Algorithmic Data Analysis

Esther Galbrun
Autumn 2020
Part II

Classification
Different paradigms
A typical supervised binary classification problem

supervised labelled training instances are available
Learning paradigms

Supervised learning  labelled training instances
  → Classification
Learning paradigms

**Supervised learning**  labelled training instances
  → **Classification**

**Unsupervised learning**  unlabelled training instances
  → **Clustering**
Learning paradigms

**Supervised learning**  labelled training instances
  → *Classification*

**Unsupervised learning**  unlabelled training instances
  → *Clustering*

**Reinforcement learning**  choose actions to maximize cumulative rewards
  → *Exploration–exploitation trade-off*
Learning paradigms

Supervised learning  labelled training instances

Unsupervised learning  unlabelled training instances

Semi-supervised learning  few labelled + mostly unlabelled
Learning paradigms

**Supervised learning** labelled training instances

**Semi-supervised learning** few labelled + mostly unlabelled

**Active learning** query labels selectively, at a cost

**Online learning** data arrives and is processed iteratively

**Transfer learning** reuse what has been learnt on one task on a different task

→ *Classification*
Learning paradigms

Supervised learning  labelled training instances

Semi-supervised learning  few labelled + mostly unlabelled
Active learning  query labels selectively, at a cost
Online learning  data arrives and is processed iteratively
Transfer learning  reuse what has been learnt on one task
on a different task

→ Classification
Semi-supervised learning
Semi-supervised learning

Labelled data is difficult and expensive to acquire.
Unlabelled data is easier to obtain and can be useful too.

![Graph showing sepal length vs. sepal width with data points marked.](image-url)
Semi-supervised learning

Labelled data is difficult and expensive to acquire
Unlabelled data is easier to obtain and can be useful too
Semi-supervised learning

Labelled data is difficult and expensive to acquire. Unlabelled data is easier to obtain and can be useful too.
Semi-supervised learning

Labelled data is difficult and expensive to acquire
Unlabelled data is easier to obtain and can be useful too

![Scatter plot of sepal length vs sepal width with labeled data points highlighted]
The aim of **semi-supervised learning** is to exploit both labelled and unlabelled data to improve learning.
Semi-supervised learning

Key assumption of semi-supervised learning: class variables vary smoothly over dense regions of the space and more significantly over sparse regions.
**Induction vs. transduction**

**Induction** reasoning from specific cases to general rule

**Transduction** reasoning from specific cases to specific cases

\[ \text{Induction} \quad \text{reasoning from specific cases to general rule} \]

\[ \text{Transduction} \quad \text{reasoning from specific cases to specific cases} \]
Inductive algorithms proceed in two well-separated phases

**Training** learn a general rule from training instances  
**Testing** apply the general rule to test instances

Transductive algorithms use test instances for training

- require test instances to be specified at training time
- use information from test instances as unlabelled data points during training
- might not allow prediction on out-of-sample instances
Transductive Support Vector Machines

**Inductive SVM**

Find a separating hyperplane with maximum margin

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \|w\|^2 + C \sum_{j=1}^{n} \xi_j \\
\text{s.t.} & \quad y_j (w \cdot x^{(j)} + b) \geq 1 - \xi_j \text{ and } 0 \leq \xi_j \forall j
\end{align*}
\]
Transductive Support Vector Machines

Transductive SVM

Find a separating hyperplane with maximum margin
Label unsupervised examples to maximize the margin

\[
\text{minimize } \frac{1}{2} \|w\|^2 + C \sum_{j=1}^{j=n} \xi_j \\
\text{s.t. } y_j(w \cdot x^{(j)} + b) \geq 1 - \xi_j \text{ and } 0 \leq \xi_j \forall j \in \mathcal{I}_L \\
z_j(w \cdot x^{(j)} + b) \geq 1 - \xi_j \text{ and } 0 \leq \xi_j \forall j \in \mathcal{I}_U
\]

\(\mathcal{I}_L\) and \(\mathcal{I}_U\) index labelled and unlabelled examples respectively

