CS 6501: Deep Learning for Computer Graphics

Basics of Machine Learning

Connelly Barnes
Overview

• Supervised, unsupervised, and reinforcement learning
• Simple learning models
  • Clustering
  • Linear regression
  • Linear Support Vector Machines (SVM)
  • k-Nearest Neighbors
• Overfitting and generalization
• Training, testing, validation
• Balanced datasets
• Measuring performance of a classifier
Supervised, Unsupervised, Reinforcement

• 3 broad categories:
  
  • **Supervised learning**: computer presented with example inputs and desired outputs by a “teacher”, goal is to learn general rule that maps inputs to outputs.
  
  • **Unsupervised learning**: No output labels are given to the algorithm, leaving it on its own to find structure in the inputs.
  
  • **Reinforcement learning**: An agent determines what actions to best take in an environment to maximize some notion of cumulative reward.
What Kind of Learning is This?

- Learn given input image, whether it is truck or car? Training data:

  Label( ) = Truck

  Label( ) = Car

  Label( ) = Truck

  Label( ) = Car

  Label( ) = Truck

  Label( ) = Car

Images are Creative Commons, sources: [1], [2], [3], [4], [5], [6]
What Kind of Learning is This?

- We have a dataset of customers, each with 2 associated attributes ($x_1$ and $x_2$). We want to discover groups of similar customers.

What features could we use as inputs for a machine learning algorithm?
What Kind of Learning is This?

Outtakes

[Peng et al., Terrain-Adaptive Locomotion…, SIGGRAPH 2016]
Overview

- Supervised, unsupervised, and reinforcement learning
- Simple learning models
  - Clustering
    - Linear regression
    - Linear Support Vector Machines (SVM)
    - k-Nearest Neighbors
- Overfitting and generalization
- Training, testing, validation
- Balanced datasets
- Measuring performance of a classifier
Clustering

• Unsupervised learning
• Requires input data, but no labels
• Detects patterns, e.g.
  • Groups of similar emails, similar web-pages in search results
  • Similar customer shopping patterns
  • Regions of images

Slides adapted from David Sontag, Luke Zettlemoyer, Vibhav Gogate, Carlos Guestrin, Andrew Moore, Dan Klein
Clustering

• **Idea**: group together similar instances
• **Example**: 2D point patterns

Slides adapted from David Sontag, Luke Zettlemoyer, Vibhav Gogate, Carlos Guestrin, Andrew Moore, Dan Klein
Clustering

- **Idea**: group together similar instances
- **Example**: 2D point patterns

Slides adapted from David Sontag, Luke Zettlemoyer, Vibhav Gogate, Carlos Guestrin, Andrew Moore, Dan Klein
Clustering

• **Idea**: group together similar instances
• **Example**: 2D point patterns
Clustering

- **Idea**: group together similar instances
- **Problem**: How to define “similar”?  
- **Problem**: How many clusters?

Slides adapted from David Sontag, Luke Zettlemoyer, Vibhav Gogate, Carlos Guestrin, Andrew Moore, Dan Klein
Clustering

- **Similarity**: in Euclidean space $\mathbb{R}^n$, could be a distance function.
- For example: $D(x, y) = \|x - y\|_2^2$
- Clustering results will depend on measure of similarity / dissimilarity

Slides adapted from David Sontag, Luke Zettlemoyer, Vibhav Gogate, Carlos Guestrin, Andrew Moore, Dan Klein
Clustering Algorithms

- Partitioning algorithms (flat)
  - K-means

- Hierarchical algorithms
  - Bottom-up: agglomerative
  - Top-down: divisive

Slides adapted from David Sontag, Luke Zettlemoyer, Vibhav Gogate, Carlos Guestrin, Andrew Moore, Dan Klein
Clustering Examples: Image Segmentation

- Divide an image into regions that are perceptually similar to humans.

Slides adapted from James Hays, David Sontag, Luke Zettlemoyer, Vibhav, Gogate, Carlos Guestrin, Andrew Moore, Dan Klein
Clustering Examples: Biology

- Cluster species based on e.g. genetic or phenotype similarity.

