Expedia Hotel Recommendations

kaggle.com/c/expedia-hotel-recommendations/discussion/21607

idle_speculation

1st place

1st Place Solution Summary

Posted in expedia-hotel-recommendations 5 years ago

arrow_drop_up

152

I'd like to note that I'll be donating 10% of the prize to the American Cancer Society in honor of Lucas. He was always an inspiration to me here on Kaggle.

In terms of the solution, let me describe a couple of the components before getting into the model.

Distance Matrix Completion

The idea was to map users and hotels to locations on a sphere. If we can do this successfully, then we can 'widen the leak' not just to previously occurring distances but to potentially any example where the distance between user and hotel is known.

One immediate issue with this approach is the question of what combinations of columns to use in uniquely identifying user and hotel locations. For users, the tuple U=userlocationcountry,userlocationregion,userlocationcity was a natural choice. For hotels things are less clear. Two parallel notions of hotel location were developed H1=hotelcountry,hotelmarket,hotelcluster and H2=srchdestinationid,hotelcluster. Both the H1 and H2 versions were used as features and the final model was relied upon to sort out which was more useful.

For both H1 and H2, user and hotel locations were randomly initialized on a sphere and gradient descent was applied to the spherical law of cosines formula on the distinct combinations of U,H,origdestinationdistance.

Convergence for the gradient descent was not quick at all. Using nesterov momentum and a gradual transition from squared error to absolute error, the process took about 10^11 iterations and 36 hours.

In the end, the average errors for H1 and H2 were around 1.8 and 3.7 miles respectively and I'm looking forward to finding out what tricks the other teams had here.

Factorization Machines

These tend to pick up on interactions between categorical variables with a large number of distinct values. The hope was that they would find some user-hotel preferences in-line with Expedia's description of the problem.

The implementation used was LIBFFM. For each of the 100 hotel clusters, a seperate factorization machine model was built using the categorical features in the training set. The attributes for each model were the same but the target for k-th model was an indicator of whether the hotel cluster was equal to cluster k.

These FM submodels added about 0.002 to the validation map@5. Aside from leak related features, these provided the most lift over the base click and book rates.

Training/Validation Setup

The strategy here was to build a model on the last 6 months of 2014, from '2014-07-01' onwards. In order to mirror the situation in the test set, the features used to train the model were generated using only hotel cluster and click data prior to '2014-07-01'.

When scoring the model for the test set, all the features were refreshed using all available training data.

A portion of the user_ids from '2014-07-01' onward was set aside as validation. This did not line up precisely with the leaderboard, but it probably agrees directionally.

Learn-to-Rank Model

In order to turn this into an xgboost "rank:pairwise" problem, each booking in the training sample was "burst" into 100 rows in the training set. The features for each row were generated relative to the corresponding cluster.

The features input into the model were essentially the historical book and click rates, distances derived from the matrix completion, and the factorization machine scores. So, the i-th row corresponding to the j-th booking would be something like:

- the historical click and book rates of cluster i for someone having the attributes of the j-th booking
- the difference between the matrix completion predicted distance and the distance provided for the j-th booking
- the factorization machine score for the i-th cluster based on the attributes of the j-th booking.

For the final scoring, the test set is also "burst" into 100 rows per booking instance and each booking-cluster combination provides a score. The top 5 scoring clusters per booking are submitted as the solution.

A few details were left out in the interest of brevity as this post is already quite long, but the above more or less summarizes the ideas in play. I'm looking forward to hearing about the approaches of the other teams.

Quote
Follow
Report

152 Upvoters
Comments 33

Sort by

Hotness

arrow_drop_down

undoredo
format_boldformat_italicformat_strikethrough
insert_link ❞ *code*
format_list_numberedformat_list_bulleted
table_chart

idle_speculationTopic Author • (1st in this Competition) • 5 years ago • Options •

Report • Reply
keyboard_arrow_up

10

@vtKMH, @Alpha

note:tryrefreshingthepageacoupletimesifthemathdoesn′tdisplaycorrectly

Let me try to explain the gradient descent part in more detail and let's agree that a unique user location u is a triple (user_location_country, user_location_region, user_location_city) and a unique hotel location h is a pair (srch_destination_id, hotel_cluster).

