.

$$I(X;Y) = E[I(x_i;y_j)] = \sum_{x_i \epsilon X} \sum_{y_j \epsilon Y} p(x_i, y_j) log \frac{p(x_i, y_j)}{p(x_i)p(y_j)}$$

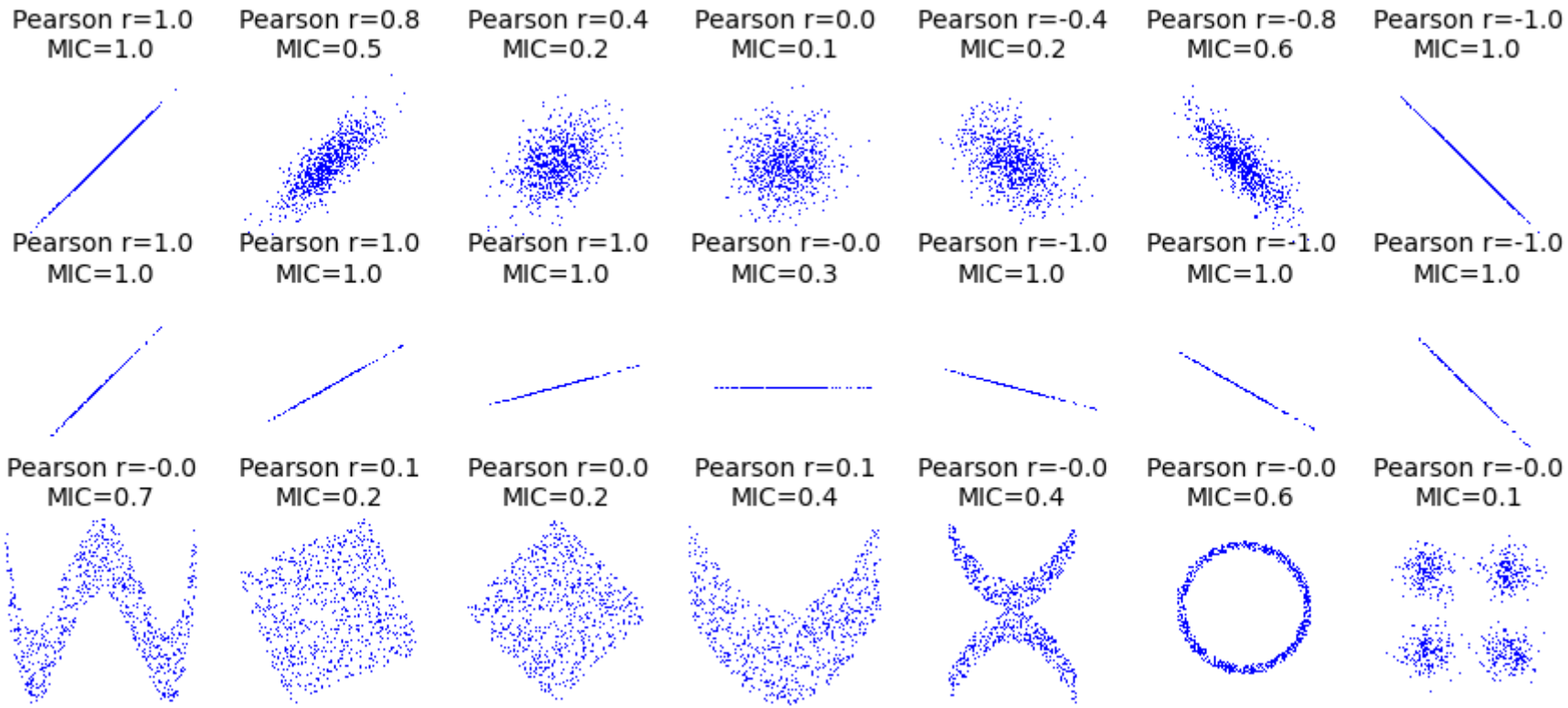
Mutual Information and Maximal Information Coefficient

- Maximal information coefficient is a measure of the strength of the linear or non-linear association between two variables X and Y.

$$I(x;y) = \int p(x,y) log_2 \frac{p(x,y)}{p(x)p(y)} \mathrm{d}x\mathrm{d}y$$

$$mic(x;y) = \max_{a*b<B} \frac{I(x;y)}{log_2 \min(a,b)}$$

Mutual Information and Maximal Information Coefficient



Mutual Information

```
from sklearn.metrics import mutual_info_score
data1 = [-1,-1,0,2,0]
data2 = [-2,0,0,1,1]
mi = mutual_info_score(data1,data2)
print(mi)
```

```
0.5004024235381879
```

