Chapter 3: Introduction to Classification

Reza Rezazadegan

Sharif University of Technology

October 26, 2022

Classification

- Classification is a supervised task similar to regression with the difference that in the former, the target variable is discrete.

Classification

- Classification is a supervised task similar to regression with the difference that in the former, the target variable is discrete.
- For example: spam filtering (more generally, document classification), detecting objects and faces in images, medical diagnosis using medical images, speaker recognition, fraud detection in ebanking, sentiment analysis of texts, etc.

Classification

- Classification is a supervised task similar to regression with the difference that in the former, the target variable is discrete.
- For example: spam filtering (more generally, document classification), detecting objects and faces in images, medical diagnosis using medical images, speaker recognition, fraud detection in ebanking, sentiment analysis of texts, etc.
- The simplest classification problem is *binary classification* with only two classes. (As opposed to *multi-class classification*.)

Classification

- Classification is a supervised task similar to regression with the difference that in the former, the target variable is discrete.
- For example: spam filtering (more generally, document classification), detecting objects and faces in images, medical diagnosis using medical images, speaker recognition, fraud detection in ebanking, sentiment analysis of texts, etc.
- The simplest classification problem is *binary classification* with only two classes. (As opposed to *multi-class classification*.)
- The two classes in binary classification can be mapped to $\{0, 1\}$.

Classification

- Classification is a supervised task similar to regression with the difference that in the former, the target variable is discrete.
- For example: spam filtering (more generally, document classification), detecting objects and faces in images, medical diagnosis using medical images, speaker recognition, fraud detection in ebanking, sentiment analysis of texts, etc.
- The simplest classification problem is *binary classification* with only two classes. (As opposed to *multi-class classification*.)
- The two classes in binary classification can be mapped to $\{0, 1\}$.
- This means we have a set of feature-class pairs $\{(\mathbf{x_i}, y_i)\}$ with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{0, 1\}$

Classification

- Classification is a supervised task similar to regression with the difference that in the former, the target variable is discrete.
- For example: spam filtering (more generally, document classification), detecting objects and faces in images, medical diagnosis using medical images, speaker recognition, fraud detection in ebanking, sentiment analysis of texts, etc.
- The simplest classification problem is *binary classification* with only two classes. (As opposed to *multi-class classification*.)
- The two classes in binary classification can be mapped to $\{0, 1\}$.
- This means we have a set of feature-class pairs $\{(\mathbf{x_i}, y_i)\}$ with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{0, 1\}$ and we want to find a function f such that $f(\mathbf{x}_i) = y_i$.

Classification

- Classification is a supervised task similar to regression with the difference that in the former, the target variable is discrete.
- For example: spam filtering (more generally, document classification), detecting objects and faces in images, medical diagnosis using medical images, speaker recognition, fraud detection in ebanking, sentiment analysis of texts, etc.
- The simplest classification problem is *binary classification* with only two classes. (As opposed to *multi-class classification*.)
- The two classes in binary classification can be mapped to $\{0, 1\}$.
- This means we have a set of feature-class pairs $\{(\mathbf{x_i}, y_i)\}$ with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{0, 1\}$ and we want to find a function f such that $f(\mathbf{x}_i) = y_i$.
- Therefore we can use regression for classification, if we can keep the values of f between zero and one!

Classification

- Classification is a supervised task similar to regression with the difference that in the former, the target variable is discrete.
- For example: spam filtering (more generally, document classification), detecting objects and faces in images, medical diagnosis using medical images, speaker recognition, fraud detection in ebanking, sentiment analysis of texts, etc.
- The simplest classification problem is *binary classification* with only two classes. (As opposed to *multi-class classification*.)
- The two classes in binary classification can be mapped to $\{0, 1\}$.
- This means we have a set of feature-class pairs $\{(\mathbf{x_i}, y_i)\}$ with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{0, 1\}$ and we want to find a function f such that $f(\mathbf{x}_i) = y_i$.
- Therefore we can use regression for classification, if we can keep the values of f between zero and one!
- This is the subject of **Logistic Regression** which is actually a classification method!

Logistic Regression

- In Logistic Regression we model our classifier by a function of the form

$$f(\mathbf{x}) = \frac{1}{1 + t^{-(\mathbf{a}^t\mathbf{x}+b)}} = \sigma(\mathbf{a}^t\mathbf{x} + b) \tag{1}$$

Logistic Regression

- In Logistic Regression we model our classifier by a function of the form
$$f(\mathbf{x}) = \frac{1}{1 + t^{-(\mathbf{a}^t\mathbf{x}+b)}} = \sigma(\mathbf{a}^t\mathbf{x} + b) \tag{1}$$

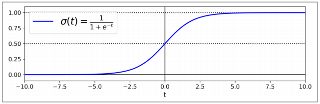
- The function $\sigma(t) = 1/(1 + e^{-t})$ is called the *logistic* or *sigmoid* function.

Logistic Regression

- In Logistic Regression we model our classifier by a function of the form

$$f(\mathbf{x}) = \frac{1}{1 + t^{-(\mathbf{a}^t\mathbf{x}+b)}} = \sigma(\mathbf{a}^t\mathbf{x} + b) \tag{1}$$

- The function $\sigma(t) = 1/(1 + e^{-t})$ is called the *logistic* or *sigmoid* function.

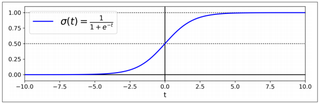


Logistic Regression

- In Logistic Regression we model our classifier by a function of the form

$$f(\mathbf{x}) = \frac{1}{1 + t^{-(\mathbf{a}^t\mathbf{x}+b)}} = \sigma(\mathbf{a}^t\mathbf{x} + b) \tag{1}$$

- The function $\sigma(t) = 1/(1 + e^{-t})$ is called the *logistic* or *sigmoid* function.

