Project: scikit-learn

2017-07-17T18:39:27Z yarikoptic 84comments

Debian test failures (was test\_preserve\_trustworthiness\_approximately fails on 32bit: AssertionError: 0.89166666666666661 not greater than 0.9)

Web url: https://github.com/scikit-learn/scikit-learn/issues/9393

API url: https://api.github.com/repos/scikit-learn/scikit-learn/issues/9393

building 0.19b2 on debian/ubuntus ... still ongoing but I see consistent failure on Debian stretch (nd90, current stable) and testing (nd100), 32bit only (ok on amd64 build):

```

neurodebian@smaug ~/deb/builds/scikit-learn/0.19~b2-1 % grep -5 AssertionError: \*build

scikit-learn\_0.19~b2-1~nd100+1\_i386.build-Traceback (most recent call last):

scikit-learn\_0.19~b2-1~nd100+1\_i386.build- File "/usr/lib/python2.7/dist-packages/nose/case.py", line 197, in runTest

scikit-learn\_0.19~b2-1~nd100+1\_i386.build- self.test(\*self.arg)

scikit-learn\_0.19~b2-1~nd100+1\_i386.build- File "/build/scikit-learn-0.19~b2/debian/tmp/usr/lib/python2.7/dist-packages/sklearn/manifold/tests/test\_t\_sne.py", line 247, in test\_preserve\_trustworthiness\_approximately

scikit-learn\_0.19~b2-1~nd100+1\_i386.build- assert\_greater(t, 0.9)

scikit-learn\_0.19~b2-1~nd100+1\_i386.build:AssertionError: 0.89166666666666661 not greater than 0.9

scikit-learn\_0.19~b2-1~nd100+1\_i386.build-

scikit-learn\_0.19~b2-1~nd100+1\_i386.build-----------------------------------------------------------------------

scikit-learn\_0.19~b2-1~nd100+1\_i386.build-Ran 7969 tests in 285.883s

scikit-learn\_0.19~b2-1~nd100+1\_i386.build-

scikit-learn\_0.19~b2-1~nd100+1\_i386.build-FAILED (SKIP=73, failures=1)

--

scikit-learn\_0.19~b2-1~nd90+1\_i386.build-Traceback (most recent call last):

scikit-learn\_0.19~b2-1~nd90+1\_i386.build- File "/usr/lib/python2.7/dist-packages/nose/case.py", line 197, in runTest

scikit-learn\_0.19~b2-1~nd90+1\_i386.build- self.test(\*self.arg)

scikit-learn\_0.19~b2-1~nd90+1\_i386.build- File "/build/scikit-learn-0.19~b2/debian/tmp/usr/lib/python2.7/dist-packages/sklearn/manifold/tests/test\_t\_sne.py", line 247, in test\_preserve\_trustworthiness\_approximately

scikit-learn\_0.19~b2-1~nd90+1\_i386.build- assert\_greater(t, 0.9)

scikit-learn\_0.19~b2-1~nd90+1\_i386.build:AssertionError: 0.89166666666666661 not greater than 0.9

scikit-learn\_0.19~b2-1~nd90+1\_i386.build-

scikit-learn\_0.19~b2-1~nd90+1\_i386.build-----------------------------------------------------------------------

scikit-learn\_0.19~b2-1~nd90+1\_i386.build-Ran 7969 tests in 288.113s

scikit-learn\_0.19~b2-1~nd90+1\_i386.build-

scikit-learn\_0.19~b2-1~nd90+1\_i386.build-FAILED (SKIP=73, failures=1)

```

in both cases python-numpy is `1:1.12.1-3` (i.e. 1.12.1 numpy) and passed ok with numpy 1.8.2 in Debian jessie.

-------------------------------------------------------------------------

2017-07-18T16:04:47Z amueller

ping @ogrisel?

-------------------------------------------------------------------------

2017-07-19T07:50:04Z ogrisel

Interesting, it's a only on a combo of numpy 1.12.1 and 32 bit python...

Those tests pass with 32 bit python and numpy 1.13.1 on our wheel building travis:

https://travis-ci.org/MacPython/scikit-learn-wheels

@tomMoral if you want to play with docker, this is a good opportunity ;)

-------------------------------------------------------------------------

2017-07-19T08:19:15Z tomMoral

@ogrisel I will give it a try :)

-------------------------------------------------------------------------

2017-07-19T09:29:13Z tomMoral

@yarikoptic I am unable to reproduce the failure on 32bit debian `strech` on docker.

I tried installing python and scikit dependency using `apt install` and the test passed for both `python2.7.13` and `python3.5.3`.

Do you have a specific configuration that could explain the difference?

-------------------------------------------------------------------------

2017-07-24T09:27:53Z ogrisel

@yarikoptic @tomMoral how can you install numpy 1.12.1 on debian stretch? Which repo did you use to produce this failure?

-------------------------------------------------------------------------

2017-07-25T08:19:51Z tomMoral

@ogrisel I installed the `python-numpy` package, which uses version 1.12.1 and used branch `0.19.b2` for `scikit`.

EDIT: I used this docker image: `mcandre/docker-debian-32bit:stretch`

-------------------------------------------------------------------------

2017-07-25T11:39:42Z ogrisel

Indeed I was using an older image (jessie). I confirm I cannot reproduce the issue on stretch with the following 32 bit image: `jhsu802701/32bit-debian-stretch-min`.

-------------------------------------------------------------------------

2017-07-30T05:05:10Z jnothman

@yarikoptic, we would like to release. We need a way to reproduce the error, or we will need to skip the tests / lower the condition on certain architectures.

-------------------------------------------------------------------------

2017-08-12T15:54:10Z yarikoptic

oh -- I have managed to miss your message @jnothman and 0.13.0 came out without the fix, my bad. I will release debian packages as is (without i386 build for some) and later give you exact instruction on how to reproduce.

-------------------------------------------------------------------------

2017-08-12T16:05:46Z yarikoptic

FWIW -- issue in general reproducible: https://buildd.debian.org/status/package.php?p=scikit-learn&suite=experimental (check i386 build) and some other builds have other issues btw (that was in beta -- let's wait for current release to get built in unstable before summarizing coming up issues on some other architectures)

-------------------------------------------------------------------------

2017-08-13T14:01:13Z jnothman

Right. I see in the logs there an alarming number of fails for a final release :(((

And none of them are about `test\_preserve\_trustworthiness\_approximately`

```

======================================================================

FAIL: sklearn.feature\_extraction.tests.test\_feature\_hasher.test\_hasher\_alternate\_sign

----------------------------------------------------------------------

Traceback (most recent call last):

File "/usr/lib/python2.7/dist-packages/nose/case.py", line 197, in runTest

self.test(\*self.arg)

File "/<<PKGBUILDDIR>>/debian/tmp/usr/lib/python2.7/dist-packages/sklearn/utils/testing.py", line 291, in wrapper

return fn(\*args, \*\*kwargs)

File "/<<PKGBUILDDIR>>/debian/tmp/usr/lib/python2.7/dist-packages/sklearn/feature\_extraction/tests/test\_feature\_hasher.py", line 122, in test\_hasher\_alternate\_sign

assert\_true(len(Xt.data) < len(X[0]))

AssertionError: False is not true

----------------------------------------------------------------------

======================================================================

FAIL: sklearn.feature\_selection.tests.test\_from\_model.test\_feature\_importances

----------------------------------------------------------------------

Traceback (most recent call last):

File "/usr/lib/python2.7/dist-packages/nose/case.py", line 197, in runTest

self.test(\*self.arg)

File "/<<PKGBUILDDIR>>/debian/tmp/usr/lib/python2.7/dist-packages/sklearn/utils/testing.py", line 667, in run\_test

return func(\*args, \*\*kwargs)

File "/<<PKGBUILDDIR>>/debian/tmp/usr/lib/python2.7/dist-packages/sklearn/feature\_selection/tests/test\_from\_model.py", line 72, in test\_feature\_importances

assert\_almost\_equal(importances, importances\_bis, decimal=4)

File "/usr/lib/python2.7/dist-packages/numpy/testing/utils.py", line 573, in assert\_almost\_equal

return assert\_array\_almost\_equal(actual, desired, decimal, err\_msg)

File "/usr/lib/python2.7/dist-packages/numpy/testing/utils.py", line 979, in assert\_array\_almost\_equal

precision=decimal)

File "/usr/lib/python2.7/dist-packages/numpy/testing/utils.py", line 796, in assert\_array\_compare

raise AssertionError(msg)

AssertionError:

Arrays are not almost equal to 4 decimals

(mismatch 70.0%)

x: array([ 0.1537, 0.2294, 0.1825, 0.0667, 0.0485, 0.0587, 0.0643,

0.0642, 0.066 , 0.066 ])

y: array([ 0.1527, 0.2294, 0.1822, 0.0675, 0.0483, 0.0587, 0.0648,

0.0642, 0.0656, 0.0665])

----------------------------------------------------------------------

ON HURD-I386:

======================================================================

FAIL: sklearn.tests.test\_multioutput.test\_multi\_output\_classification\_partial\_fit\_parallelism

----------------------------------------------------------------------

Traceback (most recent call last):

File "/usr/lib/python2.7/dist-packages/nose/case.py", line 197, in runTest

self.test(\*self.arg)

File "/<<PKGBUILDDIR>>/debian/tmp/usr/lib/python2.7/dist-packages/sklearn/tests/test\_multioutput.py", line 171, in test\_multi\_output\_classification\_partial\_fit\_parallelism

assert\_false(est1 is est2)

AssertionError: True is not false

```

-------------------------------------------------------------------------

2017-08-13T15:46:37Z amueller

The last is the most confusing to me tbh.