\(y_j\) are known, class labels of the supervised examples
\(z_j\) are unknown, binary integer variables to be optimized
Transductive Support Vector Machines

minimize \( \frac{1}{2} \|w\|^2 + C \sum_{j=1}^{j=n} \xi_j \)

s.t. \( y_j(w \cdot x^{(j)} + b) \geq 1 - \xi_j \) and \( 0 \leq \xi_j \ \forall j \in \mathcal{I}_L \)

\( z_j(w \cdot x^{(j)} + b) \geq 1 - \xi_j \) and \( 0 \leq \xi_j \ \forall j \in \mathcal{I}_U \)

\( \mathcal{I}_L \) and \( \mathcal{I}_U \) index labelled and unlabelled examples respectively

\( y_j \) are known, class labels of the supervised examples

\( z_j \) are unknown, binary integer variables to be optimized

The modified optimization formulation is an integer program far more difficult than original convex optimization problem
Transductive Support Vector Machines

\[
\text{minimize} \quad \frac{1}{2} \|w\|^2 + C \sum_{j=1}^{n} \xi_j
\]

\[
\text{s.t.} \quad y_j(w \cdot x^{(j)} + b) \geq 1 - \xi_j \text{ and } 0 \leq \xi_j \forall j \in \mathcal{I}_L
\]

\[
z_j(w \cdot x^{(j)} + b) \geq 1 - \xi_j \text{ and } 0 \leq \xi_j \forall j \in \mathcal{I}_U
\]

\(\mathcal{I}_L\) and \(\mathcal{I}_U\) index labelled and unlabelled examples respectively

\(y_j\) are known, class labels of the supervised examples

\(z_j\) are unknown, binary integer variables to be optimized

Solving requires different techniques such as iterative methods labelling most confidently predicted examples
Semi-supervised learning

Key assumption of semi-supervised learning: class variables vary smoothly over dense regions of the space and more significantly over sparse regions.
Semi-supervised learning

Key assumption of **semi-supervised learning**: class variables vary smoothly over dense regions of the space and more significantly over sparse regions i.e. classes correspond to clusters.

![Diagram of sepal length vs sepal width](image)
Semi-supervised learning

Supervised learning  classification
Unsupervised learning  clustering
$k$-means clustering

Centers $\leftarrow k$ points sampled from the domain

repeat
   Assign points to closest center
   Centers $\leftarrow$ mean of assigned points
until convergence
\(k\)-means clustering

centers \(\leftarrow k\) points sampled from the domain

repeat

   Assign points to closest center

   centers \(\leftarrow\) mean of assigned points

until convergence
\textit{k}-means clustering

\texttt{centers} \leftarrow \textit{k} \text{ points sampled from the domain}

\textbf{repeat}

\hspace{1em} \text{Assign points to closest center}

\hspace{1em} \texttt{centers} \leftarrow \text{mean of assigned points}

\textbf{until} convergence
\( k \)-means clustering

centers \( \leftarrow k \) points sampled from the domain

repeat

Assign points to closest center

centers \( \leftarrow \) mean of assigned points

until convergence
$k$-means clustering

\begin{align*}
\text{centers} & \leftarrow k \text{ points sampled from the domain} \\
\text{repeat} & \\
\quad & \text{Assign points to closest center} \\
\quad & \text{centers} \leftarrow \text{mean of assigned points} \\
\text{until} & \text{convergence}
\end{align*}
$k$-means clustering

centers $\leftarrow k$ points sampled from the domain

repeat

Assign points to closest center

centers $\leftarrow$ mean of assigned points

until convergence
\textit{k}-means clustering

centers ← $k$ points sampled from the domain
repeat
    Assign points to closest center
    centers ← mean of assigned points
until convergence
EM clustering with Gaussian mixture model

Initialize mixture components: sample $k$ normal distribution parameter pairs

\[
\theta_i \leftarrow (\mu_i, \sigma_i), \quad i = 1, \ldots, k
\]

repeat
  compute points assignment
  compute parameters $\theta_i$
until convergence
**E(xpectation) step:** estimate the posterior probability that point $x$ was generated by component $i$

$$
\theta_i \leftarrow (\mu_i, \sigma_i), \quad i = 1, \ldots, k
$$

repeat

compute points assignment
compute parameters $\theta_i$

until convergence
EM clustering with Gaussian mixture model

M(aximization) step: estimate component parameters to maximize log-likelihood fit under current assignment

\[ \theta_i \leftarrow (\mu_i, \sigma_i), \quad i = 1, \ldots, k \]

repeat
  compute points assignment
  compute parameters \( \theta_i \)
until convergence
**EM clustering with Gaussian mixture model**