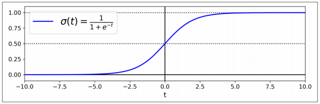


- The term b can be absorbed into $\mathbf{a}$ by adding an extra component equal to 1 to $\mathbf{x}$. We thus omit it from the notation.

Logistic Regression

- In Logistic Regression we model our classifier by a function of the form
$$f(\mathbf{x}) = \frac{1}{1 + t^{-(\mathbf{a}^t\mathbf{x}+b)}} = \sigma(\mathbf{a}^t\mathbf{x} + b) \tag{1}$$

- The function $\sigma(t) = 1/(1 + e^{-t})$ is called the *logistic* or *sigmoid* function.



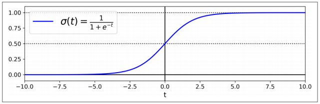
- The term b can be absorbed into **a** by adding an extra component equal to 1 to **x**. We thus omit it from the notation.
- Probabilistically:
$$p(y = 1|\mathbf{x}) = \sigma(\mathbf{a}^t\mathbf{x}) \tag{2}$$

Logistic Regression

- In Logistic Regression we model our classifier by a function of the form

$$f(\mathbf{x}) = \frac{1}{1 + t^{-(\mathbf{a}^t\mathbf{x}+b)}} = \sigma(\mathbf{a}^t\mathbf{x} + b) \tag{1}$$

- The function $\sigma(t) = 1/(1 + e^{-t})$ is called the *logistic* or *sigmoid* function.



- The term b can be absorbed into $\mathbf{a}$ by adding an extra component equal to 1 to $\mathbf{x}$. We thus omit it from the notation.
- Probabilistically:

$$p(y = 1|\mathbf{x}) = \sigma(\mathbf{a}^t\mathbf{x}) \tag{2}$$

Logistic Regression cont.

- Logistic Regression is an example of a *probabilistic model*. A probabilistic model not only tells us the predicted class or value but also its likelihood. (As opposed to *deterministic models*.)

Logistic Regression cont.

- Logistic Regression is an example of a *probabilistic model*. A probabilistic model not only tells us the predicted class or value but also its likelihood. (As opposed to *deterministic models*.)
- After training: $p(y = 1|\mathbf{x}) > 0.5 \implies \mathbf{x}$ belongs to class 1; however $p(y = 1|\mathbf{x})$ tells us how confident model is in its result.

Logistic Regression cont.

- Logistic Regression is an example of a *probabilistic model*. A probabilistic model not only tells us the predicted class or value but also its likelihood. (As opposed to *deterministic models*.)
- After training: $p(y = 1|\mathbf{x}) > 0.5 \implies \mathbf{x}$ belongs to class 1; however $p(y = 1|\mathbf{x})$ tells us how confident model is in its result.
- Advantages of probabilistic models include:
    - We would know if the probability of the predicted class is low (e.g. in multi-class classification).

Logistic Regression cont.

- Logistic Regression is an example of a *probabilistic model*. A probabilistic model not only tells us the predicted class or value but also its likelihood. (As opposed to *deterministic models*.)
- After training: $p(y = 1|\mathbf{x}) > 0.5 \implies \mathbf{x}$ belongs to class 1; however $p(y = 1|\mathbf{x})$ tells us how confident model is in its result.
- Advantages of probabilistic models include:
  - We would know if the probability of the predicted class is low (e.g. in multi-class classification).
  - We would know if there are two classes with similar probabilities e.g. if both $p(y = 1|\mathbf{x})$ and $p(y = 0|\mathbf{x})$ are close to 0.5.

Logistic Regression cont.

- Logistic Regression is an example of a *probabilistic model*. A probabilistic model not only tells us the predicted class or value but also its likelihood. (As opposed to *deterministic models*.)
- After training: $p(y = 1|\mathbf{x}) > 0.5 \implies \mathbf{x}$ belongs to class 1; however $p(y = 1|\mathbf{x})$ tells us how confident model is in its result.
- Advantages of probabilistic models include:
  - We would know if the probability of the predicted class is low (e.g. in multi-class classification).
  - We would know if there are two classes with similar probabilities e.g. if both $p(y = 1|\mathbf{x})$ and $p(y = 0|\mathbf{x})$ are close to 0.5.
  - Probabilistic models are more robust against noise in data.
- A probabilistic model that learns the conditional probability $p(y|\mathbf{x})$ is called a *discriminative model*.

Logistic Regression cont.

- Logistic Regression is an example of a *probabilistic model*. A probabilistic model not only tells us the predicted class or value but also its likelihood. (As opposed to *deterministic models*.)
- After training: $p(y = 1|\mathbf{x}) > 0.5 \implies \mathbf{x}$ belongs to class 1; however $p(y = 1|\mathbf{x})$ tells us how confident model is in its result.
- Advantages of probabilistic models include:
  - We would know if the probability of the predicted class is low (e.g. in multi-class classification).
  - We would know if there are two classes with similar probabilities e.g. if both $p(y = 1|\mathbf{x})$ and $p(y = 0|\mathbf{x})$ are close to 0.5.
  - Probabilistic models are more robust against noise in data.
- A probabilistic model that learns the conditional probability $p(y|\mathbf{x})$ is called a *discriminative model*.
- Later we see *generative models* which directly learn the joint probability distribution $p(\mathbf{x}, y)$.