-------------------------------------------------------------------------

2017-08-13T19:36:57Z yarikoptic

yeah, that 32bit issue didn't reproduce in current build. I guess it is not fully deterministic... will try to reproduce now "locally"

-------------------------------------------------------------------------

2017-08-13T23:10:56Z jnothman

and do we need to fix the other test failures for scikit-learn 0.19 to ship

with Debian?

On 14 Aug 2017 5:36 am, "Yaroslav Halchenko" <notifications@github.com>

wrote:

> yeah, that 32bit issue didn't reproduce in current build. I guess it is

> not fully deterministic... will try to reproduce now "locally"

>

> ‚Äî

> You are receiving this because you were mentioned.

> Reply to this email directly, view it on GitHub

> <https://github.com/scikit-learn/scikit-learn/issues/9393#issuecomment-322062607>,

> or mute the thread

> <https://github.com/notifications/unsubscribe-auth/AAEz6\_Vl2p-YlYcXSQOpCp6hl1B8UufNks5sX1BbgaJpZM4OaXom>

> .

>

-------------------------------------------------------------------------

2017-08-14T00:23:08Z jnothman

That last failure is not confusing after a little investigation. It's a result of `n\_jobs=-1` being the same as `n\_jobs=1` on a machine with 1 core.

-------------------------------------------------------------------------

2017-08-14T00:28:42Z jnothman

@rth, do you mind looking into the `test\_hasher\_alternate\_sign` failure above?

-------------------------------------------------------------------------

2017-08-14T08:33:04Z rth

> FWIW -- issue in general reproducible: https://buildd.debian.org/status/package.php?p=scikit-learn&suite=experimental (check i386 build)

@yarikoptic On this link I see "No entry in i386 database, check Packages-arch-specific" (with "Suite: experimental"). Is there another way of getting this i386 build for debian? or did I miss something? Thanks.

-------------------------------------------------------------------------

2017-08-14T08:48:16Z jnothman

yeah, i clicked the logs column after failing to work it out

On 14 Aug 2017 6:33 pm, "Roman Yurchak" <notifications@github.com> wrote:

> FWIW -- issue in general reproducible: https://buildd.debian.org/

> status/package.php?p=scikit-learn&suite=experimental (check i386 build)

>

> @yarikoptic <https://github.com/yarikoptic> On this link I see "No entry

> in i386 database, check Packages-arch-specific" (with "Suite:

> experimental"). Is there another way of getting this i386 build for debian?

> or did I miss something? Thanks.

>

> ‚Äî

> You are receiving this because you were mentioned.

> Reply to this email directly, view it on GitHub

> <https://github.com/scikit-learn/scikit-learn/issues/9393#issuecomment-322130322>,

> or mute the thread

> <https://github.com/notifications/unsubscribe-auth/AAEz6278vIWQxyGIzPkqRYHZohTGvgfyks5sYAZCgaJpZM4OaXom>

> .

>

-------------------------------------------------------------------------

2017-08-14T14:27:16Z yarikoptic

Thank you very much for looking into those!

FWIW -- locally I had only the test\_multi\_output\_classification\_partial\_fit\_parallelism to popup and indeed it was due to inability to do multiprocessing in my case (absent bound to /dev/shm I guess):

```

/usr/lib/python2.7/dist-packages/joblib/\_multiprocessing\_helpers.py:28: UserWarning: [Errno 13] Permission denied. joblib will operate in serial mode

```

it passes just fine when I have /dev/shm mounted and joblib does not complaint.

Re

> and do we need to fix the other test failures for scikit-learn 0.19 to ship with Debian?

well -- for unstable Debian -- no. But if we want to have it propagate into testing and thus become a part of the next Debian stable release (Whenever that would be) -- yes, should get addressed one way (fixed) or another (disabled)

-------------------------------------------------------------------------

2017-08-14T14:59:34Z yarikoptic

re original `test\_preserve\_trustworthiness\_approximately`

- specific to older versions of something (yet to figure out since numpy as nscipy are the same) since is not reproducible on current debian sid but reproducible on testing (from few days back) and other older releases.

- specific to init='pca' method='exact'

before I spend more time -- is there specific meaning for the threshold to be 0.9? may be it could just be relaxed a little? ;)

-------------------------------------------------------------------------

2017-08-14T15:02:26Z yarikoptic

btw these are the values I see for t, init, method

```

(0.95874999999999999, 'random', 'exact')

(0.96958333333333335, 'random', 'barnes\_hut')

(0.89166666666666661, 'pca', 'exact')

(0.95291666666666663, 'pca', 'barnes\_hut')

```

-------------------------------------------------------------------------

2017-08-14T23:53:39Z jnothman

i think .9 is somewhat arbitrary but we'd like to be sure that the

variation isn't pointing to something more sinister

On 15 Aug 2017 1:03 am, "Yaroslav Halchenko" <notifications@github.com>

wrote:

> btw these are the values I see for t, init, method

>

> (0.95874999999999999, 'random', 'exact')

> (0.96958333333333335, 'random', 'barnes\_hut')

> (0.89166666666666661, 'pca', 'exact')

> (0.95291666666666663, 'pca', 'barnes\_hut')

>

> ‚Äî

> You are receiving this because you were mentioned.

> Reply to this email directly, view it on GitHub

> <https://github.com/scikit-learn/scikit-learn/issues/9393#issuecomment-322214890>,

> or mute the thread

> <https://github.com/notifications/unsubscribe-auth/AAEz6wXVP59e2g1QtDqtkzOWd39xyPGNks5sYGGFgaJpZM4OaXom>

> .

>

-------------------------------------------------------------------------

2017-08-18T03:33:36Z yarikoptic

@jnothman how could we discover? ;)

-------------------------------------------------------------------------

2017-08-19T11:09:47Z jnothman

By pinpointing where this and other machines diverge in their calculation... Not that that's easy to do without at least a VM of the target machine.

-------------------------------------------------------------------------

2017-08-21T13:48:41Z jnothman

@rth could the `assert\_true(len(Xt.data) < len(X[0]))` failure be because of floating point error (i.e. a small number was in Xt.data instead of 0, and so was not removed)?

-------------------------------------------------------------------------

2017-08-21T13:52:26Z jnothman

@amueller, `test\_from\_model.test\_feature\_importances` already is marked with `skip\_if\_32bit` suggesting perhaps that this assertion is brittle. The test is failing where the importances in a model are being asserted identical to the importance in a similar model trained with sample\_weight=3\*orig\_weights. Any ideas how to fix here?

-------------------------------------------------------------------------

2017-08-21T21:29:13Z rth

@jnothman So far I am not able to reproduce the `test\_hasher\_alternate\_sign` failure above. Tried to build scikit-learn 0.19.2 it in a Debian sid/unstable i386 VM, were scipy and numpy 1.2.1 were installed with apt-get. I consistently get the failure about `test\_multi\_output\_classification\_partial\_fit\_parallelism` (that was resolved since as far as I understand) but not the one about hashing.

> could the assert\_true(len(Xt.data) < len(X[0])) failure be because of floating point error (i.e. a small number was in Xt.data instead of 0, and so was not removed)?