**E(xpectation) step:** estimate the posterior probability that point $x$ was generated by component $i$

\[ \theta_i \leftarrow (\mu_i, \sigma_i), \quad i = 1, \ldots, k \]

repeat
  compute points assignment
  compute parameters $\theta_i$
until convergence
EM clustering with Gaussian mixture model

**M(aximization) step:** estimate component parameters to maximize log-likelihood fit under current assignment

\[ \theta_i \leftarrow (\mu_i, \sigma_i), \quad i = 1, \ldots, k \]

repeat
   compute points assignment
   compute parameters \( \theta_i \)
until convergence
EM clustering with Gaussian mixture model

**E(xpectation) step:** estimate the posterior probability that point \( x \) was generated by component \( i \)

\[
\theta_i \leftarrow (\mu_i, \sigma_i), \quad i = 1, \ldots, k
\]

repeat

compute points assignment
compute parameters \( \theta_i \)

until convergence
**EM clustering with Gaussian mixture model**

**M(aximization) step:** estimate component parameters to maximize log-likelihood fit under current assignment

\[ \theta_i \leftarrow (\mu_i, \sigma_i), \quad i = 1, \ldots, k \]

repeat
  compute points assignment
  compute parameters \( \theta_i \)
until convergence
EM clustering with Gaussian mixture model

\[ \theta_i \leftarrow (\mu_i, \sigma_i), \quad i = 1, \ldots, k \]

repeat
  compute points assignment
  compute parameters \( \theta_i \)
until convergence
EM clustering with Gaussian mixture model

\[ i, \ldots, k \]

\[ \theta_i \leftarrow (\mu_i, \sigma_i), \quad i = 1, \ldots, k \]

repeat
\begin{align*}
&\text{compute points assignment} \\
&\text{compute parameters } \theta_i
\end{align*}
until convergence
EM clustering/Naives Bayes classification

Initialize mixture components: select $k$ and estimate parameters from labelled examples

\[ i (i ; i) ; i = 1, \ldots, k \]

repeat
    compute points assignment
    compute parameters $\theta_i$
until convergence
EM clustering/Naives Bayes classification

**E(xpectation) step:** estimate the posterior probability that point \( x \) was generated by component \( i \) for unlabelled examples

\[
\theta_i \leftarrow (\mu_i, \sigma_i), \quad i = 1, \ldots, k
\]

repeat

compute points assignment
compute parameters \( \theta_i \)

until convergence
**M(aximization) step:** estimate component parameters to maximize log-likelihood fit under current assignment of labelled and unlabelled examples

\[
\theta_i \leftarrow (\mu_i, \sigma_i), \quad i = 1, \ldots, k
\]

repeat
  compute points assignment
  compute parameters \(\theta_i\)
until convergence
EM clustering/Naives Bayes classification

E(xpectation) step: estimate the posterior probability that point $x$ was generated by component $i$ for unlabelled examples

$$\theta_i \leftarrow (\mu_i, \sigma_i), \quad i = 1, \ldots, k$$

repeat
  compute points assignment
  compute parameters $\theta_i$
until convergence
**M(aximization) step:** estimate component parameters to maximize log-likelihood fit under current assignment of labelled and unlabelled examples.

\[ \theta_i \leftarrow (\mu_i, \sigma_i), \quad i = 1, \ldots, k \]

repeat
    compute points assignment
    compute parameters \( \theta_i \)
until convergence
EM clustering/Naïves Bayes classification

\[
\theta_i \leftarrow (\mu_i, \sigma_i), \quad i = 1, \ldots, k
\]

repeat

compute points assignment
compute parameters \( \theta_i \)

until convergence
\[ \theta_i \leftarrow (\mu_i, \sigma_i), \quad i = 1, \ldots, k \]

repeat
  compute points assignment
  compute parameters \( \theta_i \)
until convergence
EM clustering/Naives Bayes classification