A
Clustering: k-Means

- Iterative clustering algorithm based on partitioning (flat).

- **Initialize**: Pick $k$ random points as cluster centers.

- **Iterate until convergence**:
  - Assign each point based on the closest cluster center.
  - Update each cluster center based on the mean of the points assigned to it.
Clustering: k-Means

- Iterative clustering algorithm based on partitioning (flat).

- **Initialize**: Pick $k$ random points as cluster centers.

- **Iterate until convergence**:
  - Assign each point based on the closest cluster center.
  - Update each cluster center based on the mean of the points assigned to it.
Clustering: k-Means

- Iterative clustering algorithm based on partitioning (flat).

- **Initialize**: Pick $k$ random points as cluster centers.

- **Iterate until convergence**:
  - Assign each point based on the closest cluster center.
  - Update each cluster center based on the mean of the points assigned to it.

What color should this point be?
Clustering: k-Means

- Iterative clustering algorithm based on partitioning (flat).
  - **Initialize**: Pick $k$ random points as cluster centers.
  - **Iterate until convergence**:
    - Assign each point based on the closest cluster center.
    - Update each cluster center based on the mean of the points assigned to it.

What color should this point be?
Clustering: k-Means

- Iterative clustering algorithm based on partitioning (flat).

- **Initialize**: Pick $k$ random points as cluster centers.

- **Iterate until convergence**:
  - Assign each point based on the closest cluster center.
  - Update each cluster center based on the mean of the points assigned to it.

What color should this point be?
Clustering: k-Means

- Iterative clustering algorithm based on partitioning (flat).

- **Initialize**: Pick \( k \) random points as cluster centers.

- **Iterate until convergence**:
  - Assign each point based on the closest cluster center.
  - Update each cluster center based on the mean of the points assigned to it.

What color should this point be?
Clustering: k-Means

- Iterative clustering algorithm based on partitioning (flat).
- **Initialize**: Pick $k$ random points as cluster centers.
- **Iterate until convergence**:
  - Assign each point based on the closest cluster center.
  - Update each cluster center based on the mean of the points assigned to it.
Clustering: k-Means

- Iterative clustering algorithm based on partitioning (flat).

- **Initialize**: Pick \( k \) random points as cluster centers.

- **Iterate until convergence**:
  - Assign each point based on the closest cluster center.
  - Update each cluster center based on the mean of the points assigned to it.
Clustering: k-Means

• Iterative clustering algorithm based on partitioning (flat).

• **Initialize**: Pick $k$ random points as cluster centers.

• **Iterate until convergence**:
  
  • Assign each point based on the closest cluster center.
  
  • Update each cluster center based on the mean of the points assigned to it.
Clustering: k-Means

- Iterative clustering algorithm based on partitioning (flat).

- **Initialize**: Pick $k$ random points as cluster centers.

- **Iterate until convergence**:  
  - Assign each point based on the closest cluster center.
  - Update each cluster center based on the mean of the points assigned to it.
Clustering: k-Means

• Iterative clustering algorithm based on partitioning (flat).

• **Initialize**: Pick $k$ random points as cluster centers.

• **Iterate until convergence**:
  • Assign each point based on the closest cluster center.
  • Update each cluster center based on the mean of the points assigned to it.

Result of k-Means:
Clustering: k-Means

- Minimizes within-cluster sum of squares distance:

\[
\arg\min_s \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2
\]  \hspace{1cm} (1)

- Here \(\mu_i\) is the mean of the points belonging to cluster \(S_i\).
- No guarantee algorithm will converge to global minimum.
- Can run several times and take best result according to (1).
Overview

• Supervised, unsupervised, and reinforcement learning
• Simple learning models
  • Clustering
  • **Linear regression**
  • Linear Support Vector Machines (SVM)
  • k-Nearest Neighbors
• Overfitting and generalization
• Training, testing, validation
• Balanced datasets
Linear Regression

- Uses a linear model to model relationship between dependent variable $y \in \mathbb{R}$, and input (independent) variables $x_1, \ldots, x_n \in \mathbb{R}^n$
- Is this supervised or unsupervised learning?