Training Logistic Regression

- The loss function for Logistic Regression is given by

$$L(\mathbf{a}) = -\sum_{i=1}^{n} \log p(y = y_i | \mathbf{x}_i) = -\sum_{y_i=1} \log \sigma(\mathbf{x}) - \sum_{y_i=0} \log(1 - \sigma(\mathbf{x})) \tag{3}$$

- The loss function for Logistic Regression is given by

$$L(\mathbf{a}) = -\sum_{i=1}^{n} \log p(y = y_i|\mathbf{x}_i) = -\sum_{y_i=1} \log \sigma(\mathbf{x}) - \sum_{y_i=0} \log(1 - \sigma(\mathbf{x})) \tag{3}$$

$$\nabla_{\mathbf{a}} L(\mathbf{a}) = -\sum_{y_i=1} \frac{\nabla_{\mathbf{a}}\sigma(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \sum_{y_i=0} \frac{\nabla_{\mathbf{a}}\sigma(\mathbf{x}_i)}{1 - \sigma(\mathbf{x}_i)} \tag{4}$$

- $\nabla_{\mathbf{a}}\sigma(\mathbf{x}_i) = \mathbf{x}e^{-\mathbf{a}^t\mathbf{x}_i}\sigma(\mathbf{x}_i)^2$

Training Logistic Regression

- The loss function for Logistic Regression is given by

$$L(\mathbf{a}) = -\sum_{i=1}^{n} \log p(y = y_i|\mathbf{x}_i) = -\sum_{y_i=1} \log \sigma(\mathbf{x}) - \sum_{y_i=0} \log(1 - \sigma(\mathbf{x})) \tag{3}$$

$$\nabla_{\mathbf{a}} L(\mathbf{a}) = -\sum_{y_i=1} \frac{\nabla_{\mathbf{a}}\sigma(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \sum_{y_i=0} \frac{\nabla_{\mathbf{a}}\sigma(\mathbf{x}_i)}{1 - \sigma(\mathbf{x}_i)} \tag{4}$$

- $\nabla_{\mathbf{a}}\sigma(\mathbf{x}_i) = \mathbf{x}e^{-\mathbf{a}^t\mathbf{x}_i}\sigma(\mathbf{x}_i)^2$ thus $\nabla_{\mathbf{a}}\sigma(\mathbf{x}_i)/\sigma(\mathbf{x}_i) = \mathbf{x}_i(1 - \sigma(\mathbf{x}_i))$

Training Logistic Regression

- The loss function for Logistic Regression is given by

$$L(\mathbf{a}) = -\sum_{i=1}^{n} \log p(y = y_i|\mathbf{x}_i) = -\sum_{y_i=1} \log \sigma(\mathbf{x}) - \sum_{y_i=0} \log(1 - \sigma(\mathbf{x})) \tag{3}$$

$$\nabla_{\mathbf{a}} L(\mathbf{a}) = -\sum_{y_i=1} \frac{\nabla_{\mathbf{a}}\sigma(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \sum_{y_i=0} \frac{\nabla_{\mathbf{a}}\sigma(\mathbf{x}_i)}{1 - \sigma(\mathbf{x}_i)} \tag{4}$$

- $\nabla_{\mathbf{a}}\sigma(\mathbf{x}_i) = \mathbf{x}e^{-\mathbf{a}^t\mathbf{x}_i}\sigma(\mathbf{x}_i)^2$ thus $\nabla_{\mathbf{a}}\sigma(\mathbf{x}_i)/\sigma(\mathbf{x}_i) = \mathbf{x}_i(1 - \sigma(\mathbf{x}_i))$

$$\nabla_{\mathbf{a}} L(\mathbf{a}) = \sum_{i=1}^{n} (y_i - \sigma(\mathbf{x}_i))\mathbf{x}_i \tag{5}$$

- There is no know closed-form solution for $\nabla_{\mathbf{a}} L(\mathbf{a}) = 0$.

The importance of paying attention to model hypotheses and confidence levels

- Not paying attention to the hypothesis of the model we use, and its confidence in the prediction, can have consequences.

Subscribe | Latest Issues | SCIENTIFIC AMERICAN | Cart 0 | Sign In | Newsletters

Coronavirus | Health | Mind & Brain | Environment | Technology | Space & Physics | Video | Podcasts | Opinion | Store

ENVIRONMENT

Italian Scientists Sentenced to 6 Years for Earthquake Statements

A year-long trial about downplayed risks from a 2009 quake came to a close with the verdict, which alarmed Earth scientists worldwide

By Stephanie Pappas, LiveScience on October 22, 2012

- *George Box*: All models are wrong but some are useful.

The importance of paying attention to model hypotheses and confidence levels

- Not paying attention to the hypothesis of the model we use, and its confidence in the prediction, can have consequences.

- *George Box*: All models are wrong but some are useful.
- *Ronald Coase*: If you torture the data long enough, it will confess to anything.

The importance of paying attention to model hypotheses and confidence levels

- Not paying attention to the hypothesis of the model we use, and its confidence in the prediction, can have consequences.

Subscribe Latest Issues SCIENTIFIC AMERICAN. Cart 0 Sign In | Newsletters

Coronavirus Health Mind & Brain Environment Technology Space & Physics Video Podcasts Opinion Store

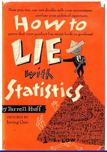
ENVIRONMENT

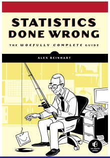
Italian Scientists Sentenced to 6 Years for Earthquake Statements

A year-long trial about downplayed risks from a 2009 quake came to a close with the verdict, which alarmed Earth scientists worldwide

By Stephanie Pappas, LiveScience on October 22, 2012

- *George Box*: All models are wrong but some are useful.
- *Ronald Coase*: If you torture the data long enough, it will confess to anything.
- Recommended reading:





Training Logistic Regression cont.

- However L is a convex function and we can find its (unique) minimum using the Method of Gradient Descent.

Training Logistic Regression cont.

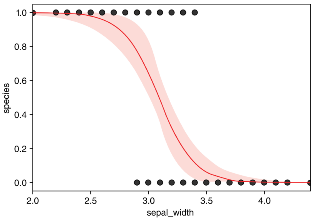
- However L is a convex function and we can find its (unique) minimum using the Method of Gradient Descent.
- In Python, Logistic Regression is provided by the class `sklearn.linear_model.LogisticRegression`

Training Logistic Regression cont.

