Section 8

Recommender systems

Recommender systems

Scenario: a service where users consume items

- predict the users' rating to unconsumed items

Recommender systems

Scenario: a service where users consume items

- predict the users' rating to unconsumed items
- great interest from industry
  - e-commerce
  - online social networks
  - entertainment on demand
- users usually pay for consuming

Example: movie recommendation

Movie	Alice	Bob	Carol	Dave
Hugs and kisses				
Sweetness day				
The true love				
Crazy Max				
The final judgement				

Example: movie recommendation

Movie	Alice	Bob	Carol	Dave
Hugs and kisses	5	5	0	0
Sweetness day	5			0
The true love		4	0	
Crazy Max	0	0	5	4
The final judgement	0	0	5	

- some users rated some movies

Example: movie recommendation

Movie	Alice	Bob	Carol	Dave
Hugs and kisses	5	5	0	0
Sweetness day	5	?	?	0
The true love	?	4	0	?
Crazy Max	0	0	5	4
The final judgement	0	0	5	?

- some users rated some movies
- predict the rating of unrated movies
  - **Q:** and then? where's the recommendation?

Content-based representation

Suppose 2 features (independent variables) exist for movies (*content of*)

- X_1 represents "romance"
- X_2 represents "action"
- $X_0 = 1$ represents bias

Movie	Alice	Bob	Carol	Dave	X_1	X_2
Hugs and kisses	5	5	0	0	0.95	0.01
Sweetness day	5	?	?	0	1	0
The true love	?	4	0	?	0.99	0
Crazy Max	0	0	5	4	0	1
The final judgement	0	0	5	?	0.02	0.99

Notation

Movie	Alice	Bob	Carol	Dave	X_1	X_2
Hugs and kisses	5	5	0	0	0.95	0.01
Sweetness day	5	?	?	0	1	0
The true love	?	4	0	?	0.99	0
Crazy Max	0	0	5	4	0	1
The final judgement	0	0	5	?	0.02	0.99

- $r_{i,j} \in \{0, 1\}$ is 1 iff user j rated movie i
- $y_{i,j}$ is rating given by user j to movie i (iff $r_{i,j} = 1$)
- x_i is the feature vector of movie i

Notation

Movie	Alice	Bob	Carol	Dave	X_1	X_2
Hugs and kisses	5	5	0	0	0.95	0.01
Sweetness day	5	?	?	0	1	0
The true love	?	4	0	?	0.99	0
Crazy Max	0	0	5	4	0	1
The final judgement	0	0	5	?	0.02	0.99

- $r_{i,j} \in \{0, 1\}$ is 1 iff user j rated movie i
- $y_{i,j}$ is rating given by user j to movie i (iff $r_{i,j} = 1$)
- x_i is the feature vector of movie i

"predict the rating of unrated movies" corresponds to **solving n_u (number of users) regression problems**

- learn $f_{\text{Alice}}(x)$, $f_{\text{Bob}}(x)$, ...

Recommendation as linear regression

Assume a linear dependency between rating and features:

RATING MOVIE i BY USER j

$$y_{i,j} = \theta_{0,j} x_{i,0} + \theta_{1,j} x_{i,1} + \theta_{2,j} x_{i,2} + \dots$$
$$= \theta_j^T x_i$$

$x_{i,1}$: ROMANCE; $x_{i,2}$: ACTION; x_i: FEATS OF MOVIE i; θ_j: PREFS OF USER j

- $\theta_j \in \mathbb{R}^p$ is the set of parameters of user j
  - θ_j represents preferences of user j

"solving n_u (number of users) regression problems" corresponds to **for each user j, learn θ_j**

Learning θ_j

Goal: learned θ_j should be minimize errors on already rated movies:

$$\min_{\theta_j} \frac{1}{2} \sum_{i=1}^{n} \left(\theta_j^T x_i - y_{i,j} \right)^2$$

OVERALL ERROR OF PREDICTION FOR USER j

ONE USER

ALL MOVIES

Learning θ_j

Goal: learned θ_j should be minimize errors on already rated movies:

$$\min_{\theta_j} \frac{1}{2} \sum_{i=1}^{n} \left(\theta_j^T x_i - y_{i,j} \right)^2$$