![Graph showing a scatter plot with a linear regression line](image-url)
Linear Regression

- Uses a linear model to model relationship between dependent variable \( y \in \mathbb{R} \), and input (independent) variables \( x_1, \ldots, x_n \in \mathbb{R}^n \).

For each observation (data point) \( i = 1, \ldots, m \):

\[
y_i = \mathbf{w} \cdot \mathbf{x}_i + b = w_1 x_{i,1} + \cdots + w_n x_{i,n} + \cdots + b
\]

Here \( x_{i,j} \) is observation \( i \) of input variable \( j \).

Parameters of model: \( \mathbf{w}, b \).
Linear Regression

- Can simply the model by adding additional input that is always one:
  \[ x_{i,n+1} = 1 \quad i = 1, \ldots, m \]
- The corresponding parameter in \( w \) is called the **intercept**.

For each observation (data point) \( i = 1, \ldots, m \):

\[
\begin{align*}
y_i &= w \cdot x_i \\
    &= w_1 x_{i,1} + \cdots + w_{n+1} x_{i,n+1}
\end{align*}
\]

Parameters of model: \( w \).
Linear Least Squares Regression

- Define an objective function or loss function to optimize the model
- One loss function: least squares ("least squared error").

\[
E = \sum_{i=1}^{m} (w \cdot x_i - y_i)^2
\]

Parameters of model: \( w \).

- What is \( E \) for the 2D line fitting case at right? (blackboard)
- How to minimize \( E \)?
Linear Least Squares Regression

Set derivatives of objective function with respect to parameters equal to zero.

\[
\frac{\partial E}{\partial w_j} = \frac{\partial}{\partial w_j} \sum_{i=1}^{m} (w \cdot x_i - y_i)^2 = 0
\]

\[
2 \sum_{i=1}^{m} (w \cdot x_i - y_i)x_{ij} = 0
\]

Normal equations:

\[
\sum_{i=1}^{m} \left( \sum_{k=1}^{n} x_{ik} w_k - y_i \right) x_{ij} = 0
\]

\[
\sum_{i=1}^{m} \sum_{k=1}^{n} x_{ik} x_{ij} w_k = \sum_{i=1}^{m} x_{ij} y_i
\]
Linear Least Squares Regression

- Normal equations in matrix form:

\[ (X^TX)w = X^T y \]

\[ w = (X^TX)^{-1} X^T y \]

- $X$ is the matrix with $x_{ij}$ being observation $i$ of input variable $j$.
- $y$ is the vector of dependent variable (output) observations.
Linear Least Squares Example

• Suppose we have three observations (m=3) of one input variable $x_1$:

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.5</td>
<td>0.6</td>
</tr>
<tr>
<td>1</td>
<td>0.9</td>
</tr>
</tbody>
</table>
Linear Least Squares Example

• Suppose we have three observations (m=3) of one input variable $x_1$.
• Add additional constant variable $x_2$:

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0.5</td>
<td>1</td>
<td>0.6</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.9</td>
</tr>
</tbody>
</table>

• $X = \begin{bmatrix} 0 & 1 \\ 0.5 & 1 \\ 1 & 1 \end{bmatrix}$, $y = \begin{bmatrix} 0 \\ 0.6 \\ 0.9 \end{bmatrix}$, so $w = (X^T X)^{-1} X^T y = \begin{bmatrix} 0.9 \\ 0.05 \end{bmatrix}$
Linear Least Squares Example

\[ y = 0.9x + 0.05 \]
Overview

- Supervised, unsupervised, and reinforcement learning
- Simple learning models
  - Clustering
  - Linear regression
  - Linear Support Vector Machines (SVM)
  - k-Nearest Neighbors
- Overfitting and generalization
- Training, testing, validation
- Balanced datasets
- Measuring performance of a classifier
Linear Support Vector Machines (SVM)

• In linear regression, we had input variables $x_1, \ldots, x_n$ and we regressed them against a dependent variable $y \in \mathbb{R}$
• But what if we want to make a classifier?
• For example, a binary classifier could predict either $y = -1, y = 1$
• One simple option: use linear regression to find a linear model that best fits the data
• But this will not necessarily generalize well to new inputs.
Linear Support Vector Machines (SVM)