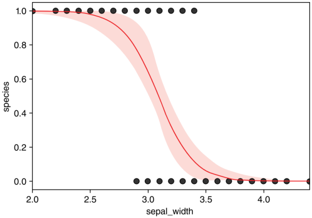
- However L is a convex function and we can find its (unique) minimum using the Method of Gradient Descent.
- In Python, Logistic Regression is provided by the class `sklearn.linear_model.LogisticRegression`
  It uses ℓ_2 regularization by default.

Training Logistic Regression cont.

- However L is a convex function and we can find its (unique) minimum using the Method of Gradient Descent.
- In Python, Logistic Regression is provided by the class `sklearn.linear_model.LogisticRegression`
  It uses ℓ_2 regularization by default.
- Logistic Regression for two species in the Iris dataset: we have one feature (sepal width) and two species of Iris as the two classes.



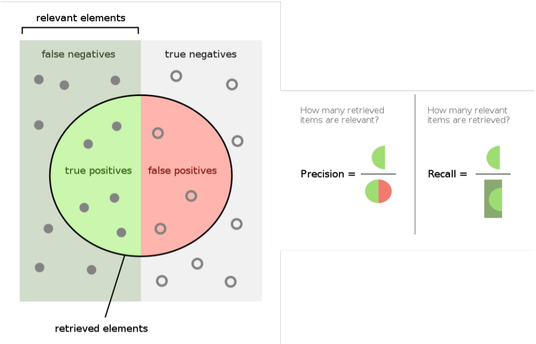
- However L is a convex function and we can find its (unique) minimum using the Method of Gradient Descent.
- In Python, Logistic Regression is provided by the class `sklearn.linear_model.LogisticRegression`
  It uses ℓ_2 regularization by default.
- Logistic Regression for two species in the Iris dataset: we have one feature (sepal width) and two species of Iris as the two classes.



- Exercise: show that the *decision boundary* $p(y=1|\mathbf{x}) = p(y=0|\mathbf{x})$ is given by the hypersurface $\mathbf{a}^t\mathbf{x} = a_0 + a_1x_1 + a_2x_2 + \cdots + a_dx_d = 0$.

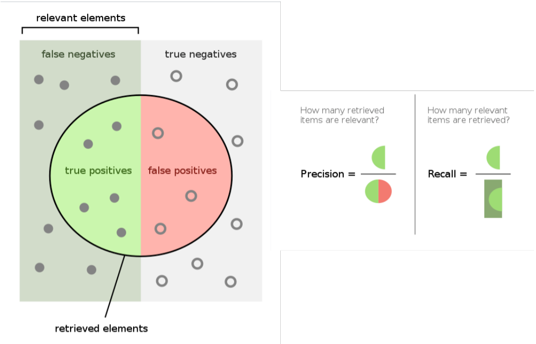
Measuring the performance of a binary classifier

- Confusion Matrix:
  Positive: belonging to the class 1, e.g. "sick".
  Negative: belonging to the class 0, e.g. "healthy".
  True: classified correctly by our model.
  Negative: classified falsely by our model.



Measuring the performance of a binary classifier

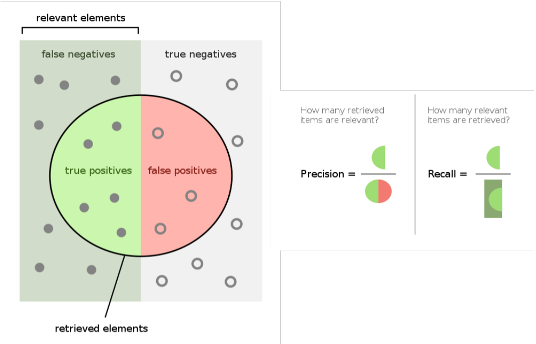
- Confusion Matrix:
  Positive: belonging to the class 1, e.g. "sick".
  Negative: belonging to the class 0, e.g. "healthy".
  True: classified correctly by our model.
  Negative: classified falsely by our model.



- Accuracy: $\frac{\textit{Correct Predictions}}{\textit{Dataset Size}}$

Measuring the performance of a binary classifier

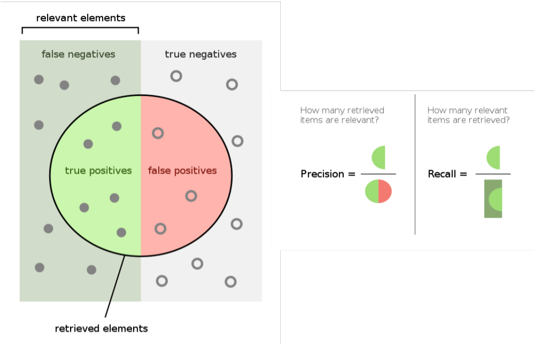
- Confusion Matrix:
  Positive: belonging to the class 1, e.g. "sick".
  Negative: belonging to the class 0, e.g. "healthy".
  True: classified correctly by our model.
  Negative: classified falsely by our model.



- Accuracy: $\frac{\textit{Correct Predictions}}{\textit{Dataset Size}} = \frac{TP+TN}{TP+TN+FP+FN}$

Measuring the performance of a binary classifier

- Confusion Matrix:
  Positive: belonging to the class 1, e.g. "sick".
  Negative: belonging to the class 0, e.g. "healthy".
  True: classified correctly by our model.
  Negative: classified falsely by our model.



- Accuracy: $\frac{Correct\ Predictions}{Dataset\ Size} = \frac{TP+TN}{TP+TN+FP+FN}$
- Accuracy is not a good measure if the two classes are imbalanced.