I don't think it's due to floating point error. So the test fails on [this line](https://github.com/scikit-learn/scikit-learn/blob/master/sklearn/feature\_extraction/tests/test\_feature\_hasher.py#L122), where the expected values are `len(Xt.data) == 4` and `len(X[0])==5` (and I get those in the 32bit VM as well).

This test assumes that the hash value of the tested tokens always produces the same results (in which case two of those produce a collision). And it looks like mumurhash3 [doesn't actually produce the same results in 64bit and 32bit](https://github.com/scikit-learn/scikit-learn/blob/master/sklearn/utils/src/MurmurHash3.cpp#L5), which would explain why this test fail. This doesn't explain why I can't reproduce it though.

Since this basically tests that in `FeatureHasher` we can disable the `alternate\_sign` functionality (enabled by default) and it doesn't validate any new functionality, it might be OK to skip it on failure on 32 bit? What do you think?

-------------------------------------------------------------------------

2017-08-21T23:58:40Z jnothman

Ha! I had no idea it worked differently on 64-bit and 32-bit... strange

choice of hash function :\

I'm okay with @skip\_if\_32bit here. Not quite satisfying as we don't

understand what's going on...

Is this testing a collision where the sign alternates and hence the value

lands up at 0? Or just testing that values are stored in the same spot due

to collision?

On 22 August 2017 at 07:29, Roman Yurchak <notifications@github.com> wrote:

> @jnothman <https://github.com/jnothman> So far I am not able to reproduce

> the test\_hasher\_alternate\_sign failure above. Tried to build scikit-learn

> 0.19.2 it in a Debian sid/unstable i386 VM, were scipy and numpy 1.2.1 were

> installed with apt-get. I consistently get the failure about

> test\_multi\_output\_classification\_partial\_fit\_parallelism (that was

> resolved since as far as I understand) but not the one about hashing.

>

> could the assert\_true(len(Xt.data) < len(X[0])) failure be because of

> floating point error (i.e. a small number was in Xt.data instead of 0, and

> so was not removed)?

>

> I don't think it's due to floating point error. So the test fails on this

> line

> <https://github.com/scikit-learn/scikit-learn/blob/master/sklearn/feature\_extraction/tests/test\_feature\_hasher.py#L122>,

> where the expected values are len(Xt.data) == 4 and len(X[0])==5 (and I

> get those in the 32bit VM as well).

>

> This test assumes that the hash value of the tested tokens always produces

> the same results (in which case two of those produce a collision). And it

> looks like mumurhash3 doesn't actually produce the same results in 64bit

> and 32bit

> <https://github.com/scikit-learn/scikit-learn/blob/master/sklearn/utils/src/MurmurHash3.cpp#L5>,

> which would explain why this test fail. This doesn't explain why I can't

> reproduce it though.

>

> Since this basically tests that in FeatureHasher we can disable the

> alternate\_sign functionality (enabled by default) and it doesn't validate

> any new functionality, it might be OK to skip it on failure on 32 bit? What

> do you think?

>

> ‚Äî

> You are receiving this because you were mentioned.

> Reply to this email directly, view it on GitHub

> <https://github.com/scikit-learn/scikit-learn/issues/9393#issuecomment-323857844>,

> or mute the thread

> <https://github.com/notifications/unsubscribe-auth/AAEz6-5e4\_hYRT3FldNHOz-N6WTl8wMFks5safasgaJpZM4OaXom>

> .

>

-------------------------------------------------------------------------

2017-08-22T08:17:09Z rth

> Is this testing a collision where the sign alternates and hence the value

lands up at 0? Or just testing that values are stored in the same spot due

to collision?

The former I think. Despite that comment in murmurhash3, I'm not sure the hash value is actually platform dependent: after all this test passes on Appveyor 32bit and 64bit (and it works fine for me on i386) . But it does seem like a plausible suspect.

We could try to make this test more robust, by just taking a large number of tokens N, hashing them with a hash table size = 1 (or any small number), and checking that with `alternate\_sign=False` the sum of hashed values is equal to `N`, and that it's strictly lower than `N` if `alternate\_sign=True` (since some +1 / -1 are bound to cancel out if N is large enough). That would be less dependent on the actual hashing implementation...

Still, for a Debian release of 0.19.0 I'm not sure how this could work: can you apply some patches on the original .tar.gz to skip tests/modify code when needed?

-------------------------------------------------------------------------

2017-08-22T09:23:27Z jnothman

we're going to release a bug-fix release in any case. i had wondered if

finding and data close to 0 would help here.

On 22 Aug 2017 6:17 pm, "Roman Yurchak" <notifications@github.com> wrote:

> Is this testing a collision where the sign alternates and hence the value

> lands up at 0? Or just testing that values are stored in the same spot due

> to collision?

>

> The former I think. Despite that comment in murmurhash3, I'm not sure the

> hash value is actually platform dependent: after all this test passes on

> Appveyor 32bit and 64bit (and it works fine for me on i386) . But it does

> seem like a plausible suspect.

>

> We could try to make this test more robust, by just taking a large number

> of tokens N, hashing them with a hash table size = 1 (or any small number),

> and checking that with alternate\_sign=False the sum of hashed values is

> equal to N, and that it's strictly lower than N if alternate\_sign=True

> (since some +1 / -1 are bound to cancel out if N is large enough). That

> would be less dependent on the actual hashing implementation...

>

> Still, for a Debian release of 0.19.0 I'm not sure how this could work:

> can you apply some patches on the original .tar.gz to skip tests/modify

> code when needed?

>

> ‚Äî

> You are receiving this because you were mentioned.

> Reply to this email directly, view it on GitHub

> <https://github.com/scikit-learn/scikit-learn/issues/9393#issuecomment-323953661>,

> or mute the thread

> <https://github.com/notifications/unsubscribe-auth/AAEz67eLrHeac5uXJQZxOWeIGhUxLyHjks5sao6IgaJpZM4OaXom>

> .

>

-------------------------------------------------------------------------

2017-08-22T09:44:49Z rth

On 22/08/17 12:23, Joel Nothman wrote:

> we're going to release a bug-fix release in any case. i had wondered if

> finding and data close to 0 would help here.

Right, I don't think the value of zero matters. It could be a +1 - 1 = 0

or a +3 - 2 = 1 (instead of 3+2=5) as long the value in the hash bucket

is lower than the sum of the absolute value of hashed terms, it is

sufficient to determine whether `alternate\_sign` is used or not during

the hash collisions, I think..

-------------------------------------------------------------------------

2017-08-22T10:45:48Z jnothman

Yes, I think you're right. We're not calling eliminate\_zeros anywhere.

On 22 August 2017 at 19:44, Roman Yurchak <notifications@github.com> wrote:

> On 22/08/17 12:23, Joel Nothman wrote:

> > we're going to release a bug-fix release in any case. i had wondered if

> > finding and data close to 0 would help here.

>

> Right, I don't think the value of zero matters. It could be a +1 - 1 = 0

> or a +3 - 2 = 1 (instead of 3+2=5) as long the value in the hash bucket

> is lower than the sum of the absolute value of hashed terms, it is

> sufficient to determine whether `alternate\_sign` is used or not during

> the hash collisions, I think..

>

> ‚Äî

> You are receiving this because you were mentioned.

> Reply to this email directly, view it on GitHub

> <https://github.com/scikit-learn/scikit-learn/issues/9393#issuecomment-323975906>,

> or mute the thread

> <https://github.com/notifications/unsubscribe-auth/AAEz65HpRe\_Nz871uQCrz6W7U13qpUgFks5saqMTgaJpZM4OaXom>

> .

>

-------------------------------------------------------------------------

2017-09-01T10:40:36Z priidukull

I can confirm that test\_preserve\_trustworthiness\_approximately also failed on a 64 bit Mac.

Error message:

AssertionError: 0.89166666666666661 not greater than 0.9

2,4 GHz Intel Core i5

8 GB 1600 MHz DDR3

Python 3.6.1

numpy 1.13.1

scikit-learn master branch, last commit hash d6a42354145c92cf88093cbcc70b13f639319c38

numpy was installed from pip, so this is with Accelerate.

OSX version 10.12.4

-------------------------------------------------------------------------

2017-09-01T13:43:30Z lesteve

FYI, I can not reproduce on my OSX version with the same numpy version, Accelerate as well.

-------------------------------------------------------------------------

2017-09-06T15:59:55Z ogrisel

I also tried on macOS (El Capitan) with Accelerate and could not reproduce either.