- Idea: if data are separable by a linear hyperplane, then maximize separation distance (margin) between points.
Linear Support Vector Machines (SVM)

- Two hyperplanes:
  \[ \mathbf{w} \cdot \mathbf{x} + b = 1 \]
  and
  \[ \mathbf{w} \cdot \mathbf{x} + b = -1. \]
- Distance between hyperplanes is:
  \[ \frac{2}{\| \mathbf{w} \|} \]
- So to maximize distance, minimize \( \| \mathbf{w} \| \)
Linear Support Vector Machines (SVM)

• For each point $i$, either:

$$\vec{w} \cdot \vec{x}_i + b \geq 1, \text{ if } y_i = 1$$

or

$$\vec{w} \cdot \vec{x}_i + b \leq -1, \text{ if } y_i = -1.$$

• This can be rewritten as, for each $i$:

$$y_i (\vec{w} \cdot \vec{x}_i + b) \geq 1$$
Linear Support Vector Machines (SVM)

- So our minimization problem becomes:

- Minimize $\|\vec{w}\|$ subject to the constraint:
  $$y_i (\vec{w} \cdot \vec{x}_i + b) \geq 1 \quad i = 1, \ldots, m$$

- Can be solved with quadratic programming

- Maximizes distance (margin) between two classes of data

From Wikipedia
Linear Support Vector Machines (SVM)

• If data are not linearly separable, can use a **soft margin classifier**, which has an objective function that sums for all data points $i$, a penalty of zero if the data point is correctly classified, otherwise, the distance to the margin.
How to Use Linear SVMs in Deep Learning?

• Linear SVMs tend to perform well with small amounts of training data
• Deep learning tends to perform well with large amounts of data

• What to do if we have a new problem with only a small dataset?
• One solution: use linear SVM
• Another solution: transfer learning.
  • Use a deep learning model trained on different problem
  • Train a linear SVM using features extracted from neural network.
Overview

- Supervised, unsupervised, and reinforcement learning
- Simple learning models
  - Clustering
  - Linear regression
  - Linear Support Vector Machines (SVM)
  - k-Nearest Neighbors
- Overfitting and generalization
- Training, testing, validation
- Balanced datasets
- Measuring performance of a classifier
k-Nearest Neighbors

• Suppose we can measure distance between input features.
• For example, Euclidean distance: \( D(x, y) = \|x - y\|_2^2 \)
• \( k \)-Nearest Neighbors simply uses the distance to the nearest \( k \) points to determine the classification or regression.
  • Classifier: take most common class within the \( k \) nearest points
  • Regression: take mean of \( k \) nearest points
• No parameters, so no need to “train” the algorithm
k-Nearest Neighbors Example, $k=3$
k-Nearest Neighbors Example, $k=5$
Overview

• Supervised, unsupervised, and reinforcement learning
• Simple learning models
  • Clustering
  • Linear regression
  • Linear Support Vector Machines (SVM)
  • k-Nearest Neighbors
• Overfitting and generalization
• Training, testing, validation
• Balanced datasets
• Measuring performance of a classifier
Overfitting and generalization

- Will this model have decent prediction for new inputs? (i.e. inputs similar to the training exemplars in blue)
Overfitting and generalization

• How about the model here, shown as the blue curve?
Overfitting and generalization

- **Overfitting**: the model describes random noise or errors instead of the underlying relationship.
Overfitting and generalization

• **Overfitting**: the model describes random noise or errors instead of the underlying relationship.

• Frequently occurs when model is overly complex (e.g. has too many parameters) relative to the number of observations.

• Has poor predictive performance.

From Wikipedia
Overfitting and generalization

- **Overfitting**: the model describes random noise or errors instead of the underlying relationship.
- Frequently occurs when model is overly complex (e.g. has too many parameters) relative to the number of observations.
- Has poor predictive performance.
Overfitting and generalization

• A rule of thumb for linear regression: one in ten rule
• One predictive variable can be studied for every ten events.