Precision and Recall

- Precision: $\frac{\textit{True Positives}}{\textit{Predicted Positives}}$

Precision and Recall

- Precision: $\frac{\textit{True Positives}}{\textit{Predicted Positives}} = \frac{TP}{TP+FP}$

Precision and Recall

- Precision: $\frac{\textit{True Positives}}{\textit{Predicted Positives}} = \frac{TP}{TP+FP}$
- Recall: $\frac{\textit{True Positives}}{\textit{All Positives}}$

Precision and Recall

- Precision: $\frac{\textit{True Positives}}{\textit{Predicted Positives}} = \frac{TP}{TP+FP}$
- Recall: $\frac{\textit{True Positives}}{\textit{All Positives}} = \frac{TP}{TP+FN}$

Precision and Recall

- Precision: $\frac{\textit{True Positives}}{\textit{Predicted Positives}} = \frac{TP}{TP+FP}$
- Recall: $\frac{\textit{True Positives}}{\textit{All Positives}} = \frac{TP}{TP+FN}$
- Precision=1 means no false positives.

Precision and Recall

- Precision: $\frac{\textit{True Positives}}{\textit{Predicted Positives}} = \frac{TP}{TP+FP}$
- Recall: $\frac{\textit{True Positives}}{\textit{All Positives}} = \frac{TP}{TP+FN}$
- Precision=1 means no false positives.
- Recall=1 means no false negatives.

Precision and Recall

- Precision: $\frac{\textit{True Positives}}{\textit{Predicted Positives}} = \frac{TP}{TP+FP}$
- Recall: $\frac{\textit{True Positives}}{\textit{All Positives}} = \frac{TP}{TP+FN}$
- Precision=1 means no false positives.
- Recall=1 means no false negatives.
- Recall is important in medical or security applications of ML, and also in content filtering where false negatives can be costly.

Precision-Recall Trade-off

- High Precision and low Recall means the predictor returns few positives but most of them are labeled correctly.

Precision-Recall Trade-off

- High Precision and low Recall means the predictor returns few positives but most of them are labeled correctly.
- Low Precision and high Recall means a lot of positives are returned but many of the predicted labels are incorrect.

Precision-Recall Trade-off

- High Precision and low Recall means the predictor returns few positives but most of them are labeled correctly.
- Low Precision and high Recall means a lot of positives are returned but many of the predicted labels are incorrect.
- Probabilistic classifiers use a threshold $p(y = 1|\mathbf{x}) \geq \epsilon$ to decide whether an instance $\mathbf{x}$ belongs to the "positive" class 1.

Precision-Recall Trade-off

- High Precision and low Recall means the predictor returns few positives but most of them are labeled correctly.
- Low Precision and high Recall means a lot of positives are returned but many of the predicted labels are incorrect.
- Probabilistic classifiers use a threshold $p(y = 1|\mathbf{x}) \geq \epsilon$ to decide whether an instance $\mathbf{x}$ belongs to the "positive" class 1.
- Threshold too high:

Precision-Recall Trade-off

- High Precision and low Recall means the predictor returns few positives but most of them are labeled correctly.
- Low Precision and high Recall means a lot of positives are returned but many of the predicted labels are incorrect.
- Probabilistic classifiers use a threshold $p(y = 1|\mathbf{x}) \geq \epsilon$ to decide whether an instance $\mathbf{x}$ belongs to the "positive" class 1.
- Threshold too high: few positive predictions which are mostly correct;

Precision-Recall Trade-off

- High Precision and low Recall means the predictor returns few positives but most of them are labeled correctly.
- Low Precision and high Recall means a lot of positives are returned but many of the predicted labels are incorrect.
- Probabilistic classifiers use a threshold $p(y = 1|\mathbf{x}) \geq \epsilon$ to decide whether an instance $\mathbf{x}$ belongs to the "positive" class 1.
- Threshold too high: few positive predictions which are mostly correct; which means high precision but low recall!

Precision-Recall Trade-off

- High Precision and low Recall means the predictor returns few positives but most of them are labeled correctly.
- Low Precision and high Recall means a lot of positives are returned but many of the predicted labels are incorrect.
- Probabilistic classifiers use a threshold $p(y = 1|\mathbf{x}) \geq \epsilon$ to decide whether an instance $\mathbf{x}$ belongs to the "positive" class 1.
- Threshold too high: few positive predictions which are mostly correct; which means high precision but low recall!
- Threshold too low:

Precision-Recall Trade-off

- High Precision and low Recall means the predictor returns few positives but most of them are labeled correctly.
- Low Precision and high Recall means a lot of positives are returned but many of the predicted labels are incorrect.
- Probabilistic classifiers use a threshold $p(y = 1|\mathbf{x}) \geq \epsilon$ to decide whether an instance $\mathbf{x}$ belongs to the "positive" class 1.
- Threshold too high: few positive predictions which are mostly correct; which means high precision but low recall!
- Threshold too low: a lot of positive predictions which contain a lot of false predictions;

Precision-Recall Trade-off

- High Precision and low Recall means the predictor returns few positives but most of them are labeled correctly.
- Low Precision and high Recall means a lot of positives are returned but many of the predicted labels are incorrect.
- Probabilistic classifiers use a threshold $p(y = 1|\mathbf{x}) \geq \epsilon$ to decide whether an instance $\mathbf{x}$ belongs to the "positive" class 1.
- Threshold too high: few positive predictions which are mostly correct; which means high precision but low recall!
- Threshold too low: a lot of positive predictions which contain a lot of false predictions; which means low precision but high recall!

Precision-Recall Trade-off

- High Precision and low Recall means the predictor returns few positives but most of them are labeled correctly.
- Low Precision and high Recall means a lot of positives are returned but many of the predicted labels are incorrect.
- Probabilistic classifiers use a threshold $p(y = 1|\mathbf{x}) \geq \epsilon$ to decide whether an instance $\mathbf{x}$ belongs to the "positive" class 1.
- Threshold too high: few positive predictions which are mostly correct; which means high precision but low recall!
- Threshold too low: a lot of positive predictions which contain a lot of false predictions; which means low precision but high recall!
- Stated differently: Recall measures how many of the positives are actually retrieved!

Precision-Recall Trade-off cont.