**Labelled examples** deterministic assignment
initialize parameters and stabilize EM process

**Unlabelled examples** probabilistic assignment
estimate the cluster structure
Represent the data as a graph

Start random walk from unlabelled node, stop at the first encountered labelled node

Assign class at which the random walk is most likely to terminate

**Key assumption:** the graph must be *label-connected*
Graph-based collective classification

Construct distance-based graph
where each instance is represented by a node
with an edge of weight $w_{ij}$ connecting instances $i$ and $j$

$$w_{ij} = e^{-d(x_i, x_j)^2 / \delta}$$
Graph-based collective classification

Turn into directed graph $G$

**unlabelled node** keep $k$ heaviest edges

**labelled node** remove outgoing edges and add self-loop

$$w_{ij} = e^{-d(x_i, x_j)^2/\delta}$$

$k = 5$
Graph-based collective classification

Simulate random walks on graph $G$
Compute the transition matrix $P$ associated to $G$
Let $N_G(i)$ denote the set of neighbors of $i$ in $G$

$$w_{ij} = e^{-d(x_i, x_j)^2 / \delta}$$

$$k = 5$$

$$p_{ij} = \frac{W_{ij}}{\sum_{j \in N_G(i)} W_{ij}}$$
Graph-based collective classification

Simulate random walks on graph $G$
Labelled instances constitute *absorbing nodes*
$P$ does not have a unique steady-state probability distribution

$$w_{ij} = e^{-\frac{d(x_i, x_j)^2}{\delta}}$$

$k = 5$

$$p_{ij} = \frac{w_{ij}}{\sum_{j \in N_G(i)} w_{ij}}$$
Graph-based collective classification

Simulate random walks on graph $G$

The steady-state distribution depends on the starting node $\pi_i^{(t)}$ probability vector after $t$ steps starting from node $i$

$\pi_i^{(0)}$ represents the starting state $i^{th}$ entry equals 1 all other entries equal 0
Graph-based collective classification

Simulate random walks on graph $G$

The steady-state distribution depends on the starting node $\pi_i^{(t)}$ probability vector after $t$ steps starting from node $i$

$$\pi_i^{(t)} = \pi_i^{(t-1)} P$$

$$= \pi_i^{(0)} p^t$$

$$\pi_i^{(\infty)} = \pi_i^{(0)} p^{\infty}$$
Graph-based collective classification

$P^\infty$ can be computed using the eigendecomposition of $P$.

In practice, steady-state distribution vectors $\pi_i^{(\infty)}$ are estimated with iterative method.

\[
\pi_i^{(t)} = \pi_i^{(t-1)} P = \pi_i^{(0)} P^t = \pi_i^{(0)} P^\infty
\]
Graph-based collective classification

Only entries of $\pi_i^{(\infty)}$ corresponding to absorbing nodes, i.e. labelled instances, are non zero.
Graph-based collective classification

Only entries of $\pi_i^{(\infty)}$ corresponding to absorbing nodes, i.e. labelled instances, are non zero
Compute class assignment probabilities as $\pi_i^{(\infty)} Y$
Graph-based collective classification

Compute class assignment probabilities as $\pi_i^{(\infty)} \gamma$
Graph-based collective classification

$k$ nearest neighbors can be seen as a special case

\[
\begin{align*}
w_{ij} &= \begin{cases} 
1 & \text{if } j \text{ is a labelled instance} \\
0 & \text{otherwise}
\end{cases} \\
k &= 5
\end{align*}
\]
Graph-based collective classification

$k$ nearest neighbors can be seen as a special case

\[ w_{ij} = \begin{cases} 
1 & \text{if } j \text{ is a labelled instance} \\
0 & \text{otherwise} 
\end{cases} \]

\[ k = 5 \]

\[ p_{ij} = \frac{w_{ij}}{\sum_{j \in N_G(i)} w_{ij}} \]
$k$ nearest neighbors can be seen as a special case.
Graph-based collective classification

$k$ nearest neighbors can be seen as a special case
Semi-supervised learning

Two types of approaches

Method-specific adaptations

- Transductive SVM
- Semi-supervised Bayes classification with EM

Graph-based collective classification

Generic meta-algorithms

- Self-training
- Co-training
Use the smoothness assumption to incrementally expand the labelled portion of the data
→ self-training
Self-training