• In general, want number of data points >> number of parameters.
• But models with more parameters often perform better!
• One solution: gradually increase number of parameters in model until it starts to overfit, and then stop.
Overfitting Example with 2D Classifier

• From ConvnetJS Demo: 2D Classification with Neural Networks

23 data points, 17 parameters (5 neurons)
Overfitting Example with 2D Classifier

- From [ConvnetJS Demo: 2D Classification with Neural Networks](https://convnetjs.com/demos/2d-classification)

23 data points, 32 parameters (10 neurons)
Overfitting Example with 2D Classifier

• From ConvnetJS Demo: 2D Classification with Neural Networks

23 data points, 102 parameters (22 neurons total)
Generalization

- **Generalization error** is a measure of how accurately an algorithm is able to predict outcome values for previously unseen data.
- A model that is **overfit** will have poor generalization.
Overview

• Supervised, unsupervised, and reinforcement learning
• Simple learning models
  • Clustering
  • Linear regression
  • Linear Support Vector Machines (SVM)
  • k-Nearest Neighbors
• Overfitting and generalization
• Training, testing, validation
• Balanced datasets
• Measuring performance of a classifier
Training, testing, validation

• Break dataset into three parts by random sampling:
  • **Training dataset**: model is fit directly to this data
  • **Testing dataset**: model sees this data only once; used to measure the final performance of the classifier.
  • **Validation dataset**: model is repeatedly “tested” on this data during the training process to gain insight into overfitting.

• **Common percentages**:
  • Training (80%), testing (15%), validation (5%).
Training, testing, validation

- For neural networks, typically keep running training until validation error increases, then stop.
Cross-validation

- Repeatedly partition data into subsets: training and test.
- Take mean performance of classifier over all such partitions.
- **Leave one out**: train on n-1 samples, test on 1 sample.
  - Requires training n times.
- **k-fold cross-validation**: randomly partition data into k subsets (folds), at each iteration, train on k-1 folds, test on the other fold.
  - Requires training k times.
  - Common: 10-fold cross-validation
- Less common for deep learning (why?)
Overview

• Supervised, unsupervised, and reinforcement learning
• Simple learning models
  • Clustering
  • Linear regression
  • Linear Support Vector Machines (SVM)
  • k-Nearest Neighbors
• Overfitting and generalization
• Training, testing, validation
• **Balanced datasets**
• Measuring performance of a classifier
Balanced datasets

- **Unbalanced dataset:**
  - Suppose we have a binary classification problem (labels: 0, 1)
  - Suppose 99% of our observations are class 0.
  - We might learn the model “everything is zero.”
  - This model would be 99% accurate, but not model class 1 at all.

- **Balanced dataset:**
  - Equal numbers of observations of each class
Overview

• Supervised, unsupervised, and reinforcement learning
• Simple learning models
  • Clustering
  • Linear regression
  • Linear Support Vector Machines (SVM)
  • k-Nearest Neighbors
• Overfitting and generalization
• Training, testing, validation
• Balanced datasets
• Measuring performance of a classifier
Confusion Matrix

- If n classes, n x n matrix comparing actual versus predicted classes.

<table>
<thead>
<tr>
<th>Act Class</th>
<th>Class Predicted by Model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cat</td>
</tr>
<tr>
<td>Cat</td>
<td>9</td>
</tr>
<tr>
<td>Dog</td>
<td>4</td>
</tr>
</tbody>
</table>
Example: Handwritten Digit Recognition

Slide from Nelson Morgan at ICSI / Berkeley
Example: Handwritten Digit Recognition

- Visualization from MathWorks
Confusion Matrix

- For binary classifier, can call one class positive, the other negative.
- Should we call cats positive or negative?

<table>
<thead>
<tr>
<th>Actual Class</th>
<th>Class Predicted by Model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cat</td>
</tr>
<tr>
<td>Cat</td>
<td>9</td>
</tr>
<tr>
<td>Dog</td>
<td>4</td>
</tr>
</tbody>
</table>

Photo from [1]
Confusion Matrix

- For binary classifier, can call one class positive, the other negative.
- Cats are cuter, so cats = positive.