- The Precision-Recall Trade-off can be visualized by plotting (precision, recall) pairs for different values of the threshold.

Precision-Recall Trade-off cont.

- The Precision-Recall Trade-off can be visualized by plotting (precision, recall) pairs for different values of the threshold.

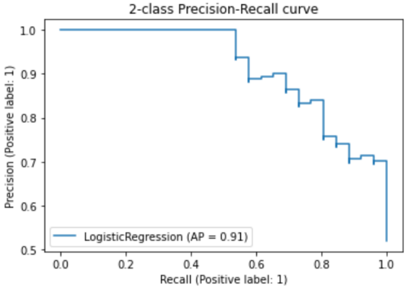


Figure: Precision-Recall curve for two species in the Iris dataset.

- The Precision-Recall Trade-off can be visualized by plotting (precision, recall) pairs for different values of the threshold.

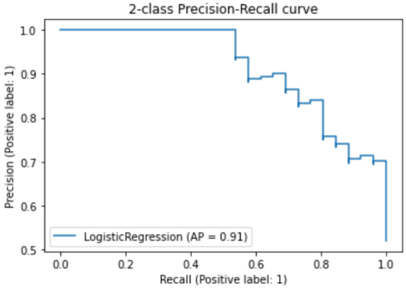


Figure: Precision-Recall curve for two species in the Iris dataset.

The higher the area under the Precision-Recall curve the better the classifier.

Multi-class Logistic Regression

- Imagine we have m different classes which we denote by $1, 2, \ldots, m$.

Multi-class Logistic Regression

- Imagine we have m different classes which we denote by $1, 2, \ldots, m$.
- We set

$$p(y = k|\mathbf{x}) = \frac{\exp(\mathbf{a}_k^t \mathbf{x})}{\sum_{i=1}^m \exp(\mathbf{a}_i^t \mathbf{x})} \tag{6}$$

Multi-class Logistic Regression

- Imagine we have m different classes which we denote by $1, 2, \ldots, m$.
- We set

$$p(y = k|\mathbf{x}) = \frac{\exp(\mathbf{a}_k^t \mathbf{x})}{\sum_{i=1}^m \exp(\mathbf{a}_i^t \mathbf{x})} \tag{6}$$

- The function $\sigma : \mathbb{R}^m \to (0,1)^m$ whose k'th component is given by the above $p(y = k|\mathbf{x})$ is called the **Softmax Function**.

Multi-class Logistic Regression

- Imagine we have m different classes which we denote by $1, 2, \ldots, m$.
- We set

$$p(y = k|\mathbf{x}) = \frac{\exp(\mathbf{a}_k^t \mathbf{x})}{\sum_{i=1}^m \exp(\mathbf{a}_i^t \mathbf{x})} \tag{6}$$

- The function $\sigma : \mathbb{R}^m \to (0, 1)^m$ whose k'th component is given by the above $p(y = k|\mathbf{x})$ is called the **Softmax Function**.
- The predicted class for $\mathbf{x}$ is the one that maximizes $p(y = k|\mathbf{x})$

Multi-class Logistic Regression

- Imagine we have m different classes which we denote by $1, 2, \ldots, m$.
- We set

$$p(y = k|\mathbf{x}) = \frac{\exp(\mathbf{a}_k^t \mathbf{x})}{\sum_{i=1}^{m} \exp(\mathbf{a}_i^t \mathbf{x})} \tag{6}$$

- The function $\sigma : \mathbb{R}^m \to (0, 1)^m$ whose k'th component is given by the above $p(y = k|\mathbf{x})$ is called the **Softmax Function**.
- The predicted class for $\mathbf{x}$ is the one that maximizes $p(y = k|\mathbf{x})$ or equivalently the one that maximizes $\mathbf{a}_k^t \mathbf{x}$.

Multi-class Logistic Regression

- Imagine we have m different classes which we denote by $1, 2, \ldots, m$.
- We set

$$p(y = k|\mathbf{x}) = \frac{\exp(\mathbf{a}_k^t \mathbf{x})}{\sum_{i=1}^{m} \exp(\mathbf{a}_i^t \mathbf{x})} \tag{6}$$

- The function $\sigma : \mathbb{R}^m \to (0,1)^m$ whose k'th component is given by the above $p(y = k|\mathbf{x})$ is called the **Softmax Function**.
- The predicted class for $\mathbf{x}$ is the one that maximizes $p(y = k|\mathbf{x})$ or equivalently the one that maximizes $\mathbf{a}_k^t \mathbf{x}$.
- Multi-class Logistic Regression is used when the classes are mutually exclusive.

Training Multi-class Logistic Regression: Cross Entropy

- To train Multi-class Logistic Regression we need a loss function which penalizes low predicted probability for the correct class:

Training Multi-class Logistic Regression: Cross Entropy

- To train Multi-class Logistic Regression we need a loss function which penalizes low predicted probability for the correct class:

$$L = -\sum_{i=1}^{N} \log(p(y = y_i|\mathbf{x}_i)) \tag{7}$$

Training Multi-class Logistic Regression: Cross Entropy

- To train Multi-class Logistic Regression we need a loss function which penalizes low predicted probability for the correct class:

$$L = -\sum_{i=1}^{N} \log(p(y = y_i|\mathbf{x}_i)) \tag{7}$$

- This loss function has an information theoretic interpretation as follows.