- minimize sum of squared errors

Learning θ_j

Goal: learned θ_j should be minimize errors on already rated movies:

$$\min_{\theta_j} \frac{1}{2} \sum_{i=1}^{n} r_{i,j} \left(\theta_j^T x_i - y_{i,j} \right)^2$$

- minimize sum of squared errors
- consider only rated movies ($r_{i,j} = 0$ for unrated)

Learning θ_j

Goal: learned θ_j should be minimize errors on already rated movies:

$$\min_{\theta_j} \frac{1}{2} \sum_{i=1}^{n} r_{i,j} \left(\theta_j^T x_i - y_{i,j} \right)^2 + \frac{\lambda}{2} \sum_{k=1}^{p} \theta_{k,j}^2$$

- minimize sum of squared errors
- consider only rated movies ($r_{i,j} = 0$ for unrated)
- with regularization (**Q:** role of λ?)

Learning θ_j

Goal: learned θ_j should be minimize errors on already rated movies:

$$\min_{\theta_j} \frac{1}{2} \sum_{i=1}^{n} r_{i,j} \left(\theta_j^T x_i - y_{i,j} \right)^2 + \frac{\lambda}{2} \theta_j^T \theta_j$$

- minimize sum of squared errors
- consider only rated movies ($r_{i,j} = 0$ for unrated)
- with regularization (**Q:** role of λ?)

Learning θ_j

Goal: learned θ_j should be minimize errors on already rated movies:

$$\min_{\theta_j} \frac{1}{2} \sum_{i=1}^{n} r_{i,j} \left(\theta_j^T x_i - y_{i,j} \right)^2 + \frac{\lambda}{2} \theta_j^T \theta_j$$

- minimize sum of squared errors
- consider only rated movies ($r_{i,j} = 0$ for unrated)
- with regularization (**Q:** role of λ?)

For all users:

$$\min_{\theta_1, \ldots, \theta_{n_u}} \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i=1}^{n} r_{i,j} \left(\theta_j^T x_i - y_{i,j} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \theta_j^T \theta_j$$

P×1 USER 1×P MOVIE

Can be solved with any optimization algorithm, e.g., gradient descent

The movie features?

- Which features to characterize movies? (or songs, products, people, . . .)
- Who assign feature values to movies? Is it costly?
  - scale?

The movie features?

- Which features to characterize movies? (or songs, products, people, . . .)
- Who assign feature values to movies? Is it costly?
  - scale?

Assume we "want" p features:

Movie	Alice	Bob	Carol	Dave	X_1	. . .	X_p
Hugs and kisses	5	5	0	0	?	?	?
Sweetness day	5	?	?	0	?	?	?
The true love	?	4	0	?	?	?	?
Crazy Max	0	0	5	4	?	?	?
The final judgement	0	0	5	?	?	?	?

Collaborative filtering

In content-based:

- we know movie features $x_1, x_2, \ldots, x_n$ and ratings $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{n_u}$
- we learn users' preferences $\theta_1, \theta_2, \ldots, \theta_{n_u}$

Collaborative filtering

In content-based:

- we know movie features $x_1, x_2, \ldots, x_n$ and ratings $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{n_u}$
- we learn users' preferences $\theta_1, \theta_2, \ldots, \theta_{n_u}$

Assume we know users' preferences:

- we learn movie features

IMPLICITLY COLLABORATE

Collaborative filtering

In content-based:

- we know movie features $x_1, x_2, \ldots, x_n$ and ratings $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{n_u}$
- we learn users' preferences $\theta_1, \theta_2, \ldots, \theta_{n_u}$

Assume we know users' preferences:

- we learn movie features

Users (implicitly) **collaborate** to characterize content

Learning features from preferences

One movie:

$$\min_{x_i} \frac{1}{2} \sum_{j=1}^{n_u} r_{i,j} \left(\theta_j^T x_i - y_{i,j} \right)^2 + \frac{\lambda}{2} x_i^T x_i$$

FEATS

PREDICT ERR FOR USER/MOVIE

REGULARIZ.