Distance Correlation

- Distance correlation is a measure of dependence between two paired random vectors of arbitrary dimension. The population distance correlation coefficient is zero if and only if the random vectors are independent. Thus, distance correlation measures both linear and nonlinear association between two random variables or random vectors.

$$\mathrm{dCor}(X,Y) = \frac{\mathrm{dCov}(X,Y)}{\sqrt{\mathrm{dVar}(X)\ \mathrm{dVar}(Y)}}$$

Model Based Ranking

- Build prediction model based on each individual feature and its corresponding response variable.
- Python sklearn:

    RandonForestRegressor

Example:

[('0.620', 'LSTAT'), ('0.591', 'RM'), ('0.467', 'NOX'), ('0.342', 'INDUS'), ('0.305', 'TAX'), ('0.240', 'PTRATIO'), ('0.206', 'CRIM'), ('0.187', 'RAD'), ('0.184', 'ZN'), ('0.135', 'B'), ('0.082', 'DIS'), ('0.020', 'CHAS'), ('0.002', 'AGE')]

FEATURE SELECTION METHOD

- Filter
  - Removing features with low variance
  - Univariate feature selection
    - Chi2 (Y Discrete)
    - Pearson Correlation (Y Continuous)
    - Mutual Information (Y Discrete) and Maximal Information Coefficient (Y Continuous)
    - Distance Correlation
    - Model Based Ranking
- **Wrapper**
  - **Recursive Feature Elimination**
- Embedded
  - Feature selection using SelectFromModel
    - L1-based Feature Selection
    - Randomized Sparse Models
    - Tree-based Feature Selection
  - Feature selection as part of a pipeline

Recursive Feature Elimination

- Recursive feature elimination is a feature selection method that fits a model and removes the weakest feature (or features) until the specified number of features is reached.
- If the number of features is d, the number of all subsets is $2^d - 1$ (including empty set). Use a machine learning method such as SVM to calculate validation error for all subsets. The selected feature is the subset which has the least validation error.
- Python sklearn

```
from sklearn.feature_selection import RFECV
rfecv.estimator_.feature_importances_
```

FEATURE SELECTION METHOD

- Filter
  - Removing features with low variance
  - Univariate feature selection
    - Chi2 (Y Discrete)
    - Pearson Correlation (Y Continuous)
    - Mutual Information (Y Discrete) and Maximal Information Coefficient (Y Continuous)
    - Distance Correlation
    - Model Based Ranking
- Wrapper
  - Recursive Feature Elimination
- **Embedded**
  - **Feature selection using SelectFromModel (python sklearn)**
    - **L1 Regularization**
    - **Randomized Sparse Models**
    - **Tree-based Feature Selection**
  - **Feature selection as part of a pipeline**

How to avoid overfitting?

- Input Data
  - Train with more data (clean data)
  - Data augmentation
- Feature Engineering:
  - Feature Extraction (Dimension Reduction)
  - Feature Selection
- **Model:**
  - **Regularization**
  - **Dropout Layers**
- Training Stage:
  - Cross-Validation
  - Early Stopping

Regularization

- Regularization applies a "penalty" to the input parameters with the larger coefficients, which subsequently limits the model's variance.
- If there is one attribute w, loss function is $L(w)$

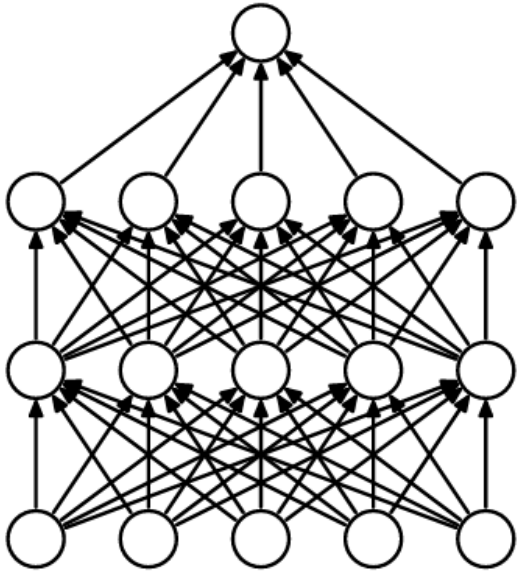
$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}, y) + \frac{\lambda}{m}\|w\|_1 \qquad J_{L1}(w) = L(w) + \lambda|w|$$

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}, y) + \frac{\lambda}{2m}\|w\|_2^2 \qquad J_{L2}(w) = L(w) + \lambda w^2$$