-------------------------------------------------------------------------

2017-09-06T16:06:15Z ogrisel

> That last failure is not confusing after a little investigation. It's a result of n\_jobs=-1 being the same as n\_jobs=1 on a machine with 1 core.

We probably need to raise a `SkipTest` if `multiprocessing.cpu\_count() == 1`.

-------------------------------------------------------------------------

2017-09-07T01:03:52Z jnothman

The n\_jobs=1 issue has been fixed.

We appear to have the following issues:

\* [x] `test\_hasher\_alternate\_sign`: [mips](https://buildd.debian.org/status/fetch.php?pkg=scikit-learn&arch=mips&ver=0.19.0-1&stamp=1502636171&raw=0), [powerpc](https://buildd.debian.org/status/fetch.php?pkg=scikit-learn&arch=powerpc&ver=0.19.0-1&stamp=1502556098&raw=0), [hppa](https://buildd.debian.org/status/fetch.php?pkg=scikit-learn&arch=hppa&ver=0.19.0-1&stamp=1502574224&raw=0), [ppc64](https://buildd.debian.org/status/fetch.php?pkg=scikit-learn&arch=ppc64&ver=0.19.0-1&stamp=1502556652&raw=0), [s390x](https://buildd.debian.org/status/fetch.php?pkg=scikit-learn&arch=s390x&ver=0.19.0-1&stamp=1502555384&raw=0), [sparc64](https://buildd.debian.org/status/fetch.php?pkg=scikit-learn&arch=sparc64&ver=0.19.0-1&stamp=1502575352&raw=0) fixed in #9710

\* [x] `test\_feature\_importances` [arm64](https://buildd.debian.org/status/fetch.php?pkg=scikit-learn&arch=arm64&ver=0.19.0-1&stamp=1502557273&raw=0), [ppc64](https://buildd.debian.org/status/fetch.php?pkg=scikit-learn&arch=ppc64&ver=0.19.0-1&stamp=1502556652&raw=0), [ppc64el](https://buildd.debian.org/status/fetch.php?pkg=scikit-learn&arch=ppc64el&ver=0.19.0-1&stamp=1502557902&raw=0), [s390x](https://buildd.debian.org/status/fetch.php?pkg=scikit-learn&arch=s390x&ver=0.19.0-1&stamp=1502555384&raw=0). PR in #9733

\* [x] `test\_preserve\_trustworthiness\_approximately` PR in #9808

\* [ ] `test\_forest.test\_parallel` (not listed above): [mips](https://buildd.debian.org/status/fetch.php?pkg=scikit-learn&arch=mips&ver=0.19.0-1&stamp=1502636171&raw=0). PR in ~#9734~ #9830

\* [x] `test\_multi\_output\_classification\_partial\_fit\_parallelism` (fixed in #9544)

-------------------------------------------------------------------------

2017-09-07T01:05:37Z jnothman

The original issue with `test\_preserve\_trustworthiness\_approximately` remains the most concerning, IMO.

-------------------------------------------------------------------------

2017-09-08T18:58:05Z amueller

Hm none of the links at ``test\_preserve\_trustworthyness\_approximately`` above have failures for that, right? Or I'm blind.

-------------------------------------------------------------------------

2017-09-08T19:00:17Z amueller

have we seen this before:

```

sklearn.metrics.tests.test\_pairwise.test\_pairwise\_parallel(<function pairwise\_distances at 0x81153b7b8>, <function wminkowski at 0x80f478e18>, {'w': array([ 1., 2., 3., 4.]), 'p': 1}) ... E: Caught signal ‚ÄòTerminated‚Äô: terminating immediately

```

from [kfreebsd-amd64](https://buildd.debian.org/status/fetch.php?pkg=scikit-learn&arch=kfreebsd-amd64&ver=0.19.0-1%2Bb1&stamp=1504884720&raw=0)

-------------------------------------------------------------------------

2017-09-09T11:59:46Z rth

> have you tried finding a docker to reproduce somehow? (from https://github.com/scikit-learn/scikit-learn/pull/9710)

@jnothman I just tried, but I'm not able to run e.g. a ppc64 Docker image on my amd64 system. With the `Dockerfile` below I get an error,

```

standard\_init\_linux.go:178: exec user process caused "exec format error"

```

at the first `apt-get` suggesting there is [a platform conflict](https://stackoverflow.com/a/42886112/1791279). Using `FROM debian` as the first line this works fine for amd64. So unless I missed something it doesn't look like this could be reproducible in Docker. Will need to find a VM image instead...

\*\*Dockerfile\*\*

```

FROM ppc64le/debian:unstable-20170907

RUN apt-get update && apt-get install -y gcc g++ git make wget bzip2

RUN wget http://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86\_64.sh -O miniconda.sh && \

chmod +x miniconda.sh && \

./miniconda.sh -b

ENV PATH "/root/miniconda3/bin/:${PATH}"

RUN conda create -n sklearn-env numpy scipy cython pytest nose docutils python=3.6

# switch from dash to bash to make "source activate" work

RUN echo "dash dash/sh boolean false" | debconf-set-selections && \

DEBIAN\_FRONTEND=noninteractive dpkg-reconfigure dash

RUN source activate sklearn-env && \

git clone https://github.com/scikit-learn/scikit-learn.git && \

cd scikit-learn && git checkout 0.19.0

RUN cd scikit-learn && source activate sklearn-env && \

make

```

built with

```

docker build -t sklearn-019-debian-ppc64 .

```

-------------------------------------------------------------------------

2017-09-09T13:42:33Z rth

Actually, the above Docker setup with conda wouldn't have worked anyway for other platforms, it should have been, something along the lines of, I think,

```

FROM ppc64le/debian:unstable-20170907

RUN apt-get update && apt-get install -y build-essential git make

RUN apt-get install -y python3 python3-scipy python3-numpy python3-setuptools python3-pkg-resources python3-pip

RUN git clone https://github.com/scikit-learn/scikit-learn.git && \

cd scikit-learn && git checkout 0.19.0

RUN pip3 install Cython nose pytest

RUN cd scikit-learn && python3 setup.py develop

RUN pytest -sv scikit-learn/sklearn

```

but this still wouldn't help unless someone has access to non amd64 platforms and is able to run it there, using the [appropriate Docker Debian image](https://github.com/docker-library/official-images#architectures-other-than-amd64) ...

-------------------------------------------------------------------------

2017-09-09T23:16:22Z jnothman

@rth, okay. Thanks for trying.

> Hm none of the links at test\_preserve\_trustworthyness\_approximately above have failures for that, right? Or I'm blind.

No, I must have sorted these things incorrectly. @yarikoptic, I can't find the `test\_preserve\_trustworthiness\_approximately` failure under 0.19.0-1 logs.

-------------------------------------------------------------------------

2017-09-10T23:45:36Z jnothman

@yarikoptic, any suggestion of how we can reproduce these test environments?

-------------------------------------------------------------------------

2017-09-11T14:11:34Z amueller

@jnothman any ideas about the ``test\_pairwise\_parallel`` failure?

-------------------------------------------------------------------------

2017-09-11T21:42:39Z jnothman

test\_pairwise\_parallel I had missed, but I also suspect it's something we'll find impossible to debug... Terminated after 150 minutes of inactivity during parallel execution of a simple function

-------------------------------------------------------------------------

2017-09-12T00:43:42Z jnothman

I'm guessing `test\_forest.test\_parallel` has failed because of precision errors due to partitioning the ensemble summation across jobs. I'll submit a PR to reduce precision of the test.

-------------------------------------------------------------------------

2017-09-12T01:04:49Z jnothman

@priidukull is your test failure reproducible? Could you help us debug? Which `init` and `method` combination is the first to fail?

-------------------------------------------------------------------------

2017-09-12T05:21:52Z priidukull

```

method = 'barnes\_hut'

init = 'pca'

```

What I've done is to reduce the size of X... with something like:

`X = random\_state.randn(3, n\_components).astype(np.float32)`

And then I debugged through the code on both of my environments and the best I could tell was that the divergence happened in C-code. But I could not tell where exactly with full certainty because it is tough to debug.

```

def test\_preserve\_trustworthiness\_approximately():

# Nearest neighbors should be preserved approximately.

random\_state = check\_random\_state(0)

n\_components = 2

method = 'barnes\_hut'

X = random\_state.randn(50, n\_components).astype(np.float32)

init = 'pca'

tsne = TSNE(n\_components=n\_components, init=init, random\_state=0, method=method)

X\_embedded = tsne.fit\_transform(X)

t = trustworthiness(X, X\_embedded, n\_neighbors=1)

assert\_greater(t, 0.9)

```

-------------------------------------------------------------------------

2017-09-12T05:55:55Z jnothman

What do you mean by the divergence? What were you comparing it against? The different methods and inits produce different trustworthiness scores on all platforms.

