-1

بیش‌برازش یا overfitting زمانی اتفاق می‌افتد که مدل داده‌های training را حفظ می‌کند و یاد نمی‌گیرد بنابراین روی داده‌ی test خوب تعمیم نمیابد. مثلا دقت روی داده‌های training برابر 99 درصد و روی داده‌ی test برابر 50 درصد می‌شود. اگر می‌خواهیم overfitting را بهتر بفهمیم بهتر است ابتدا نگاهی به underfitting بیندازیم.
Underfitting زمانی اتفاق می‌افتد که مدل بسیار ساده است (تعداد ویژگی‌ها کم است یا مدل بیش از حد regularize شده است). یادگیرنده‌های ساده واریانس (یا پراکندگی) کمی دارند و بایاسشان نسبت به جواب غلط زیاد است. Overfitting ناشی از پیچیدگی بیش از حد مدل است. مدل‌های پیچیده واریانس بیشتری در پیشبینی‌هایشان دارند. Overfitting هنگامی اتفاق می‌افتد که یک مدل جزئیات و noise را در داده های آموزشtraining بیاموزد تا حدی که بر عملکرد مدل روی داده های جدید test تأثیر منفی بگذارد. این بدان معنی است که noise یا نوسانات تصادفی در داده های آموزش توسط مدل به عنوان مفاهیم انتخاب و یاد گرفته می شود. مسئله این است که این مفاهیم در مورد داده های جدید اعمال نمی شود و بر توانایی مدل سازی برای تعمیم تأثیر منفی می گذارد.

در اینجا به چند راه‌حل معروف برای Overfitting اشاره شده است:

1- Cross validation

Cross validation یک اقدام پیشگیرانه قدرتمند در برابر Overfitting است. از داده های آموزش اولیه خود برای تولید چندین قسمت داده‌ی train و test کوچک استفاده کنید. برای تنظیم مدل خود از این قسمت‌ها استفاده کنید.

2- با داده های بیشتر آموزش دهید

این همیشه کار نمی‌کند، اما آموزش با داده های بیشتر به الگوریتم ها کمک می کند تا بهتر یاد بگیرند.

3- حذف ویژگی‌ها

بعضی از الگوریتم‌ها به صورت built-in از بین ویژگی‌ها چند تا را انتخاب می‌کنند. برای آن دسته از الگوریتم‌ها که این کار را نمی‌کنند ، می‌توانید با حذف ویژگی‌های ورودی بی‌ربط ، قابلیت تعمیم آنها را به صورت دستی بهبود ببخشید.

4- Early stopping یا توقف زودهنگام

هنگامی که یک الگوریتم یادگیری را به صورت تکراری آموزش می دهید ، می توانید عملکرد هر تکرار مدل را به خوبی ارزیابی کنید. تا تعداد مشخصی از تکرارها ، تکرارهای جدید مدل را بهبود می بخشند. با این وجود ، پس از آن مرحله ، توانایی تعمیم مدل می تواند ضعیف شود تا جایی که شروع بهoverfit شدن می‌کند. Early stopping به توقف روند آموزش قبل از رسیدن یادگیرنده ابه آن نقطه اشاره دارد.

5- Regularization

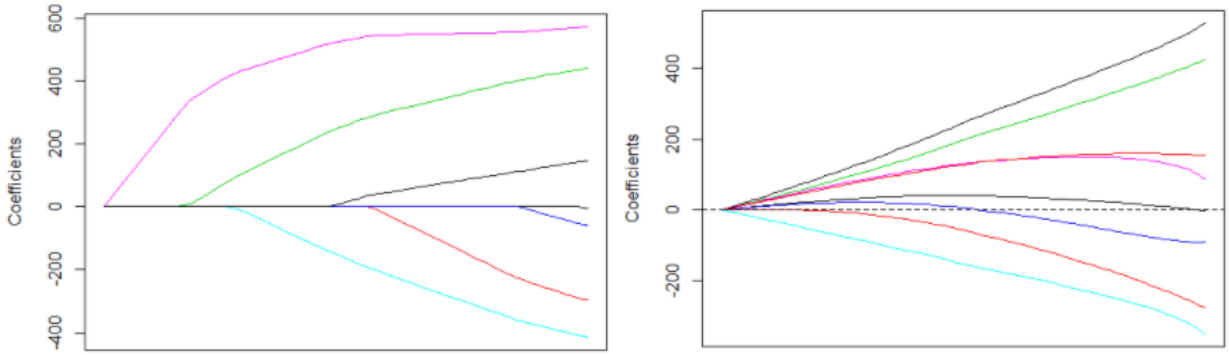
Regularization به طیف گسترده‌ای از تکنیک‌ها اشاره دارد که مدل را مجبور به ساده شدن می‌کند. تکنیک مورد نظر به نوع learnerای که استفاده می‌کنید بستگی دارد. به عنوان مثال ، شما می توانید یک درخت تصمیم را هرس کنید ، از dropout در یک شبکه عصبی استفاده کنید ، یا یک پارامتر جریمه به تابع ضرر در رگرسیون اضافه کنید.