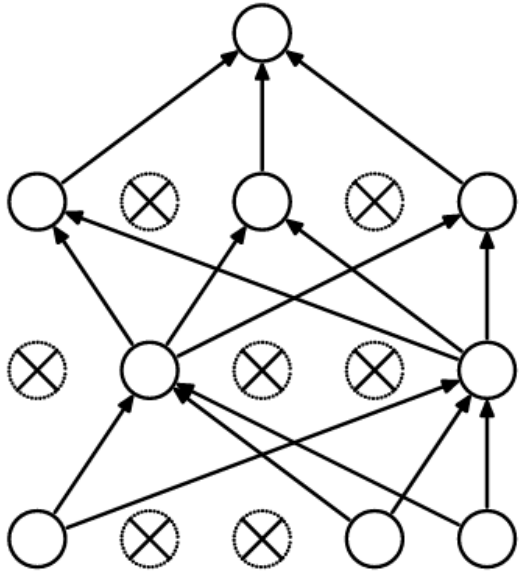
Question: Which regularization can do feature selection? L1 or L2? Why?

Dropout

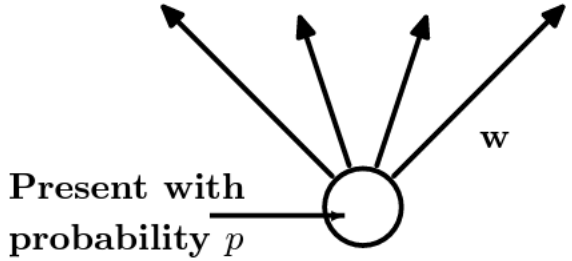
Dropout: A Simple Way to Prevent Neural Networks from Overfitting
Journal of Machine Learning Research 2014



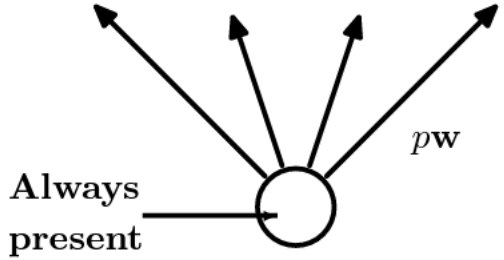
(a) Standard Neural Net



(b) After applying dropout.



(a) At training time



(b) At test time

How to avoid overfitting?

- Input Data
  - Train with more data (clean data)
  - Data augmentation
- Feature Engineering:
  - Feature Extraction (Dimension Reduction)
  - Feature Selection
- Model:
  - Regularization
  - Dropout Layers
- **Training Stage:**
  - **Cross-Validation**
  - **Early Stopping**

Validation set

- Splitting training set: A *fake* test set to tune hyper-parameters

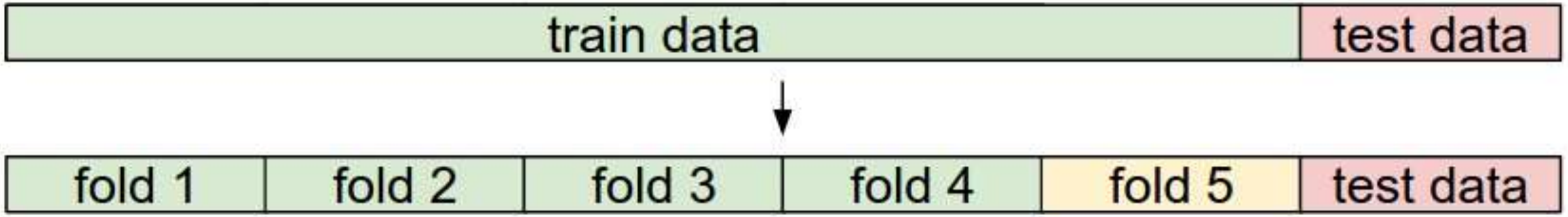
```
# assume we have Xtr_rows, Ytr, Xte_rows, Yte as before
# recall Xtr_rows is 50,000 x 3072 matrix
Xval_rows = Xtr_rows[:1000, :] # take first 1000 for validation
Yval = Ytr[:1000]
Xtr_rows = Xtr_rows[1000:, :] # keep last 49,000 for train
Ytr = Ytr[1000:]

# find hyperparameters that work best on the validation set
validation_accuracies = []
for k in [1, 3, 5, 10, 20, 50, 100]:

  # use a particular value of k and evaluation on validation data
  nn = NearestNeighbor()
  nn.train(Xtr_rows, Ytr)
  # here we assume a modified NearestNeighbor class that can take a k as input
  Yval_predict = nn.predict(Xval_rows, k = k)
  acc = np.mean(Yval_predict == Yval)
  print 'accuracy: %f' % (acc,)

  # keep track of what works on the validation set
  validation_accuracies.append((k, acc))
```