Input data consists of subset of labelled data points $L$ and subset of unlabelled data points $U$

\[
\textbf{while } U \neq \emptyset \textbf{ do} \\
\quad \text{Train model } \mathcal{M} \text{ on } L \\
\quad \text{Generate predictions for } U \text{ with } \mathcal{M} \\
\quad W \leftarrow \{k \text{ data points from } U \text{ with highest confidence prediction}\} \\
\quad U \leftarrow U \setminus W; \quad L \leftarrow L \cup \{(x, f_\mathcal{M}(x)), x \in W\}
\]
Self-training example
Self-training with naive Bayes

Step 1   Step 2   Step 3   Final training
Self-training with naive Bayes

Input data | Prediction | Ground truth
---|---|---

acc = 0.89
Semi-supervised learning: generic meta-algorithms

Use the smoothness assumption to incrementally expand the labelled portion of the data
→ self-training

! Risk of error propagation and overfitting

Similar procedure but with two models trained on separate subsets of attributes generate labels for one another
→ co-training
Co-training

Input data consists of subset of labelled data points $L$ and subset of unlabelled data points $U$

Data attributes are split into disjoint subsets $A_1$ and $A_2$ should be as independent from each other as possible

Initially, $L_1$ and $U_1$ contain the data points from $L$ and $U$ projected on $A_1$, respectively for $L_2$ and $U_2$
Co-training

while $U_1 \cup U_2 \neq \emptyset$ do

Use $M_1$ to generate training labels for $M_2$

Train model $M_1$ on $L_1$
Generate predictions for $U_2$ with $M_1$
$W \leftarrow \{k$ data points from $U_2$
with highest confidence prediction$\}$
$U_2 \leftarrow U_2 \setminus W; \quad L_2 \leftarrow L_2 \cup \{(x, f_{M_1}(x)), x \in W\}$

Use $M_2$ to generate training labels for $M_1$

Train model $M_2$ on $L_2$
Generate predictions for $U_1$ with $M_2$
$W \leftarrow \{k$ data points from $U_1$
with highest confidence prediction$\}$
$U_1 \leftarrow U_1 \setminus W; \quad L_1 \leftarrow L_1 \cup \{(x, f_{M_2}(x)), x \in W\}$
The co-training procedure results in two models $\mathcal{M}_1$ and $\mathcal{M}_2$ making decision based on attributes in $A_1$ and $A_2$ respectively. Final predictions are obtained by combining the predictions from both models.
Co-training example

\[ A_1 = \{ \text{sepal length, petal width} \} \]
\[ A_2 = \{ \text{sepal width, petal length} \} \]
Co-training with naive Bayes

Step 1

\( M_1 \) on \( A_1 \)

\( M_2 \) on \( A_2 \)

Step 2

\( M_1 \) on \( A_1 \)

\( M_2 \) on \( A_2 \)
Co-training with naive Bayes

Step 2

$M_2$ on $A_2$

$M_1$ on $A_1$

Step 3

$M_2$ on $A_2$

$M_1$ on $A_1$
Co-training with naive Bayes

Step 3

\( M_1 \) on \( A_1 \)  
\( M_2 \) on \( A_2 \)

Final training

\( M_1 \) on \( A_1 \)  
\( M_2 \) on \( A_2 \)
Co-training with naive Bayes

\[ M_1 \]

\[ M_2 \]

\[ \text{combined} \]
Co-training with naive Bayes

Input data

Prediction
Co-training with naive Bayes

Ground truth

Prediction

acc = 0.95
Semi-supervised learning

Method-specific adaptations

- Transductive SVM
- Semi-supervised Bayes classification with EM

Graph-based collective classification

Generic meta-algorithms

- Self-training
- Co-training

Working assumption:
class structure approximately matches clustering structure

Most useful when labelled examples are scarce
Active learning
Active learning

Labelled data is difficult and expensive to acquire
Often requires manual labelling, through e.g. expert examination or crowdsourcing mechanisms
Labels might constitute sensitive information
Data collection might raise privacy concerns and require obtaining permission from relevant entities
The process of acquiring labels can be time consuming, work-intensive, expensive, susceptible to errors
Cost can be evaluated or at least estimated
Querying iris species

Not all training instances are equally useful
Querying iris species

Not all training instances are equally useful
Active learning

Labelled data is difficult and expensive to acquire
Cost can be evaluated or at least estimated
Not all training instances are equally useful
Active learning

Labelled data is difficult and expensive to acquire
Cost can be evaluated or at least estimated
Not all training instances are equally useful

The aim of active learning is to train the most accurate model within a given budget
Integrate label acquisition and model building to achieve highest cost-efficiency

Active learning is sometimes known as query learning or optimal experimental design
Active learning assumes access to an oracle, i.e. a means to obtain labels for queried instances, seen as a black-box.