1 MOVIE

All movies:

$$\min_{x_i, \ldots, x_n} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n_u} r_{i,j} \left(\theta_j^T x_i - y_{i,j} \right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n} x_i^T x_i$$

Users' preferences

"Assume we want p features" corresponds to "users' preferences are p dimensional"

- how to collect users' preferences?
- how many?
- relation with "linear dependency assumption"?

Users' preferences

"Assume we want p features" corresponds to "users' preferences are p dimensional"

- how to collect users' preferences?
- how many?
- relation with "linear dependency assumption"?

It may be preferable to learn features and preferences together!

Learning features and preferences

$$\min_{x_i,\ldots,x_n,\theta_1,\ldots,\theta_{n_u}} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n_u} r_{i,j}\left(\theta_j^T x_i - y_{i,j}\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{n} x_i^T x_i + \frac{\lambda}{2}\sum_{j=1}^{n_u}\theta_j^T\theta_j$$

New user?

Movie	Alice	Bob	Carol	Dave	Eric
Hugs and kisses	5	5	0	0	?
Sweetness day	5	?	?	0	?
The true love	?	4	0	?	?
Crazy Max	0	0	5	4	?
The final judgement	0	0	5	?	?

$$\min_{x_i,\ldots,x_n,\theta_1,\ldots,\theta_{n_u}} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n_u} r_{i,j}\left(\theta_j^T x_i - y_{i,j}\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{n} x_i^T x_i + \frac{\lambda}{2}\sum_{j=1}^{n_u}\theta_j^T\theta_j$$

New user?

Movie	Alice	Bob	Carol	Dave	Eric
Hugs and kisses	5	5	0	0	?
Sweetness day	5	?	?	0	?
The true love	?	4	0	?	?
Crazy Max	0	0	5	4	?
The final judgement	0	0	5	?	?

$r_{i,5} = 0, \forall i$

$$\min_{x_i,\ldots,x_n,\theta_1,\ldots,\theta_{n_u}} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n_u} r_{i,j}\left(\theta_j^T x_i - y_{i,j}\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{n} x_i^T x_i + \frac{\lambda}{2}\sum_{j=1}^{n_u}\theta_j^T\theta_j$$

- sum of squared errors is always zero for the new user

New user?

COLD START PROBLEM

Movie	Alice	Bob	Carol	Dave	Eric
Hugs and kisses	5	5	0	0	?
Sweetness day	5	?	?	0	?
The true love	?	4	0	?	?
Crazy Max	0	0	5	4	?
The final judgement	0	0	5	?	?

$$\min_{x_i,\ldots,x_n,\theta_1,\ldots,\theta_{n_u}} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n_u} r_{i,j}\left(\theta_j^T x_i - y_{i,j}\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{n} x_i^T x_i + \frac{\lambda}{2}\sum_{j=1}^{n_u} \theta_j^T \theta_j$$

- sum of squared errors is always zero for the new user
- the only goal is to minimize "sum of preferences"
  - **Q:** what happens without regularization?
- which results in no preferences ($\forall_k, \theta_{k,j} = 0$), and hence equal predicted ratings for all movies

Cold start problem

When something "new" arrives and no data is available:

- new user
- new movie (**Q:** is really a problem?)

One possible solution: use mean values

$$\mathbf{y}_{j'} = \frac{1}{n_u} \sum_{j=1}^{n_u} \mathbf{y}_j$$

Recommender system assessment

Many options:

- as a regression problem, RMSE

Recommender system assessment

Many options:

- as a regression problem, RMSE
- as a classification problem, accuracy
  - does the tool recommend the most preferred item to the user?

Recommender system assessment

Many options:

- as a regression problem, RMSE
- as a classification problem, accuracy
  - does the tool recommend the most preferred item to the user?
- as a classification problem, accuracy@K
  - does the tool recommend the most preferred item to the user among the top k recommendations?

Recommender system assessment

Many options:

- as a regression problem, RMSE
- as a classification problem, accuracy
  - does the tool recommend the most preferred item to the user?
- as a classification problem, accuracy@K
  - does the tool recommend the most preferred item to the user among the top k recommendations?
- as an information retrieval problem, *precision* and *recall*
  - does the tool recommend *relevant* items?

$$\text{Prec.} = \frac{\#(\text{relevant} \wedge \text{recomm.})}{\#\text{recomm.}} \qquad \text{Rec.} = \frac{\#(\text{relevant} \wedge \text{recomm.})}{\#\text{relevant}}$$

Q: relation with FPR, FNR?