Now let's place the distinct user and hotel locations on the globe completely at random. So each user u_i will have a latitude and longitude $(\phi_{i,1},\phi_{i,2})$. Likewise, each hotel h_j has a latitude and longitude $(\theta_{j,1},\theta_{j,2})$. It's possible to write a down a formula which gives the distance along the surface of a sphere:

$$D(\phi_{i,1},\phi_{i,2},\theta_{j,1},\theta_{j,2})=r*\arccos(\sin\phi_{i,1}\sin\theta_{j,1}+\cos\phi_{i,1}\cos\theta_{j,1}\sin(\phi_{i,2}-\theta_{j,2}))$$

Where r is the radius of the sphere.
From the training set we collect all the combinations of (u_i,h_j,d_{ij}) where d_{ij} is the orig destination distance. Our goal is to adjust the placement of each u_i and h_j in order to make the computed distance D as close as possible to the actual distance d. One way to quantify this is to define a loss L: $$L=(D(\phi_{i,1},\phi_{i,2},\theta_{j,1},\theta_{j,2})-d_{ij})^2$$ Whatever placement of users and hotels minimizes this loss function should make our predicted distances very close to the actuals.

From calculus, you may recall the notion of the gradient. The gradient of \\(L\\), denoted \\(\nabla L\\), is just the vector formed from the partial derivatives of \\(L\\):$$\nabla L=(\frac{\partial L}{\partial \phi_{i,1}}, \frac{\partial L}{\partial \phi_{i,2}},\frac{\partial L}{\partial \theta_{i,1}}, \frac{\partial L}{\partial \theta_{i,2}})$$

For our purposes, the important fact is that the negative of the gradient points in the direction the loss is decreasing most rapidly.

If the gradient isn't sounding familiar, don't panic, there is an easy geometric description of what's going on. Since we have two points on a sphere, we can draw the geodesic great circle through them. If predicted distance along the great circle \\(D\\) is larger than the actual distance \\(d\\), then we can think of the negative gradient as a pair of vectors emanating from the user and hotel and pointing toward each other along the shorter arc of the great circle. Conversely, when \\(D\\) is smaller than \\(d\\) then the negative gradient is a pair of vectors pointing toward each other along the bigger arc of the great circle.

Either way you want to think about the gradient, taking the current position for our user-hotel pair and moving in the direction of the negative gradient should make our loss a little smaller. All that really happens in gradient descent is that we iterate through each triple \\((u_i,h_j,d_{ij})\\) and make a very small step from our current configuration in the direction of the negative gradient. After many, many iterations, the hope is that we end up with a configuration of users and hotels whose distances agree with the actual distances reasonably well.

idle_speculationTopic Author • (1st in this Competition) • 5 years ago • Options •

Report • Reply
keyboard_arrow_up

9

@aldente

1 Yes, the indicator is binary

2 Perhaps someone more familiar with the python interface can chime in, but that sounds right. One thing to note is that you'll also want your input data sorted by group.

3 I built the model on a dual socket E5-2699v3 with ~700GB ram and was only able to use about half the bookings after '2014-07-01' due to memory limitations. The training set was around 58 million rows and it took 38 hours to fit 1200 trees. Honestly though, it was a huge waste of electricity because there was virtually no improvement over using a training set 1/10th the size.

idle_speculationTopic Author • (1st in this Competition) • 5 years ago • Options •

Report • Reply
keyboard_arrow_up

5

@data_js

Yes, the features were one-hot encoded. Even user_id with its 1,198,786 distinct values. It might seem a little nutty to add a million columns to your data and then try fitting a linear model on it, but the "trick" to FM models is that they always project the one-hot encoding to some lower dimensional subspace. I used libFFM's default 4-dimensional space in this case.

idle_speculationTopic Author • (1st in this Competition) • 5 years ago • Options •

Report • Reply
keyboard_arrow_up

3

@Laurae

The loss distribution for gradient descent, in my experience, had a fat tail. L-BFGS probably suffers from the same issue.

It was possible to get much faster convergence by capping actual−expected term in the gradient. Unfortunately, capping too aggressively had a negative impact on the eventual quality of the fit. The compromise I came up with was to lower the cap very gradually, hence the long fit time.

I'm using C++ as my gradient descent library. The user interface leaves a lot to be desired, but the execution time is hard to beat.

Jun Shi • (70th in this Competition) • 5 years ago • Options •

Report • Reply
keyboard_arrow_up

4

Thanks for sharing your approach. I have a question regarding step #2, Factorization Machines.

Some of the variables have large cardinalities, for example, In the train set:

user_location_region 1008,
user_location_city 50447,
srch_destination_id 59455.