-------------------------------------------------------------------------

2017-09-12T07:23:51Z jnothman

@priidukull, could you please provide the output of:

```

import numpy as np

from sklearn.manifold.t\_sne import TSNE, trustworthiness

from sklearn.utils import check\_random\_state

random\_state = check\_random\_state(0)

X = random\_state.randn(50, 2).astype(np.float32)

tsne = TSNE(n\_components=2, init='pca', random\_state=0,

method='barnes\_hut', verbose=10)

X\_embedded = tsne.fit\_transform(X)

t = trustworthiness(X, X\_embedded, n\_neighbors=1)

assert t > 0.9

```

and of:

```

import numpy as np

from sklearn.manifold.t\_sne import TSNE, trustworthiness

from sklearn.utils import check\_random\_state

random\_state = check\_random\_state(0)

X = random\_state.randn(50, 2).astype(np.float32)

for n\_iter in range(250, 1001, 25):

tsne = TSNE(n\_components=2, init='pca', random\_state=0,

method='barnes\_hut', n\_iter=n\_iter)

X\_embedded = tsne.fit\_transform(X)

t = trustworthiness(X, X\_embedded, n\_neighbors=1)

print(n\_iter, t)

```

Thanks.

For reference, I have:

```

[t-SNE] Computing 49 nearest neighbors...

[t-SNE] Indexed 50 samples in 0.000s...

[t-SNE] Computed neighbors for 50 samples in 0.001s...

[t-SNE] Computed conditional probabilities for sample 50 / 50

[t-SNE] Mean sigma: 1.228846

[t-SNE] Computed conditional probabilities in 0.002s

[t-SNE] Iteration 50: error = 50.3999405, gradient norm = 0.4808916 (50 iterations in 0.023s)

[t-SNE] Iteration 100: error = 51.3690414, gradient norm = 0.4514403 (50 iterations in 0.022s)

[t-SNE] Iteration 150: error = 49.3410950, gradient norm = 0.5894858 (50 iterations in 0.023s)

[t-SNE] Iteration 200: error = 47.4731636, gradient norm = 0.5425554 (50 iterations in 0.025s)

[t-SNE] Iteration 250: error = 49.5253944, gradient norm = 0.4418719 (50 iterations in 0.028s)

[t-SNE] KL divergence after 250 iterations with early exaggeration: 49.525394

[t-SNE] Iteration 300: error = 0.9909145, gradient norm = 0.0019572 (50 iterations in 0.027s)

[t-SNE] Iteration 350: error = 0.7841685, gradient norm = 0.0009371 (50 iterations in 0.024s)

[t-SNE] Iteration 400: error = 0.6556321, gradient norm = 0.0009947 (50 iterations in 0.029s)

[t-SNE] Iteration 450: error = 0.4789825, gradient norm = 0.0010353 (50 iterations in 0.022s)

[t-SNE] Iteration 500: error = 0.3935374, gradient norm = 0.0004297 (50 iterations in 0.028s)

[t-SNE] Iteration 550: error = 0.3284135, gradient norm = 0.0005897 (50 iterations in 0.024s)

[t-SNE] Iteration 600: error = 0.2722271, gradient norm = 0.0001759 (50 iterations in 0.027s)

[t-SNE] Iteration 650: error = 0.2614944, gradient norm = 0.0001755 (50 iterations in 0.023s)

[t-SNE] Iteration 700: error = 0.2059281, gradient norm = 0.0004730 (50 iterations in 0.029s)

[t-SNE] Iteration 750: error = 0.1852958, gradient norm = 0.0001332 (50 iterations in 0.038s)

[t-SNE] Iteration 800: error = 0.1842453, gradient norm = 0.0000304 (50 iterations in 0.037s)

[t-SNE] Iteration 850: error = 0.1831711, gradient norm = 0.0000466 (50 iterations in 0.028s)

[t-SNE] Iteration 900: error = 0.1797995, gradient norm = 0.0000830 (50 iterations in 0.024s)

[t-SNE] Iteration 950: error = 0.1796996, gradient norm = 0.0000333 (50 iterations in 0.026s)

[t-SNE] Iteration 1000: error = 0.1796487, gradient norm = 0.0000288 (50 iterations in 0.025s)

[t-SNE] Error after 1000 iterations: 0.179649

```

and

```

250 0.518333333333

275 0.710416666667

300 0.695833333333

325 0.74625

350 0.78125

375 0.83625

400 0.83875

425 0.915416666667

450 0.917916666667

475 0.89875

500 0.951666666667

525 0.949583333333

550 0.950416666667

575 0.955

600 0.96875

625 0.9775

650 0.957083333333

675 0.95875

700 0.96875

725 0.972916666667

750 0.975

775 0.978333333333

800 0.98125

825 0.98125

850 0.981666666667

875 0.98125

900 0.97875

925 0.980833333333

950 0.980416666667

975 0.980416666667

1000 0.975833333333

```

-------------------------------------------------------------------------

2017-09-12T16:16:11Z priidukull

```

I: Seeding RNGs with 965003854

[t-SNE] Computing 49 nearest neighbors...

[t-SNE] Indexed 50 samples in 0.006s...

[t-SNE] Computed neighbors for 50 samples in 0.004s...

[t-SNE] Computed conditional probabilities for sample 50 / 50

[t-SNE] Mean sigma: 1.228846

[t-SNE] Computed conditional probabilities in 0.007s

[t-SNE] Iteration 50: error = 48.4517632, gradient norm = 0.4771377 (50 iterations in 0.037s)

[t-SNE] Iteration 100: error = 52.6823158, gradient norm = 0.6178778 (50 iterations in 0.033s)

[t-SNE] Iteration 150: error = 54.6605072, gradient norm = 0.4823068 (50 iterations in 0.051s)

[t-SNE] Iteration 200: error = 51.8806496, gradient norm = 0.5132505 (50 iterations in 0.072s)

[t-SNE] Iteration 250: error = 46.6646004, gradient norm = 0.5804820 (50 iterations in 0.071s)

[t-SNE] KL divergence after 250 iterations with early exaggeration: 46.664600

[t-SNE] Iteration 300: error = 0.8332810, gradient norm = 0.0018833 (50 iterations in 0.106s)

[t-SNE] Iteration 350: error = 0.6711912, gradient norm = 0.0015837 (50 iterations in 0.048s)

[t-SNE] Iteration 400: error = 0.5546064, gradient norm = 0.0013941 (50 iterations in 0.041s)

[t-SNE] Iteration 450: error = 0.4738233, gradient norm = 0.0007648 (50 iterations in 0.033s)

[t-SNE] Iteration 500: error = 0.4517629, gradient norm = 0.0002036 (50 iterations in 0.040s)

[t-SNE] Iteration 550: error = 0.4201813, gradient norm = 0.0004996 (50 iterations in 0.050s)

[t-SNE] Iteration 600: error = 0.4113496, gradient norm = 0.0001206 (50 iterations in 0.034s)

[t-SNE] Iteration 650: error = 0.4101939, gradient norm = 0.0000771 (50 iterations in 0.036s)

[t-SNE] Iteration 700: error = 0.4098324, gradient norm = 0.0000733 (50 iterations in 0.031s)

[t-SNE] Iteration 750: error = 0.4081649, gradient norm = 0.0001148 (50 iterations in 0.036s)

[t-SNE] Iteration 800: error = 0.4059633, gradient norm = 0.0001579 (50 iterations in 0.045s)

[t-SNE] Iteration 850: error = 0.4020343, gradient norm = 0.0004064 (50 iterations in 0.042s)

[t-SNE] Iteration 900: error = 0.9964069, gradient norm = 0.0819057 (50 iterations in 0.044s)

[t-SNE] Iteration 950: error = 0.6955136, gradient norm = 0.0015677 (50 iterations in 0.049s)

[t-SNE] Iteration 1000: error = 0.4993443, gradient norm = 0.0011854 (50 iterations in 0.054s)

[t-SNE] Error after 1000 iterations: 0.499344

```

```

I: Seeding RNGs with 1697231718

250 0.515

275 0.619166666667

300 0.804166666667

325 0.823333333333

350 0.808333333333

375 0.875833333333

400 0.910416666667

425 0.902083333333

450 0.933333333333

475 0.948333333333

500 0.942916666667

525 0.91875

550 0.950833333333

575 0.95375

600 0.940833333333

625 0.942916666667

650 0.941666666667

675 0.939583333333

700 0.94875

725 0.952083333333

750 0.954583333333

775 0.950833333333

800 0.950416666667

825 0.952083333333

850 0.954166666667

875 0.955

900 0.628333333333

925 0.753333333333

950 0.798333333333

975 0.8575

1000 0.891666666667

```

-------------------------------------------------------------------------

2017-09-12T21:45:44Z jnothman

So the error is reducing much more slowly...