Recommender system assessment

Many options:

- as a regression problem, RMSE
- as a classification problem, accuracy
  - does the tool recommend the most preferred item to the user?
- as a classification problem, accuracy@K
  - does the tool recommend the most preferred item to the user among the top k recommendations?
- as an information retrieval problem, *precision* and *recall*
  - does the tool recommend *relevant* items?

$$\text{Prec.} = \frac{\#(\text{relevant} \wedge \text{recomm.})}{\#\text{recomm.}} \qquad \text{Rec.} = \frac{\#(\text{relevant} \wedge \text{recomm.})}{\#\text{relevant}}$$

Q: relation with FPR, FNR?

In practice, how to measure them?

Beyond accuracy

- diversity
- serendipity
  - positive surprise
- revenue? number of click/user/usages?

More in general, UI plays a crucial role!

Section 9

Evolutionary computation

What is Machine Learning?

Definition

Machine Learning is the science of getting computer to learn without being explicitly programmed.

Up to now

"learn without being explicitly" $\rightarrow$ refine some predefine more general solution scheme

- RF for regression $\rightarrow$ find a *good* forest
- SVM for binary classification $\rightarrow$ find a *good* hyperplane

Up to now

"learn without being explicitly" $\rightarrow$ refine some predefine more general solution scheme

- RF for regression $\rightarrow$ find a *good* forest
- SVM for binary classification $\rightarrow$ find a *good* hyperplane

We have some (quite precise) idea (the *hypothesis*) about the nature of the solution: a tree, an hyperplane, ...

What if we do not?

No hypothesis

- We just have a way to assess a candidate solution
- No hypothesis
- Computer: be free, learn a (good) solution!

How?

No hypothesis

- We just have a way to assess a candidate solution
- No hypothesis
- Computer: be free, learn a (good) solution! (= program yourself!)

How? A significant case:

- problem: life
- user: God?
- computer: nature
- learning method: natural evolution

Evolutionary process

A general and basic scheme:

- a population of individuals compete for limited resources
- the population is dynamic: individuals die and are born
- fittest individual survive and reproduce more than the others
- offspring inherit some characters from parents (they are similar but not identical)

Evolutionary process

A general and basic scheme:

- a population of individuals compete for limited resources
- the population is dynamic: individuals die and are born
- fittest individual survive and reproduce more than the others
- offspring inherit some characters from parents (they are similar but not identical)

On/by/for computers? Evolutionary computation (EC)

EC: a bit of history

1930s	first ideas
1960s	ideas development using first computers
1970s	exploration
1980s	exploitation
1990s	unification
2000s+	mature expansion

Communities

At least three communities:

- biologists: simulate/understand real evolution
- computer scientists/engineers: build interesting artifacts
- artificial-life researchers: build/study artificial worlds

Result:

- some duplications
- different vocabularies
- strong habits

Kenneth A De Jong. *Evolutionary computation: a unified approach*. MIT press, 2006

What can be taught/learned?

Here:

- general scheme
- terminology
- some significant variants
- general usage guidelines

Not here:

- (variant) details
- detailed motivation ("theory")
- specific tools

General scheme

- a population of individuals compete for limited resources
- the population is dynamic: individuals die and are born
- fittest individual survive and reproduce more than the others
- offspring inherit some characters from parents (they are similar but not identical)

EVOL. ALGORITHM

Some questions:

- what is an individual?
- what is a population? what are resources?
- how individuals compete?
- how fitness is measured? → IF NOT OPEN-ENDED EVOLUTION
- how do individual reproduce?

Individual

A candidate solution for the considered problem:

- a program in a given programming language
- a set of numerical parameters
- a picture
- ...