Training Multi-class Logistic Regression: Cross Entropy

- To train Multi-class Logistic Regression we need a loss function which penalizes low predicted probability for the correct class:

$$L = -\sum_{i=1}^{N} \log(p(y = y_i|\mathbf{x}_i)) \tag{7}$$

- This loss function has an information theoretic interpretation as follows.
- The **entropy** of a discrete probability distribution p is defined as

$$H(p) = -E(\log p) = -\sum_{i=1}^{m} p_i \log(p_i) \tag{8}$$

Training Multi-class Logistic Regression: Cross Entropy

- To train Multi-class Logistic Regression we need a loss function which penalizes low predicted probability for the correct class:

$$L = -\sum_{i=1}^{N} \log(p(y = y_i|\mathbf{x}_i)) \tag{7}$$

- This loss function has an information theoretic interpretation as follows.
- The **entropy** of a discrete probability distribution p is defined as

$$H(p) = -E(\log p) = -\sum_{i=1}^{m} p_i \log(p_i) \tag{8}$$

- Entropy is zero when $m = 1$ and is maximal when $p_i = 1/m$ for each i.

Training Multi-class Logistic Regression: Cross Entropy

- To train Multi-class Logistic Regression we need a loss function which penalizes low predicted probability for the correct class:

$$L = -\sum_{i=1}^{N} \log(p(y = y_i | \mathbf{x}_i)) \tag{7}$$

- This loss function has an information theoretic interpretation as follows.
- The **entropy** of a discrete probability distribution p is defined as

$$H(p) = -E(\log p) = -\sum_{i=1}^{m} p_i \log(p_i) \tag{8}$$

- Entropy is zero when $m = 1$ and is maximal when $p_i = 1/m$ for each i. Thus entropy is a measure of a "mixedness" of a distribution.

Training Multi-class Logistic Regression: Cross Entropy

- To train Multi-class Logistic Regression we need a loss function which penalizes low predicted probability for the correct class:

$$L = -\sum_{i=1}^{N} \log(p(y = y_i|\mathbf{x}_i)) \tag{7}$$

- This loss function has an information theoretic interpretation as follows.
- The **entropy** of a discrete probability distribution p is defined as

$$H(p) = -E(\log p) = -\sum_{i=1}^{m} p_i \log(p_i) \tag{8}$$

- Entropy is zero when $m = 1$ and is maximal when $p_i = 1/m$ for each i. Thus entropy is a measure of a "mixedness" of a distribution.
- **Cross Entropy** of two distributions q, p equals $-E_q[\log p]$ which means we have used a "wrong" distribution p to sample our data.

Multi-class Logistic Regression cont.

- Cross Entropy of q and p:

$$H(q, p) = -\sum_i q_i \log p_i \tag{9}$$

Multi-class Logistic Regression cont.

- Cross Entropy of q and p:

$$H(q, p) = -\sum_i q_i \log p_i \tag{9}$$

- In the case of Logistic Regression $q(y = k|\mathbf{x}_i) = \delta_{k,y_i}$.

Multi-class Logistic Regression cont.

- Cross Entropy of q and p:

$$H(q,p) = -\sum_i q_i \log p_i \tag{9}$$

- In the case of Logistic Regression $q(y=k|\mathbf{x}_i) = \delta_{k,y_i}$.
- Cross Entropy is a measure of difference between two probability distributions and is used as a loss function for training classification problems, e.g. in Deep Learning.

Multi-class Logistic Regression cont.

- Cross Entropy of q and p:

$$H(q,p) = -\sum_i q_i \log p_i \tag{9}$$

- In the case of Logistic Regression $q(y=k|\mathbf{x}_i) = \delta_{k,y_i}$.
- Cross Entropy is a measure of difference between two probability distributions and is used as a loss function for training classification problems, e.g. in Deep Learning.
- Minimizing the Cross Entropy loss function

Multi-class Logistic Regression cont.

- Cross Entropy of q and p:

$$H(q, p) = -\sum_i q_i \log p_i \tag{9}$$

- In the case of Logistic Regression $q(y = k|\mathbf{x}_i) = \delta_{k,y_i}$.
- Cross Entropy is a measure of difference between two probability distributions and is used as a loss function for training classification problems, e.g. in Deep Learning.
- Minimizing the Cross Entropy loss function is done using the method of Gradient Descent.

- Logistic Regression for the 3 classes in the Iris dataset, using two features:

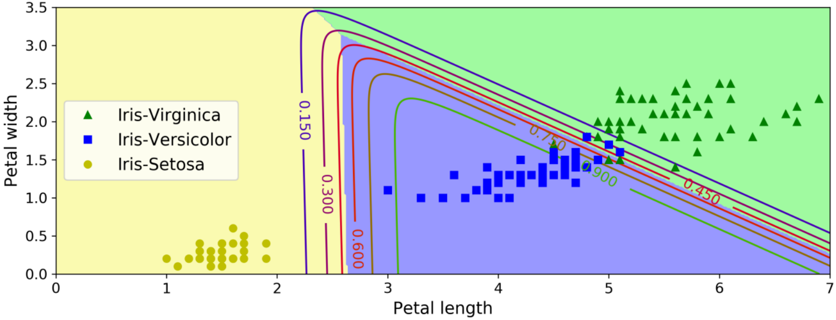


Figure: Credit: A. Geron, Hands-on Machine Learning

Multi-class Logistic Regression cont.

- Logistic Regression for the 3 classes in the Iris dataset, using two features:

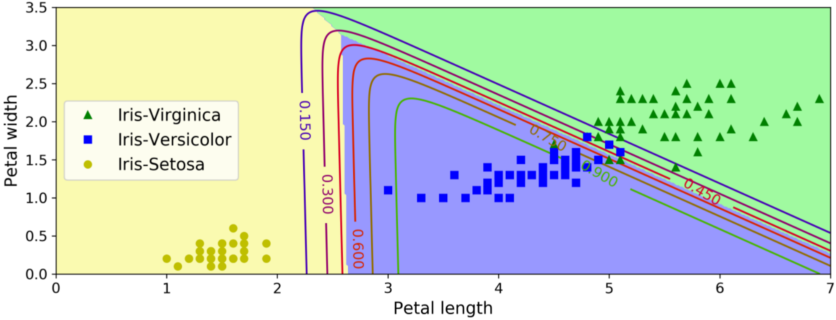


Figure: Credit: A. Geron, Hands-on Machine Learning

- Exercise: show that the decision boundaries $p(y = k|\mathbf{x}) = p(y = l|\mathbf{x}) = 0.5$ between any two classes in Multi-class Logistic Regression are line segments.