Did you one-hot encode all of them?

ZYE • (31st in this Competition) • 5 years ago • Options •

Report • Reply
keyboard_arrow_up

1

quote=vtKMH;123726

quote=idlespeculation;123433

For both H1 and H2, user and hotel locations were randomly initialized on a sphere and gradient descent was applied to the spherical law of cosines formula on the distinct combinations of U,H,origdestinationdistance.

/quote

Congratulations on a great solution!

I'm really interested in how this works, if you're inclined to share a little detail in your write-up.

My typical use of gradient descent just updates weights to minimize cost. But in this case, the requirement is to update weights, origin location AND destination location, and at the outset, I don't really understand how that works. I see papers like this... but that requires distances be known between EVERY location and to have absolute locations known for at least 3 points... and this problem met neither of those requirements.

Anyway... anything you're kind enough to share is tremendously appreciated, as this seems like an immensely practical tool to have tucked away.

Thanks and congratulations again!
kevin

/quote

My understanding is, gradient descent is applied to iteratively solve a set of equations - here the equations are governed by the spherical law of cosines among points on a sphere. The case that you mentioned using gradient descent to minimize cost e.g.,minc(x), is equivalently to solve a set of gradient equations c′(x=0) and gradient descent is serving same purpose.

happycube • (217th in this Competition) • 5 years ago • Options •

Report • Reply
keyboard_arrow_up

1

Amazing work, and thanks for sharing!

BTW the Amazon X1 instance is over twice as big than that Xeon box. *Iff* you need it to win a competition *this* powerfully and know it going in, it'd be worth every penny ;)

But usually if one has the skills, a less powerful machine is enough...

idle_speculationTopic Author • (1st in this Competition) • 5 years ago • Options •

Report • Reply
keyboard_arrow_up

2

@FengLi

The choice of the "rank:pairwise" algorithm was motivated by the evaluation metric. The metric belongs to a class of information retrieval metrics where learning-to-rank models such as "rank:pairwise" perform quite well. Had the metric been something else, multiclass logloss for instance, I would choose a different algorithm.

A similar approach can work for situations with thousands of classes. It depends on whether large number of classes can be rejected up-front. The ICDM 2015 is an example with millions of potential classes where similar techniques were successful.

Lawrence Chernin • (144th in this Competition) • 5 years ago • Options •

Report • Reply
keyboard_arrow_up

0

Who is the person of @idle_speculation ?

xiaojian • 2 years ago • Options •

Report • Reply
keyboard_arrow_up

0

Hi , I saw your message from a instructional video . He gave you a very good rating.
I don't know if you're going to reply to me , Because I know it's a lucky thing to accept the advice of a strong man.
I see you've won a lot of first place . I'm very interested in data science .

And I want to become a Data Scientists . Now my Learning route is probability theory , EDA and Visualization .But I think I lack that kind of data science thinking . I don't know how to cultivate this kind of thinking, because I can't find any information around me.
I look forward to your reply.

Mayank More • 3 years ago • Options •

Report • Reply
keyboard_arrow_up

0

@idle_speculation :

Since the factorization machines were built to exploit interactions between categorical variables, does that mean we should not train the model on numerical variables and instead always convert them to categorical variables as well?

Vincent Pham • 4 years ago • Options •

Report • Reply
keyboard_arrow_up

0

Thanks, explanation was very helpful

Vijay Krishnavanshi • (691st in this Competition) • 5 years ago • Options •

Report • Reply
keyboard_arrow_up

0

Thanks for the explanation :)

TomBiernacki • 5 years ago • Options •

Report • Reply
keyboard_arrow_up

0

Great job and congrats!! I am a newbie and learning so thanks for the details.

ec-ccs • 5 years ago • Options •

Report • Reply
keyboard_arrow_up

0

Wow, @idle_speculation:

amazing job...

your preparation and determination to reach a high rank in this project, the fact that you took the time to explain your project to everyone after wining it, and then donating part of your prize on Lucas' honour...

All are admirable...

Thanks and Success!

Pietro Marinelli • (62nd in this Competition) • 5 years ago • Options •

Report • Reply
keyboard_arrow_up

0

Congrats and most of all thanks for sharing!!