@priidukull, What did you mean by telling that the divergence happened in C code? Do you have another system you're comparing against?

@tommoral, if we continue to not be able to reproduce this bug, what kind of debugging output do you think would help us understand what's going wrong? Or what kind of more low-level unit tests might help us hone in on it?

-------------------------------------------------------------------------

2017-09-13T05:55:39Z priidukull

I was putting print statements into the code and comparing the values of the variable X during different stages of execution... one environment my Mac desktop and another one that I had set up with docker running on my Mac.

-------------------------------------------------------------------------

2017-09-13T06:05:31Z jnothman

Great. It's extremely helpful to have someone reporting the issue who is

also capable and willing to debug it. If only I could reproduce it on my

mac. I've wasted lots of time failing to set up an appropriate debian

virtual machine.

Do you recall which C function was responsible for the divergence? Is the

input to TSNE.\_tsne identical on both platforms?

I've just realised we have a higher level of verbosity available to us.

Perhaps comparing outputs at verbose=20 will be more informative. Might as

well limit n\_iter to 250, as we know divergence precedes that.

On 13 September 2017 at 15:55, priidukull <notifications@github.com> wrote:

> I was putting print statements into the code and comparing the values of

> the variable X during different stages of execution... one environment my

> Mac desktop and another one that I had set up with docker running on my Mac.

>

> ‚Äî

> You are receiving this because you were mentioned.

> Reply to this email directly, view it on GitHub

> <https://github.com/scikit-learn/scikit-learn/issues/9393#issuecomment-329066826>,

> or mute the thread

> <https://github.com/notifications/unsubscribe-auth/AAEz6547-Q5h6RwXoZP-yphgaBTOiMljks5sh25dgaJpZM4OaXom>

> .

>

-------------------------------------------------------------------------

2017-09-13T06:06:03Z jnothman

Let's use verbose=100 just to be sure (there are some things reported at

verbose=20)

On 13 September 2017 at 16:05, Joel Nothman <joel.nothman@gmail.com> wrote:

> Great. It's extremely helpful to have someone reporting the issue who is

> also capable and willing to debug it. If only I could reproduce it on my

> mac. I've wasted lots of time failing to set up an appropriate debian

> virtual machine.

>

> Do you recall which C function was responsible for the divergence? Is the

> input to TSNE.\_tsne identical on both platforms?

>

> I've just realised we have a higher level of verbosity available to us.

> Perhaps comparing outputs at verbose=20 will be more informative. Might as

> well limit n\_iter to 250, as we know divergence precedes that.

>

> On 13 September 2017 at 15:55, priidukull <notifications@github.com>

> wrote:

>

>> I was putting print statements into the code and comparing the values of

>> the variable X during different stages of execution... one environment my

>> Mac desktop and another one that I had set up with docker running on my Mac.

>>

>> ‚Äî

>> You are receiving this because you were mentioned.

>> Reply to this email directly, view it on GitHub

>> <https://github.com/scikit-learn/scikit-learn/issues/9393#issuecomment-329066826>,

>> or mute the thread

>> <https://github.com/notifications/unsubscribe-auth/AAEz6547-Q5h6RwXoZP-yphgaBTOiMljks5sh25dgaJpZM4OaXom>

>> .

>>

>

>

-------------------------------------------------------------------------

2017-09-13T06:18:09Z jnothman

Okay, I've just noticed a likely bug by reviewing the code (with an eye for a certain issue I previously found in tsne: uninitialised memory), which could be platform dependent or otherwise hard to reproduce:

\* `neg\_f` is [defined as (n\_samples \* n\_dimensions)](https://github.com/scikit-learn/scikit-learn/blob/0.19.0/sklearn/manifold/\_barnes\_hut\_tsne.pyx#L70)

\* `neg\_f` is [populated](https://github.com/scikit-learn/scikit-learn/blob/0.19.0/sklearn/manifold/\_barnes\_hut\_tsne.pyx#L205) for the samples in range [start..stop](https://github.com/scikit-learn/scikit-learn/blob/0.19.0/sklearn/manifold/\_barnes\_hut\_tsne.pyx#L177)

\* `neg\_f` is accessed for gradient computation in range [start..n\_samples](https://github.com/scikit-learn/scikit-learn/blob/0.19.0/sklearn/manifold/\_barnes\_hut\_tsne.pyx#L91)

\* elements (stop+1)..n\_samples may not be initialised

@priidukull, are you able to recompile the cython using calloc instead of malloc?

\* import calloc along with malloc at the top of `\_barnes\_hut\_tsne.pyx`

\* replace `cdef float\* neg\_f = <float\*> malloc(sizeof(float) \* n\_samples \* n\_dimensions)` with `cdef float\* neg\_f = <float\*> calloc(n\_samples \* n\_dimensions, sizeof(float))`

Does this fix the discrepancy?

Ping @tommoral, @ogrisel

-------------------------------------------------------------------------

2017-09-13T07:02:50Z jnothman

This may not be our issue: I'm not managing to get the assertion to fail by merely populating neg\_f with junk

-------------------------------------------------------------------------

2017-09-13T13:26:16Z jnothman

Unfortunately, that's not the issue here (although it should be fixed):

compute\_gradient is only ever called with stop=-1.

On 13 September 2017 at 16:05, Joel Nothman <joel.nothman@gmail.com> wrote:

> Let's use verbose=100 just to be sure (there are some things reported at

> verbose=20)

>

> On 13 September 2017 at 16:05, Joel Nothman <joel.nothman@gmail.com>

> wrote:

>

>> Great. It's extremely helpful to have someone reporting the issue who is

>> also capable and willing to debug it. If only I could reproduce it on my

>> mac. I've wasted lots of time failing to set up an appropriate debian

>> virtual machine.

>>

>> Do you recall which C function was responsible for the divergence? Is the

>> input to TSNE.\_tsne identical on both platforms?

>>

>> I've just realised we have a higher level of verbosity available to us.

>> Perhaps comparing outputs at verbose=20 will be more informative. Might as

>> well limit n\_iter to 250, as we know divergence precedes that.

>>

>> On 13 September 2017 at 15:55, priidukull <notifications@github.com>

>> wrote:

>>

>>> I was putting print statements into the code and comparing the values of

>>> the variable X during different stages of execution... one environment my

>>> Mac desktop and another one that I had set up with docker running on my Mac.

>>>

>>> ‚Äî

>>> You are receiving this because you were mentioned.

>>> Reply to this email directly, view it on GitHub

>>> <https://github.com/scikit-learn/scikit-learn/issues/9393#issuecomment-329066826>,

>>> or mute the thread

>>> <https://github.com/notifications/unsubscribe-auth/AAEz6547-Q5h6RwXoZP-yphgaBTOiMljks5sh25dgaJpZM4OaXom>

>>> .

>>>

>>

>>

>

-------------------------------------------------------------------------

2017-09-13T23:32:15Z jnothman

@priidukull your verbose=20 output would be welcome. I'm otherwise at a loss.

-------------------------------------------------------------------------

2017-09-14T06:18:45Z priidukull

Where can I set verbose=20?