<table>
<thead>
<tr>
<th>Actual Class</th>
<th>Class Predicted by Model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Positive</td>
</tr>
<tr>
<td>Positive</td>
<td>9</td>
</tr>
<tr>
<td>Negative</td>
<td>4</td>
</tr>
</tbody>
</table>
Confusion Matrix

- For binary classifier, can call one class positive, the other negative.

<table>
<thead>
<tr>
<th>Actual Class</th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>True positive</td>
<td>1</td>
</tr>
<tr>
<td>Negative</td>
<td>4</td>
<td>True negative</td>
</tr>
</tbody>
</table>
Confusion Matrix

- For binary classifier, can call one class positive, the other negative.

<table>
<thead>
<tr>
<th>Actual Class</th>
<th>Class Predicted by Model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Positive</td>
</tr>
<tr>
<td>Positive</td>
<td>True positive</td>
</tr>
<tr>
<td>Negative</td>
<td>?</td>
</tr>
</tbody>
</table>
Confusion Matrix

• For binary classifier, can call one class positive, the other negative.
Classifier Performance

- **Accuracy**: \( \frac{(TP + TN)}{(Population\ Size)} \)
- **Precision**: \( \frac{TP}{(Predicted\ Positives)} = \frac{(TP + FP)}{(Population\ Size)} \)
- **Recall**: \( \frac{TP}{(Actual\ Positives)} = \frac{TP}{(TP + FN)} \)
  (also known as sensitivity, true positive rate)
- **Specificity**: \( \frac{TN}{(Actual\ Negatives)} = \frac{TN}{(TN + FP)} \)
  (also known as true negative rate)

<table>
<thead>
<tr>
<th>Actual Class</th>
<th>Class Predicted by Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>Positive\ true positive</td>
</tr>
<tr>
<td>Negative</td>
<td>False positive</td>
</tr>
</tbody>
</table>
Classifier Performance: Unbalanced Dataset

- **Accuracy**: \( \frac{TP + TN}{\text{Population Size}} \)
- **Precision**: \( \frac{TP}{\text{Predicted Positives}} = \frac{TP + FP}{TP} \)
- **Recall**: \( \frac{TP}{\text{Actual Positives}} = \frac{TP}{TP + FN} \) (also known as **sensitivity, true positive rate**)
- **Specificity**: \( \frac{TN}{\text{Actual Negatives}} = \frac{TN}{TN + FP} \) (also known as **true negative rate**)

<table>
<thead>
<tr>
<th>Class Predicted by Model</th>
<th>Actual Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class Predicted by Model</td>
<td>Positive</td>
</tr>
<tr>
<td>Positive</td>
<td>TP: 99</td>
</tr>
<tr>
<td>Negative</td>
<td>FP: 1</td>
</tr>
</tbody>
</table>
Classifier Performance: Unbalanced Dataset

- **Accuracy**: \( \frac{TP + TN}{\text{Population Size}} \) = ?
- **Precision**: \( \frac{TP}{\text{Predicted Positives}} = \frac{TP + FP}{TP + FN} \)
- **Recall**: \( \frac{TP}{\text{Actual Positives}} = \frac{TP}{TP + FN} \) 
  (also known as **sensitivity**, **true positive rate**)
- **Specificity**: \( \frac{TN}{\text{Actual Negatives}} = \frac{TN}{TN + FP} \) 
  (also known as **true negative rate**)

<table>
<thead>
<tr>
<th>Actual Class</th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>TP: 99</td>
<td>FN: 0</td>
</tr>
<tr>
<td>Negative</td>
<td>FP: 1</td>
<td>TN: 0</td>
</tr>
</tbody>
</table>
### Classifier Performance: Unbalanced Dataset

- **Accuracy**: \( \frac{TP + TN}{\text{Population Size}} = 99\% \)
- **Precision**: \( \frac{TP}{\text{Predicted Positives}} = \frac{TP + FP}{\text{Predicted Positives}} = \) ?
- **Recall**: \( \frac{TP}{\text{Actual Positives}} = \frac{TP}{TP + FN} \)
  (also known as **sensitivity, true positive rate**)
- **Specificity**: \( \frac{TN}{\text{Actual Negatives}} = \frac{TN}{TN + FP} \)
  (also known as **true negative rate**)