CROSS-VALIDATION

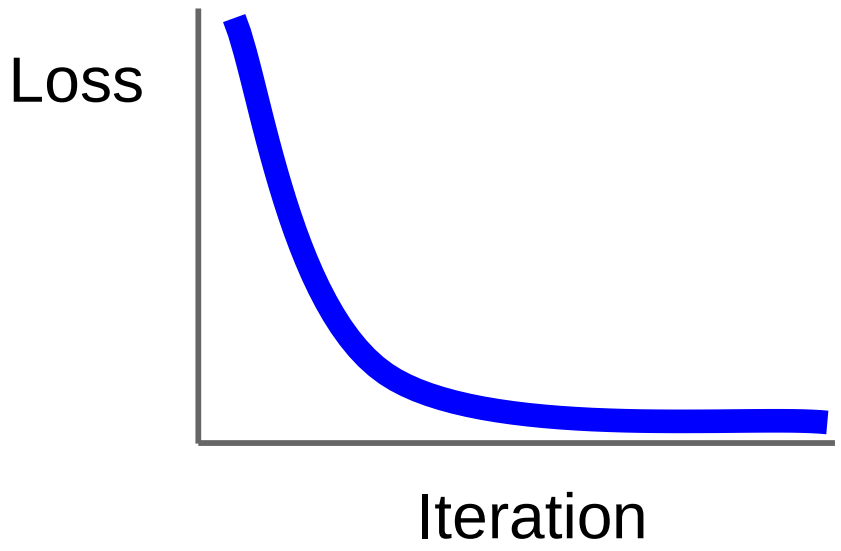
- A technique for evaluating ML models by training several ML models on subsets of the available input data and evaluating them on the complementary subset of the data.

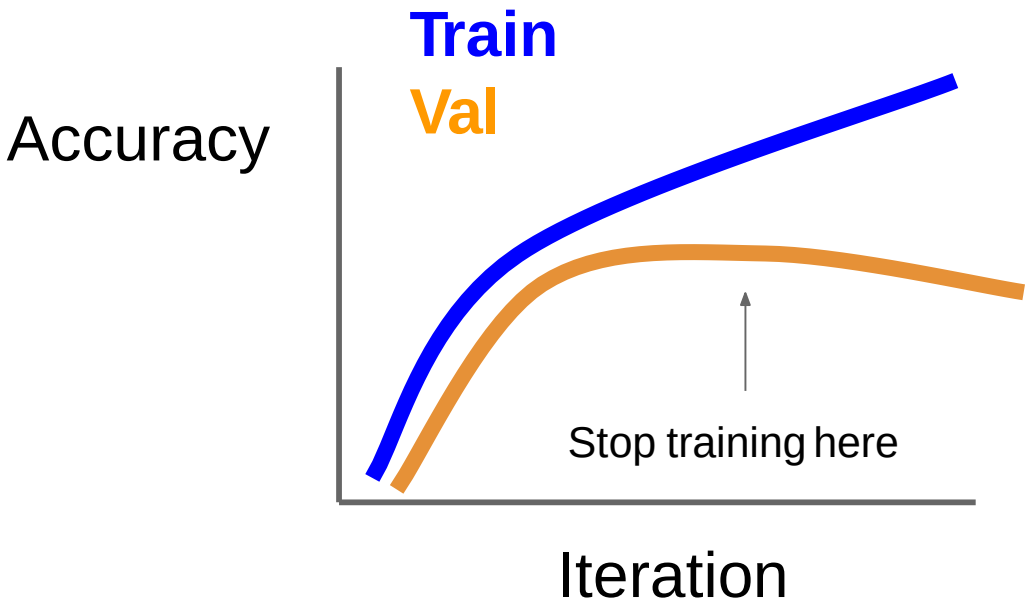
Cross-validation

- 5-fold cross-validation -> split the training data into 5 equal folds
- 4 of them for training and 1 for validation



Early Stopping: Always do this





Stop training the model when accuracy on the validation set decreases
Or train for a long time, but always keep track of the model snapshot that worked best on val

Conclusion

- Input Data
  - Train with more data (clean data)
  - Data augmentation
- Feature Engineering:
  - Feature Extraction (Dimension Reduction)
  - Feature Selection
- Model:
  - Regularization
  - Dropout Layers
- Training Stage:
  - Cross-Validation
  - Early Stopping