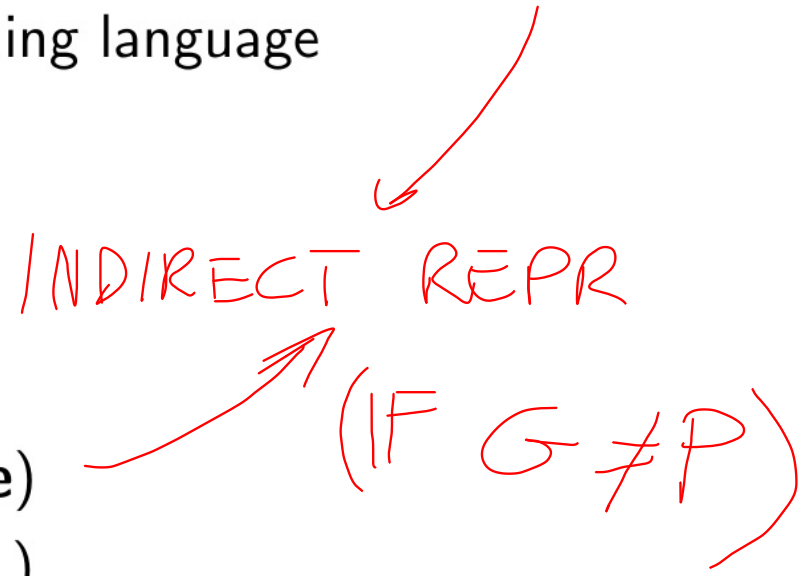
Internally represented as:

- itself (program, set, picture, ...)
- some well defined data structure:
  - a fixed/variable-length string of bits
  - an abstract syntax tree
  - ...

Individual

A candidate solution for the considered problem: (**phenotype**)

- a program in a given programming language
- a set of numerical parameters
- a picture
- ...



Internally represented as: (**genotype**)

- itself (program, set, picture, ...)
- some well defined data structure:
  - a fixed/variable-length string of bits
  - an abstract syntax tree
  - ...

Individual: why genotype/phenotype?

GENOTYPE FOR RECOMB, PHENOTYPE FOR FITNESS
- BIT/INT STRING
- $\in \mathbb{R}^m$
- TREES

- To resemble nature
- To ease manipulation
  - how two programs should reproduce?
  - how two images should reproduce?
- To allow reuse, hence enabling actual usage of EC
  - someone found a good way of making bits strings reproduce
  - user "just" need to decide how to transform (**genotype-phenotype mapping**) a bits string to his/her solution form (e.g., numerical parameters)

Population and competition for resources

Mainstream:

- a population is a set of individuals with a fixed (max) size
- "limited resources" is a place in the population

The population is dynamic:

- when a new individual is born, some individual must leave the population (die): which one?

Population dynamics

How/when individuals are replaced? (**generational model** or replacement strategy)

Underlying (and common) assumptions:

- individuals life is instantaneous
  - given the genotype, the phenotype (if any) and the fitness are immediately known
- time flowing is determined by births (and deaths)

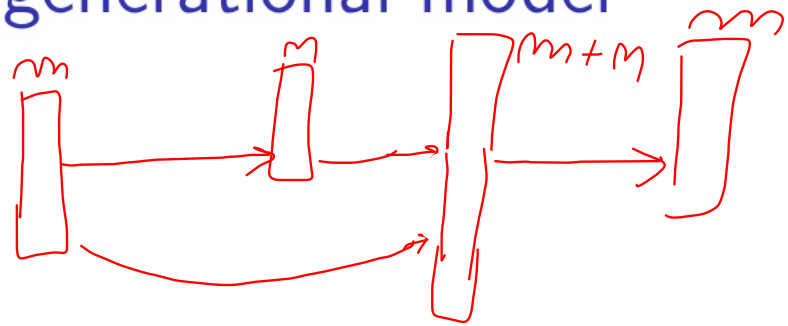
Generational model: general scheme

Parameters:

- a population of m parents
- a population of n offspring (built from parents; how? later)
- a boolean flag (overlapping vs. non-overlapping)

(Recall: population size is fixed)

Overlapping generational model



At each time tick:

1. build n offspring from the m parents
2. obtain an $n + m$ population by merging parens and offspring
3. select m individuals to survive

Non-overlapping generational model

At each time tick:

1. build n offspring from the m parents (assume $n \geq m$)
2. select m individuals to survive among the n offspring

All parents die!