The querying system asks the oracle for the labels of specific instances, selected following some strategy.
Active learning scenarios

Membership query synthesis generates a synthetic instance, but the instance might not be realistic.

Selective sampling: unlabelled instances arrive one by one and the learner makes a decision to query the label from the oracle or to discard (a.k.a. stream-based or sequential AL).

Pool-based sampling: a collection of interesting examples to query is sampled from a large pool of available unlabelled instances.

Focus on the latter, most common scenario.
The active learning process is iterative and starts with:

- small collection of labelled instances $L$
- large collection of unlabelled instances $U$
- query budget $b$
The active learning process is iterative and starts with

- small collection of labelled instances $L$
- large collection of unlabelled instances $U$
- query budget $b$

$f_O(x)$ is the label for data point $x$ obtained from oracle $O$
$c_O(x)$ is the associated cost

Cost might be instance-specific, but most often assumed equal for all instances
Active learning process

The active learning process is iterative and starts with

- small collection of labelled instances $L$
- large collection of unlabelled instances $U$
- query budget $b$

$f_O(x)$ is the label for data point $x$ obtained from oracle $O$
$c_O(x)$ is the associated cost

\[
\text{while } b > 0 \text{ and accuracy improves do }
\]
\[
\begin{align*}
\text{Train model } M \text{ on } L \\
C &\leftarrow \{\text{most interesting instances from } U\} \\
U &\leftarrow U \setminus C \\
L &\leftarrow L \cup \{(x, f_O(x)) \text{ for } x \in C\} \\
b &\leftarrow b - \sum_{x \in C} c_O(x)
\end{align*}
\]
Clearly, the crucial part of active learning is the selection of *most interesting instances*, i.e. the querying strategy

\[
\text{while } b > 0 \text{ and accuracy improves do} \\
\begin{align*}
\text{Train model } M \text{ on } L \\
C &\leftarrow \{\text{most interesting instances from } U\} \\
U &\leftarrow U \setminus C \\
L &\leftarrow L \cup \{(x, f_0(x)) \text{ for } x \in C\} \\
b &\leftarrow b - \sum_{x \in C} c_0(x)
\end{align*}
\]
**Heterogeneity-based strategies** sample regions that are uncertain, heterogeneous or dissimilar to what has been seen so far
Heterogeneity-based strategies

The uncertainty sampling strategy selects the instances whose label is least certain, i.e. for which current prediction has lowest confidence.

If the classifier returns class assignment probabilities compute

$$\text{entropy}(x) = \sum_{c \in C} -\rho_c(x) \log_2(\rho_c(x))$$

with $\rho_c(x)$ the probability that $x$ belongs to class $c$ according to the classifier.

Higher entropy corresponds to greater uncertainty.

Intuitively, selects points closest to the decision boundary.
Querying iris species

Random sampling

Uncertainty sampling

\[ \text{acc} = 0.706 \]
Querying iris species

Random sampling

![Random sampling graph]

Uncertainty sampling

![Uncertainty sampling graph]

acc = 0.824

acc = 0.882
Querying iris species

Random sampling

Uncertainty sampling

\[ \text{acc} = 0.882 \]

\[ \text{acc} = 1.000 \]
Querying iris species

Random sampling

Uncertainty sampling

acc = 0.882

acc = 0.941
Heterogeneity-based strategies

The **query-by-committee** strategy uses a committee of classifiers and selects the instances on which there is most disagreement among classifiers.

Compute

$$\text{entropy}(x) = \sum_{c \in C} - \rho_c(x) \log_2(\rho_c(x))$$

with $\rho_c(x)$ the fraction of classifiers predicting that $x$ belongs to class $c$.