-------------------------------------------------------------------------

2017-09-14T06:19:55Z jnothman

Sorry

```python

import numpy as np

from sklearn.manifold.t\_sne import TSNE, trustworthiness

from sklearn.utils import check\_random\_state

random\_state = check\_random\_state(0)

X = random\_state.randn(50, 2).astype(np.float32)

tsne = TSNE(n\_components=2, init='pca', random\_state=0,

method='barnes\_hut', verbose=10, n\_iter=250)

X\_embedded = tsne.fit\_transform(X)

t = trustworthiness(X, X\_embedded, n\_neighbors=1)

assert t > 0.9

```

-------------------------------------------------------------------------

2017-09-14T06:35:38Z priidukull

Ran:

```

def test\_preserve\_trustworthiness\_approximately():

import numpy as np

from sklearn.manifold.t\_sne import TSNE, trustworthiness

from sklearn.utils import check\_random\_state

random\_state = check\_random\_state(0)

X = random\_state.randn(50, 2).astype(np.float32)

n\_iter = 250

tsne = TSNE(n\_components=2, init='pca', random\_state=0,

method='barnes\_hut', verbose=20, n\_iter=n\_iter)

X\_embedded = tsne.fit\_transform(X)

t = trustworthiness(X, X\_embedded, n\_neighbors=1)

print(n\_iter, t)

```

Output: https://gist.github.com/priidukull/1453adb7cf2bca2093b2dd9d6646f64e

-------------------------------------------------------------------------

2017-09-14T08:06:04Z jnothman

@priidukull thanks for that. But the verbose output you have sent had substantial discrepancy with what it should, and not just in the numbers. Are you certain that the library is correctly compiled? Do you get this error when running the test on the wheel version of scikit-learn 0.19?

-------------------------------------------------------------------------

2017-09-14T08:32:29Z jnothman

Or maybe that comment was wrong and I was just confused because your output

isn't complete: the beginning is cut off

On 14 Sep 2017 4:35 pm, "priidukull" <notifications@github.com> wrote:

> Ran:

>

> def test\_preserve\_trustworthiness\_approximately():

> import numpy as np

> from sklearn.manifold.t\_sne import TSNE, trustworthiness

> from sklearn.utils import check\_random\_state

> random\_state = check\_random\_state(0)

> X = random\_state.randn(50, 2).astype(np.float32)

> n\_iter = 250

> tsne = TSNE(n\_components=2, init='pca', random\_state=0,

> method='barnes\_hut', verbose=20, n\_iter=n\_iter)

> X\_embedded = tsne.fit\_transform(X)

> t = trustworthiness(X, X\_embedded, n\_neighbors=1)

> print(n\_iter, t)

>

>

> Output: https://gist.github.com/priidukull/1453adb7cf2bca2093b2dd9d6646f6

> 4e

>

> ‚Äî

> You are receiving this because you were mentioned.

> Reply to this email directly, view it on GitHub

> <https://github.com/scikit-learn/scikit-learn/issues/9393#issuecomment-329387495>,

> or mute the thread

> <https://github.com/notifications/unsubscribe-auth/AAEz60XYRRel6OV\_ndR-HVrMdFYzs3oPks5siMk8gaJpZM4OaXom>

> .

>

-------------------------------------------------------------------------

2017-09-14T08:57:58Z priidukull

Uups, I missed that. The output is more than 1Mb in size, so I did not find a pastebin for that. Can run the test again. How do you suggest that I send the output to you?

-------------------------------------------------------------------------

2017-09-14T09:20:37Z jnothman

you can email my personal address for the, thanks

On 14 Sep 2017 6:58 pm, "priidukull" <notifications@github.com> wrote:

> Uups, I missed that. The output is more than 1Mb in size, so I did not

> find a pastebin for that. Can run the test again. How do you suggest that I

> send the output to you?

>

> ‚Äî

> You are receiving this because you were mentioned.

> Reply to this email directly, view it on GitHub

> <https://github.com/scikit-learn/scikit-learn/issues/9393#issuecomment-329419033>,

> or mute the thread

> <https://github.com/notifications/unsubscribe-auth/AAEz65k\_b9cjArMLyXwbcm8vOUBL13GRks5siOqXgaJpZM4OaXom>

> .

>

-------------------------------------------------------------------------

2017-09-14T09:20:52Z jnothman

or zip it

On 14 Sep 2017 6:58 pm, "priidukull" <notifications@github.com> wrote:

> Uups, I missed that. The output is more than 1Mb in size, so I did not

> find a pastebin for that. Can run the test again. How do you suggest that I

> send the output to you?

>

> ‚Äî

> You are receiving this because you were mentioned.

> Reply to this email directly, view it on GitHub

> <https://github.com/scikit-learn/scikit-learn/issues/9393#issuecomment-329419033>,

> or mute the thread

> <https://github.com/notifications/unsubscribe-auth/AAEz65k\_b9cjArMLyXwbcm8vOUBL13GRks5siOqXgaJpZM4OaXom>

> .

>

-------------------------------------------------------------------------

2017-09-15T13:19:26Z priidukull

[output.txt.zip](https://github.com/scikit-learn/scikit-learn/files/1306459/output.txt.zip)

-------------------------------------------------------------------------

2017-09-15T13:20:23Z priidukull

Test code:

```

def test\_preserve\_trustworthiness\_approximately():

import numpy as np

from sklearn.manifold.t\_sne import TSNE, trustworthiness

from sklearn.utils import check\_random\_state

random\_state = check\_random\_state(0)

X = random\_state.randn(50, 2).astype(np.float32)

n\_iter = 250

tsne = TSNE(n\_components=2, init='pca', random\_state=0,

method='barnes\_hut', verbose=20, n\_iter=n\_iter)

X\_embedded = tsne.fit\_transform(X)

t = trustworthiness(X, X\_embedded, n\_neighbors=1)

print(n\_iter, t)

```

-------------------------------------------------------------------------

2017-09-15T15:01:45Z tomMoral

@priidukull Thanks for the log. I tried to read it but there is only the logs after iteration 200 ("QuadTree" is way too verbose and I think we lose the beginning as the log growth too big).

I created a branch with better debugging logs here on top of @jnothman nonstop branch : https://github.com/tomMoral/scikit-learn/tree/nonstop . Could you please check it out and re-run the same code?

It prints the squared norm of the gradient and the error at each iteration so we can see which part of the code is diverging.

Here is the output for the first 100 iterations (it is still too big for the gist)

[output.text](https://gist.github.com/tomMoral/b1e19e3f5e9578e082b25155c299c6d1)

-------------------------------------------------------------------------

2017-09-15T22:07:46Z priidukull

Test code:

```

def test\_preserve\_trustworthiness\_approximately():

import numpy as np

from sklearn.manifold.t\_sne import TSNE, trustworthiness

from sklearn.utils import check\_random\_state

random\_state = check\_random\_state(0)

X = random\_state.randn(50, 2).astype(np.float32)

n\_iter = 250

tsne = TSNE(n\_components=2, init='pca', random\_state=0,

method='barnes\_hut', verbose=20, n\_iter=n\_iter)

X\_embedded = tsne.fit\_transform(X)

t = trustworthiness(X, X\_embedded, n\_neighbors=1)

from builtins import print

print(n\_iter, t)

```

[output2.txt.zip](https://github.com/scikit-learn/scikit-learn/files/1307832/output2.txt.zip)

-------------------------------------------------------------------------

2017-09-17T15:46:22Z jnothman

Thanks again. Still all we can compare by is error, and not, say, gradients:

The first 20:

```

@jnothman's @priidukull's

1 [t-SNE] Computed error=30.6823 [t-SNE] Computed error=30.6823

2 [t-SNE] Computed error=37.7064 [t-SNE] Computed error=37.7064

3 [t-SNE] Computed error=38.4978 [t-SNE] Computed error=38.4978

4 [t-SNE] Computed error=40.5980 [t-SNE] Computed error=40.5980

5 [t-SNE] Computed error=41.7238 [t-SNE] Computed error=41.7239

6 [t-SNE] Computed error=42.2833 [t-SNE] Computed error=42.2837

7 [t-SNE] Computed error=46.8638 [t-SNE] Computed error=46.8647

8 [t-SNE] Computed error=48.3388 [t-SNE] Computed error=48.3393

9 [t-SNE] Computed error=50.7234 [t-SNE] Computed error=50.7251

10 [t-SNE] Computed error=46.2566 [t-SNE] Computed error=46.2448

11 [t-SNE] Computed error=43.9031 [t-SNE] Computed error=43.8736

12 [t-SNE] Computed error=44.5618 [t-SNE] Computed error=44.4002

13 [t-SNE] Computed error=44.1744 [t-SNE] Computed error=43.7296

14 [t-SNE] Computed error=43.2287 [t-SNE] Computed error=42.8211

15 [t-SNE] Computed error=47.0822 [t-SNE] Computed error=48.7362

16 [t-SNE] Computed error=48.5537 [t-SNE] Computed error=48.3436

17 [t-SNE] Computed error=48.3894 [t-SNE] Computed error=47.8601

18 [t-SNE] Computed error=47.2820 [t-SNE] Computed error=47.6722

19 [t-SNE] Computed error=48.2992 [t-SNE] Computed error=49.3187

20 [t-SNE] Computed error=47.6544 [t-SNE] Computed error=50.4518

```

We see that the first error is a small numerical imprecision at line 5, but that this quite quickly blows out.