Common cases

$m \sim [10, 1000]$

- $n = m$, overlapping
- $n = m$, non-overlapping
- $n = 0.8m$, overlapping
- $n = 1$, overlapping (steady state)

Common cases

- $n = m$, overlapping
- $n = m$, non-overlapping
- $n = 0.8m$, overlapping
- $n = 1$, overlapping (steady state)

Problem:

- different degrees of dynamicity in the single time tick
  - makes different variants comparison difficult

Solution:

- measure time flowing as number of births referred to population size m
- a **generation** occurs each m births

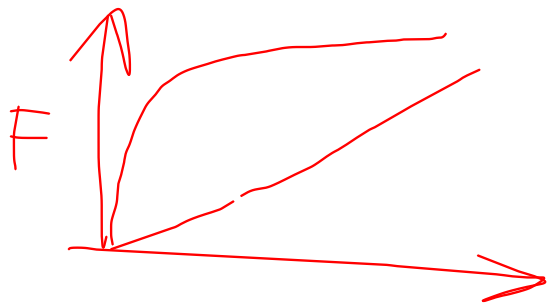
Selection criteria

How to

- select individuals to survive?
- select parents to reproduce?

Many options:

- uniform (neutral) selection
- fitness-proportional selection
- rank-proportional selection
- truncation selection
- tournament selection
- ...



Fitness/rank-proportional

Fitness-proportional:

1. given the numerical fitness of each individual
2. randomly pick one individual with probability proportional to the fitness (the better, the larger probability)

Rank-proportional:

1. given the rank of each individual in a fitness-based ranking
2. randomly pick one individual with probability proportional to the rank (the better, the larger probability)

(Can be applied to a non-numerical fitness, in principle)

Uniform and truncation

Uniform:

1. pick randomly an individual (with uniform probability)

Truncation:

1. pick the best individual (**elitism**)

(Deterministic)

Tournament selection

Given a parameter n_{size} (size of the tournament):

1. randomly (with uniform probability) pick n_{size} individuals
2. from them, choose the one with the best fitness

UNIFORM + TRUNCATION

Selection criteria differences

Is criterion A better than criterion B? *Just* measure!

Criteria differ in how strongly they tend to prefer fit vs. unfit individuals:

- uniform selection: no preferences
- truncation selection: strong preference of fit individuals
- tournament: $n_{\text{size}} \to 1$: no preference, $n_{\text{size}} \to m$: strong preference

Selecting fit/unfit individuals

Strong preference (or selective/evolutionary pressure):

- population tends to converge to fittest individuals
- evolution concentrates in improving most promising solutions (**exploitation**)
- risk of "falling" in local optimum

Weak preference (or selective/evolutionary pressure):

- population includes also unfit individuals
- evolution investigates many different (maybe not promising) solutions (**exploration**)
- risk of not finding a good solution

Exploration/exploitation trade-off is hard to rule!

Selectors: common cases

- Reproduction: tournament of n_{size}
  - e.g., $m = n_{\text{pop}} = 500$, $n_{\text{size}} = 5$
- Survival: truncation

- Reproduction: fitness proportional
- Survival: truncation

Reproduction

Build n offspring from the m parents. How?
General scheme:

- given one or more parents, an offspring is generated by applying a unary or binary **genetic operator** on parent genotypes
  - unary (**mutation**): $f : \mathcal{G} \to \mathcal{G}$
  - binary (recombination or **crossover**): $f : \mathcal{G}^2 \to \mathcal{G}$
- given n and a set of weighted operators, generate offspring with operators according to their weights (deterministically or stochastically)

Choice of operators

Operators:

- crossover for generating 80% of offspring
- mutation for generating 20% of offspring

Deterministically:

1. for $0.8n$ times
   - 1.1 select 2 parents (with reproduction selection criterion)
   - 1.2 apply crossover to genotypes
2. for $0.2n$ times
   - 2.1 select 1 parent (with reproduction selection criterion)
   - 2.2 apply mutation to genotype

Choice of operators

Operators:

- crossover for generating 80% of offspring
- mutation for generating 20% of offspring

Stochastically:

1. for n times
   - 1.1 randomly choose between mutation/crossover with 20/80 probability
   - 1.2 select 1 or 2 parents (with reproduction selection criterion) accordingly
   - 1.3 apply operator to genotype(s)