Walraaf • (29th in this Competition) • 5 years ago • Options •

Report • Reply
keyboard_arrow_up

0

Hi idle_speculation, grats on the giant win!
You asked for how other teams approached the distance matrix problem, so I thought I'd share what we did. We only started attempting this in the last week after I saw your score come in and I knew the suspicion I had of this being possible had to be true. We were unable to complete the approach, but what we did was attempt to identify unique hotels to improve the accuracy, by only selecting hotel cluster + destination combinations that always had consistent distances from all cities ietwodifferentdistancesfromonecity=morethanonehotel. Then we found sets of 2 unique hotels and 2 cities that had to be on a straight line, i.e. distance a + b + c = d. This let us accurately determine the distance between city a & b
we found cities & hotels that were apart by hundreds of miles while lying within a foot of a straight line. The main issue we then had was that even though we were using the WGS ellipsoid for our model of the earth instead of a sphere, we needed to know some initial coordinates, otherwise the curvature of the earth would always mess with our accuracy. I never got past this point, but looking at your

elegant solution now with gradient descent, I wonder whether it would've been possible to use that to initialize the location of the first few coordinates, and triangulate everything with high accuracy from there.

In any case, cheers for the great solution and thanks for the explanation.

idle_speculationTopic Author • (1st in this Competition) • 5 years ago • Options •

Report • Reply
keyboard_arrow_up

0

@Watts

By "burst" I just mean that each booking instance was repeated 100 times, once for each hotel cluster. This was done because learning-to-rank models have an additional group structure in which comparisons are made. In this case, the group consists of a single booking instance and the items to be ranked are the hotel clusters.

@CPMP

You're right the distance minimization problem is very far from convex. Even after all the parameter tuning, my final solution was definitely not the global minimum. One technique used to mitigate this problem, which was glossed over in the summary, was to run both versions of the matrix completion 18 times in parallel with different seeds. The average and minimum distances across the different seeds were what actually got used in the model.

CPMP • (57th in this Competition) • 5 years ago • Options •

Report • Reply
keyboard_arrow_up

0

@idle_speculation, adding kudos to all you received.

The distance minimization problem you describe isn't convex. Were you stuck in a local minima?

Ashish Lal • (1207th in this Competition) • 5 years ago • Options •

Report • Reply
keyboard_arrow_up

0

@idle_speculation

Congratulations!

Can u also explain how you used "bursting" to convert the problem into xgboost "rank:pairwise" problem. I am unable to understand that part.

Thanks!

Alpha • (230th in this Competition) • 5 years ago • Options •

Report • Reply
keyboard_arrow_up

0

Congrats for this winning and for your helping hand .... :-)
I want to learn more about distance matrix completion part ....
Any links to understand the theory and code of this part will be highly helpful ....
Was your complete intention was to hot encode everything other than numeric variables and then reduce the space into lower dimentional space and then using xgboost

Thank u very much for your help

vtKMH • (23rd in this Competition) • 5 years ago • Options •

Report • Reply
keyboard_arrow_up

0

quote=idlespeculation;123433

For both H1 and H2, user and hotel locations were randomly initialized on a sphere and gradient descent was applied to the spherical law of cosines formula on the distinct combinations of U,H,origdestinationdistance.

/quote

Congratulations on a great solution!

I'm really interested in how this works, if you're inclined to share a little detail in your write-up.

My typical use of gradient descent just updates weights to minimize cost. But in this case, the requirement is to update weights, origin location AND destination location, and at the outset, I don't really understand how that works. I see papers like this... but that requires distances be known between EVERY location and to have absolute locations known for at least 3 points... and this problem met neither of those requirements.

Anyway... anything you're kind enough to share is tremendously appreciated, as this seems like an immensely practical tool to have tucked away.

Thanks and congratulations again!
kevin

Alessandro Mariani • (643rd in this Competition) • 5 years ago • Options •

Report • Reply
keyboard_arrow_up

0

"Chapeau" for the solution, sharing and mostly donating! Fantastic solution :)

Davut Polat • (21st in this Competition) • 5 years ago • Options •

Report • Reply
keyboard_arrow_up

0

Thanks for the details,
this is an epic win! very well deserved position

FengLi • (151st in this Competition) • 5 years ago • Options •

Report • Reply
keyboard_arrow_up

0

@ idle_speculation Congratulations, thank you for sharing and I'm impressed with what you did to American Cancer Society in honor of Lucas.

For the solution, I have one question. I noticed you that you just made only one submission. How did you know that converting this problem into *rank:pairwise* problem will help you a lot? Is there any efficient way to solve the similar problem which has thousands of classes? Thank you.