<table>
<thead>
<tr>
<th>Actual Class</th>
<th>Class Predicted by Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>TP: 99</td>
</tr>
<tr>
<td>Negative</td>
<td>FN: 0</td>
</tr>
<tr>
<td>Positive</td>
<td>FP: 1</td>
</tr>
<tr>
<td>Negative</td>
<td>TN: 0</td>
</tr>
</tbody>
</table>
Classifier Performance: Unbalanced Dataset

- **Accuracy**: \( \frac{TP + TN}{\text{Population Size}} = 99\% \)
- **Precision**: \( \frac{TP}{\text{Predicted Positives}} = \frac{TP + FP}{TP} = 99\% \)
- **Recall**: \( \frac{TP}{\text{Actual Positives}} = \frac{TP}{TP + FN} \)
  (also known as **sensitivity, true positive rate**)
- **Specificity**: \( \frac{TN}{\text{Actual Negatives}} = \frac{TN}{TN + FP} \)
  (also known as **true negative rate**)

<table>
<thead>
<tr>
<th>Actual Class</th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>TP: 99</td>
<td>FN: 0</td>
</tr>
<tr>
<td>Negative</td>
<td>FP: 1</td>
<td>TN: 0</td>
</tr>
</tbody>
</table>
Classifier Performance: Unbalanced Dataset

- **Accuracy**: \( \frac{(TP + TN)}{(Population\ Size)} = 99\% \)
- **Precision**: \( \frac{TP}{(Predicted\ Positives)} = \frac{(TP + FP)}{99\%} \)
- **Recall**: \( \frac{TP}{(Actual\ Positives)} = \frac{TP}{(TP + FN)} = ? \)
  (also known as **sensitivity**, **true positive rate**)
- **Specificity**: \( \frac{TN}{(Actual\ Negatives)} = \frac{TN}{(TN + FP)} \)
  (also known as **true negative rate**)

### Confusion Matrix

<table>
<thead>
<tr>
<th>Actual Class</th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>TP: 99</td>
<td>FN: 0</td>
</tr>
<tr>
<td>Negative</td>
<td>FP: 1</td>
<td>TN: 0</td>
</tr>
</tbody>
</table>
Classifier Performance: Unbalanced Dataset

- **Accuracy**: \( \frac{TP + TN}{\text{Population Size}} = 99\% \)
- **Precision**: \( \frac{TP}{\text{Predicted Positives}} = \frac{TP + FP}{TP + FP} = 99\% \)
- **Recall**: \( \frac{TP}{\text{Actual Positives}} = \frac{TP}{TP + FN} = 100\% \)
  (also known as *sensitivity, true positive rate*)
- **Specificity**: \( \frac{TN}{\text{Actual Negatives}} = \frac{TN}{TN + FP} \)
  (also known as *true negative rate*)

<table>
<thead>
<tr>
<th>Actual Class</th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>TP: 99</td>
<td>FN: 0</td>
</tr>
<tr>
<td>Negative</td>
<td>FP: 1</td>
<td>TN: 0</td>
</tr>
</tbody>
</table>
Classifier Performance: Unbalanced Dataset

- **Accuracy**: \( (TP + TN) / (\text{Population Size}) = 99\% \)
- **Precision**: \( TP / (\text{Predicted Positives}) = (TP + FP) = 99\% \)
- **Recall**: \( TP / (\text{Actual Positives}) = TP / (TP + FN) = 100\% \) (also known as sensitivity, true positive rate)
- **Specificity**: \( TN / (\text{Actual Negatives}) = 0\% \) (also known as true negative rate)

<table>
<thead>
<tr>
<th>Actual Class</th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>TP: 99</td>
<td>FN: 0</td>
</tr>
<tr>
<td>Negative</td>
<td>FP: 1</td>
<td>TN: 0</td>
</tr>
</tbody>
</table>
Summary

• Supervised, unsupervised, and reinforcement learning
• Simple learning models
  • Clustering
  • Linear regression
  • Linear Support Vector Machines (SVM)
  • k-Nearest Neighbors
• Overfitting and generalization
• Training, testing, validation
• Balanced datasets
• Measuring performance of a classifier