I'm not sure that this is quite sufficient to say that there is nothing fundamentally broken in the implementation (e.g. accessing randomly initialised memory), but that:

\* it is more susceptible to numerical imprecision than we would like, but perhaps we should (seek contributions that) investigate stability improvements

\* the test is brittle and already provides only weak assurances in asserting t > 0.9

\* given this, we can probably get away with lowering the threshold, with a comment referencing this issue

However, we may also be able to improve stability by choosing a better random data production approach; this random seed produces data where the following are the smallest differences between any pairwise distances in X:

```python

>>> D = pairwise\_distances(X)

>>> d = D[np.tril\_indices\_from(D, -1)]

>>> deltas = np.sort(np.diff(np.sort(d)))

>>> deltas

array([ 6.55651093e-07, 3.57627869e-06, 4.11272049e-06, ...,

9.16552544e-02, 1.93810940e-01, 5.23059368e-01], dtype=float32)

```

That's very small differences for float32 data, and a large range in exponent from min to max.

Is there a reason this test needs to use randn? Can it have a higher variance? Multiplying X by 1000 will mean at least the pairwise distances are much more distinguished in a float32, which I \*think\* may help.

-------------------------------------------------------------------------

2017-09-18T12:15:54Z jnothman

So I guess that's a question to @priidukull too. Does the following `test\_preserve\_trustworthiness\_approximately(s=100)` easily pass for you?

```

def test\_preserve\_trustworthiness\_approximately(s=1):

import numpy as np

from sklearn.manifold.t\_sne import TSNE, trustworthiness

from sklearn.utils import check\_random\_state

random\_state = check\_random\_state(0)

X = random\_state.randn(50, 2).astype(np.float32) \*s

tsne = TSNE(n\_components=2, init='pca', random\_state=0,

method='barnes\_hut', n\_iter=100000)

X\_embedded = tsne.fit\_transform(X)

t = trustworthiness(X, X\_embedded, n\_neighbors=1)

print(t)

assert t > .9

```

-------------------------------------------------------------------------

2017-09-19T11:58:10Z ogrisel

I think @albertcthomas's [fix](https://github.com/scikit-learn/scikit-learn/pull/9340/commits/b360c9a4e169a02d81febe1024223dcc4708ecff) in #9340 is the right fix:

- This test generates training data as 32 bit float

- The Barnes Hut Cython code works on 32 bit float

- The Python validation of the `TSNE.fit` code would therefore upcast 32-bit float data into 64 bit floats before casting down back to 32 bit float to call into the Cython code.

Upcasting from 32 bit to 64 bit is platform specific (the new bits are not necessarily set to zero) and can explain the non deterministic behaviour with observed on some platforms / machines.

We need to pass 32 bit to 32 bit cython code without upcasting (which also wastes memory for nothing).

-------------------------------------------------------------------------

2017-09-19T12:20:45Z albertcthomas

Just to be sure we are on the same page, the fix I suggested in #9340 consists in having `dtype=[np.float32, np.float64]` in a `check\_X\_y` I added in this same PR. The `check\_X\_y` in master already has a `dtype=[np.float32, np.float64]` (see [here](https://github.com/scikit-learn/scikit-learn/blob/master/sklearn/manifold/t\_sne.py#L659)) (#9340 is older than the tSNE memory usage fix that was merged in July).

-------------------------------------------------------------------------

2017-09-19T12:30:52Z ogrisel

Ah then there is something I do not understand. Will need to investigate further.

-------------------------------------------------------------------------

2017-09-19T14:27:59Z ogrisel

> Is there a reason this test needs to use randn? Can it have a higher variance? Multiplying X by 1000 will mean at least the pairwise distances are much more distinguished in a float32, which I think may help.

+1 for trying with larger variance or even a different distribution (e.g. uniform).

-------------------------------------------------------------------------

2017-09-20T09:22:45Z ogrisel

Actually, multiplying the data by 100 does not make the algorithm more stable with the PCA init, quite the opposite actually. On the original machine, the exact method + PCA init was triggering the instability according to: https://github.com/scikit-learn/scikit-learn/issues/9393#issuecomment-322214890

Changing the random seed can have a large impact on the outcome. So maybe the rounding errors can indeed also have a large impact. By increasing the number of samples to 100 (instead of 50), the trustworthiness gets much better (and therefore much more stable) but the test is significantly slower (couple of seconds on my machine).

-------------------------------------------------------------------------

2017-09-20T13:35:58Z ogrisel

Ok after playing extensively with different random seeds and platforms (mkl vs openblas PCA for the init) I think that 0.9 is just too strict. We could keep the 0.9 threshold and stabilize this test by:

- running TSNE on larger datasets (in which case the trustworthiness score gets more stable)

- running the tests several times with different random seeds and make an assertion on the median score.

However both approaches are too expensive in my opinion. While running my test with several hundred seeds on the original 50 samples random dataset I have never seen this score go below 0.87. So I think setting it to 0.85 should fix the issue. I will submit a PR.

-------------------------------------------------------------------------

2017-10-24T02:01:18Z yarikoptic

FWIW, this issue still happens on 32bit debian stretch with 0.19.1

```shell

======================================================================

FAIL: sklearn.manifold.tests.test\_t\_sne.test\_preserve\_trustworthiness\_approximately

----------------------------------------------------------------------

Traceback (most recent call last):

File "/usr/lib/python2.7/dist-packages/nose/case.py", line 197, in runTest

self.test(\*self.arg)

File "/build/scikit-learn-0.19.1/debian/tmp/usr/lib/python2.7/dist-packages/sklearn/manifold/tests/test\_t\_sne.py", line 249, in test\_preserve\_trustworthiness\_approximately

assert\_greater(t, 0.9)

AssertionError: 0.89166666666666661 not greater than 0.9

```

-------------------------------------------------------------------------

2017-10-24T02:42:57Z jnothman

It looks like that PR was not copied across correctly to 0.19.1. My fault.

Should be working in master, though, and seeing as the solution was simply

to lower the threshold to 0.85, I don't think we're going to make another

bug-fix release. Feel free to patch for Debian.

On 24 October 2017 at 13:01, Yaroslav Halchenko <notifications@github.com>

wrote:

> FWIW, this issue still happens on 32bit debian stretch with 0.19.1

>

> ======================================================================

> FAIL: sklearn.manifold.tests.test\_t\_sne.test\_preserve\_trustworthiness\_approximately

> ----------------------------------------------------------------------

> Traceback (most recent call last):

> File "/usr/lib/python2.7/dist-packages/nose/case.py", line 197, in runTest

> self.test(\*self.arg)

> File "/build/scikit-learn-0.19.1/debian/tmp/usr/lib/python2.7/dist-packages/sklearn/manifold/tests/test\_t\_sne.py", line 249, in test\_preserve\_trustworthiness\_approximately

> assert\_greater(t, 0.9)

> AssertionError: 0.89166666666666661 not greater than 0.9

>

> ‚Äî

> You are receiving this because you were mentioned.

> Reply to this email directly, view it on GitHub

> <https://github.com/scikit-learn/scikit-learn/issues/9393#issuecomment-338848932>,

> or mute the thread

> <https://github.com/notifications/unsubscribe-auth/AAEz63RTLsnmw0CIzX94a-xd0CVbnTxZks5svUTvgaJpZM4OaXom>

> .

>

-------------------------------------------------------------------------

2017-10-24T02:43:49Z jnothman

You can cherry-pick 6c99d797 if you wish.

-------------------------------------------------------------------------

2017-10-24T21:28:41Z amueller

there was apparently also a 32bit failure on windows for 0.19.1, but I don't think it was this one.

-------------------------------------------------------------------------

ons/unsubscribe-auth/AAEz63RTLsnmw0CIzX94a-xd0CVbnTxZks5svUTvgaJpZM4OaXom>

> .

>

-------------------------------------------------------------------------

2017-10-24T02:43:49Z jnothman

You can cherry-pick 6c99d797 if you wish.

-------------------------------------------------------------------------

2017-10-24T21:28:41Z amueller

there was apparently also a 32bit failure on windows for 0.19.1, but I don't think it was this one.

-------------------------------------------------------------------------