K-Nearest Neighbors (KNN) classifier

- Imagine we have a training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ where $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in 1, 2, \ldots, m$.

K-Nearest Neighbors (KNN) classifier

- Imagine we have a training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ where $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in 1, 2, \ldots, m$.
- In the KNN method, the class of a new instance $\mathbf{x}$ is determined by the most common class among the k nearest neighbors of $\mathbf{x}$ where k is a hyperparameter.

K-Nearest Neighbors (KNN) classifier

- Imagine we have a training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ where $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in 1, 2, \ldots, m$.
- In the KNN method, the class of a new instance $\mathbf{x}$ is determined by the most common class among the k nearest neighbors of $\mathbf{x}$ where k is a hyperparameter.
- Distance is measured in Euclidean distance (for continuous data) or edit (Hamming) distance for discrete data.

K-Nearest Neighbors (KNN) classifier

- Imagine we have a training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ where $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in 1, 2, \ldots, m$.
- In the KNN method, the class of a new instance $\mathbf{x}$ is determined by the most common class among the k nearest neighbors of $\mathbf{x}$ where k is a hyperparameter.
- Distance is measured in Euclidean distance (for continuous data) or edit (Hamming) distance for discrete data.
- KNN is very sensitive to the noise in data; specially for small k.

K-Nearest Neighbors (KNN) classifier

- Imagine we have a training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ where $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in 1, 2, \ldots, m$.
- In the KNN method, the class of a new instance $\mathbf{x}$ is determined by the most common class among the k nearest neighbors of $\mathbf{x}$ where k is a hyperparameter.
- Distance is measured in Euclidean distance (for continuous data) or edit (Hamming) distance for discrete data.
- KNN is very sensitive to the noise in data; specially for small k.
- KNN (like other distance-based methods) does not perform well if the features have different scales.

K-Nearest Neighbors (KNN) classifier

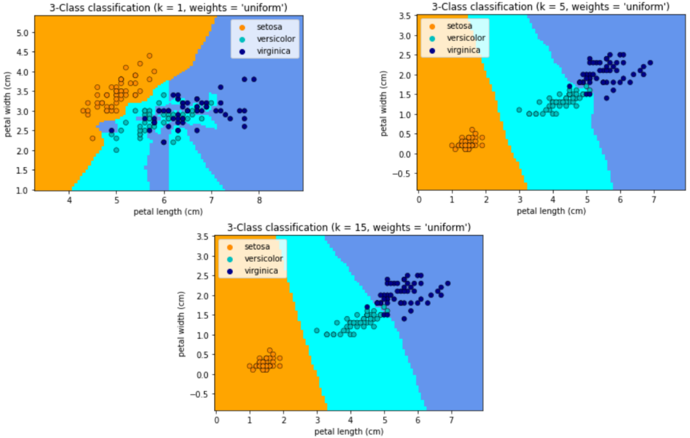
- Imagine we have a training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ where $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in 1, 2, \ldots, m$.
- In the KNN method, the class of a new instance $\mathbf{x}$ is determined by the most common class among the k nearest neighbors of $\mathbf{x}$ where k is a hyperparameter.
- Distance is measured in Euclidean distance (for continuous data) or edit (Hamming) distance for discrete data.
- KNN is very sensitive to the noise in data; specially for small k.
- KNN (like other distance-based methods) does not perform well if the features have different scales. The remedy is to scale the features e.g. using Scikit's `StandardScaler` method.

K-Nearest Neighbors (KNN) classifier

- Imagine we have a training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ where $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in 1, 2, \ldots, m$.
- In the KNN method, the class of a new instance $\mathbf{x}$ is determined by the most common class among the k nearest neighbors of $\mathbf{x}$ where k is a hyperparameter.
- Distance is measured in Euclidean distance (for continuous data) or edit (Hamming) distance for discrete data.
- KNN is very sensitive to the noise in data; specially for small k.
- KNN (like other distance-based methods) does not perform well if the features have different scales. The remedy is to scale the features e.g. using Scikit's `StandardScaler` method.
- In weighted KNN, the "vote" of each nearest neighbor is weighted by e.g. the inverse of its distance.

K-Nearest Neighbors (KNN) classifier

- Imagine we have a training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ where $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in 1, 2, \ldots, m$.
- In the KNN method, the class of a new instance $\mathbf{x}$ is determined by the most common class among the k nearest neighbors of $\mathbf{x}$ where k is a hyperparameter.
- Distance is measured in Euclidean distance (for continuous data) or edit (Hamming) distance for discrete data.
- KNN is very sensitive to the noise in data; specially for small k.
- KNN (like other distance-based methods) does not perform well if the features have different scales. The remedy is to scale the features e.g. using Scikit's `StandardScaler` method.
- In weighted KNN, the "vote" of each nearest neighbor is weighted by e.g. the inverse of its distance.
- KNN can be used as a Regression method too: $f(\mathbf{x}) = \sum_{i=1}^k f(\mathbf{x}_{n_i})$ where the $\mathbf{x}_{n_i}$ are the nearest neighbors of $\mathbf{x}$.


3-Class classification (k = 1, weights = 'uniform')
3-Class classification (k = 5, weights = 'uniform')
3-Class classification (k = 15, weights = 'uniform')
setosa
versicolor
virginica
petal width (cm)
petal length (cm)

Multi-label Classification

- In *Multi-label Classification*, each instance can belong to more than one class.

Multi-label Classification

- In *Multi-label Classification*, each instance can belong to more than one class.
- Examples include detecting faces in photos.

Multi-label Classification

- In *Multi-label Classification*, each instance can belong to more than one class.
- Examples include detecting faces in photos.
- KNN can be used for multi-label classification.