Higher entropy corresponds to lower agreement.
The **expected model change** strategy selects the instance that, when added to training instances, results in the largest expected model change.

Considering an optimization-based classification model and current training set $L$, let $\nabla G(L)$ denote the gradient of the objective function with respect to model parameters.

The change resulting from the addition of $x$, assuming it belongs to class $c$ is

$$\| \nabla G(L \cup \{(x, c)\}) - \nabla G(L) \|$$
Heterogeneity-based strategies

The expected model change strategy selects the instance that, when added to training instances, results in the largest expected model change.

Considering an optimization-based classification model and current training set $L$, let $\nabla G(L)$ denote the gradient of the objective function with respect to model parameters.

The expected model change is

$$\sum_{c \in C} \rho_c(x) \| \nabla G(L \cup \{(x, c)\}) - \nabla G(L) \|$$

with $\rho_c(x)$ the probability that $x$ belongs to class $c$ according to the classifier.
Heterogeneity-based strategies sample regions that are uncertain, heterogeneous or dissimilar to what has been seen so far.

Sampling at the proximity of decision boundary prone to adding unrepresentative outliers.
Querying strategies

Heterogeneity-based strategies sample regions that are uncertain, heterogeneous or dissimilar to what has been seen so far.

Performance-based strategies evaluate the impact of adding the queried instance on the performance of the model.
Performance-based strategies

The **expected error reduction** strategy selects the instance that, when added to training instances, minimizes the expected label uncertainty on remaining unlabelled instances.

Motivation: greater certainty in prediction of class labels of the remaining unlabelled instances, should result in lower error rate on unseen test set.
Querying strategies

uncertainty sampling

**maximize** label uncertainty on **queried instance**

vs.

**expected error reduction**

**minimize** label uncertainty on **remaining unlabelled instances**
Performance-based strategies

The expected error reduction strategy selects the instance $x'$ that minimizes

$$
\sum_{c' \in C} \rho_{c'}(x') \sum_{x \in U \setminus \{x'\}} \text{entropy}^{(x', c')}(x)
$$

with

$$
\rho_c(x) \text{ the probability that } x \text{ belongs to class } c \\
\rho_c^{(x', c')}(x) \text{ the probability that } x \text{ belongs to class } c \\
\text{according to the current classifier, trained on } L \\
\text{according to the classifier trained on } L \cup \{(x', c')\}
$$

$$
\text{entropy}^{(x', c')}(x) = \sum_{c \in C} -\rho_c^{(x', c')}(x) \log_2(\rho_c^{(x', c')}(x))
$$
Performance-based strategies

The expected error reduction strategy selects the instance that, when added to training instances, minimizes the expected label uncertainty on remaining unlabelled instances.

Training a new model to test the effect of adding a particular instance and class label is computationally expensive.

$$\text{error} = \text{bias} + \text{variance} + \text{noise}$$

Only the variance depends highly on the choice of instances. The variance can be expressed in closed form.

Expected variance reduction strategies aim to reduce the variance instead of the error.
Querying strategies

Heterogeneity-based strategies sample regions that are uncertain, heterogeneous or dissimilar to what has been seen so far.

Performance-based strategies evaluate the impact of adding the queried instance on the performance of the model.

Instead of focusing on the queried instance, try to improve the error behavior on the aggregate set of unlabelled instances.

Avoid unrepresentative instances.
Querying strategies

**Heterogeneity-based strategies** sample regions that are uncertain, heterogeneous or dissimilar to what has been seen so far.

**Performance-based strategies** evaluate the impact of adding the queried instance on the performance of the model.

**Representativeness-based strategies** query instances so as to obtain a distribution of instances that is representative of the underlying population.
Representativeness can be measured using density.

Incorporate representativeness weighting to heterogeneity-based strategies to avoid selecting outliers.

**Heterogeneity score** of $x$ w.r.t. the current model $H(x)$ is the uncertainty of the model prediction on $x$ or some other heterogeneity criterion to maximize.

**Representativeness** of $x$ w.r.t. unlabelled instances $U$ $R(x, U)$ is larger if $x$ is located in a dense region of $U$.

Select instances that maximize $H(x) \cdot R(x, U)$.