اغلب اوقات ، روش Regularization نیز یک hyperparameter است ، به این معنی که می‌توان آن را از طریق cross-validation تنظیم کرد.

https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/

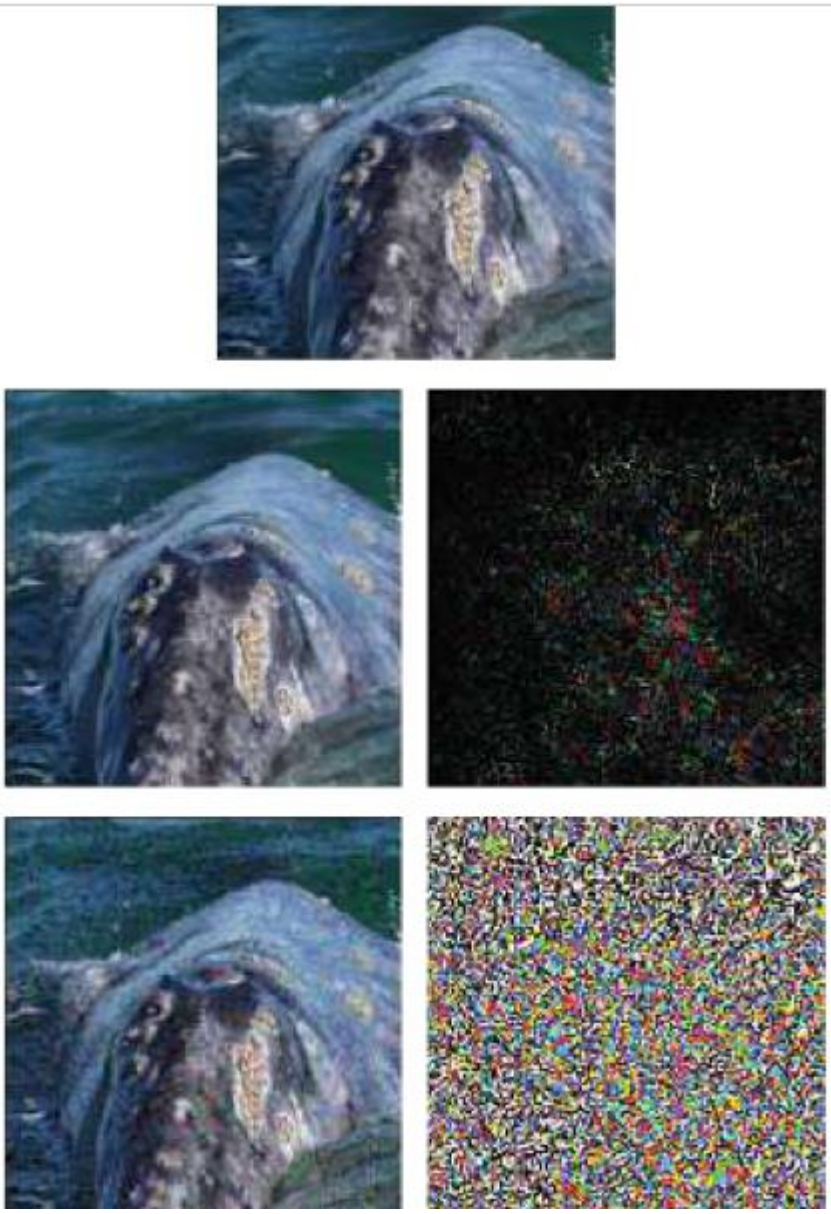
https://elitedatascience.com/overfitting-in-machine-learning

2-



در L1-Regularization دنبال این هستیم که برخی از وزن‌ها مقدار صفر داشته باشند و در L2-Regularization دنبال این هستیم که تمام وزن‌ها مقدار کوچکی داشته باشند بنابراین شکل سمت راست که در آن تلاش شده تمام ضرائب مقدار کوچکی داشته باشند مربوط به L2-Regularization و شکل سمت چپ که مثلا خط آبی رنگ در بازه‌ای کاملا صفر هست مربوط به L1-Regularization هست.

3- شبکه‌های عمیق ابزار قدرتمندی هستند و عملکرد بسیار خوبی در زمینه‌های مختلف مثل بیوانفورماتیک، پردازش صوت و بینایی کامپیوتر دارند. اگرچه این شبکه‌ها عملکرد بسیار خود در کارهای classification دارند اما اخیرا نشان داده شده است که این شبکه‌ها در مقابل تغییرات خصمانه (adversarial) در داده‌ها ناپایدار هستند. در واقع تغییرات بسیار کوچک و نامحسوس در نمونه‌های داده کافی است تا طبقه‌بند (classifier) فریب بخورد و منجر به تشخیص اشتباه کلاس داده می‌شود. به عبارت رسمی‌تر برای یک طبقه‌بند (classifier) تغییر خصمانه (adversarial) را به صورت کمترین تغییرات که باعث می‌شود لیبل تشخیص داده شده k^(x) تغییر کند.



برای مثال در شکل قوق در سطر اول یک نهنگ را می‌بینیم که در سطرهای دوم و سوم سمت چپ که تغییر adversarial روی آن اعمال شده و به اشتباه لاک‌پشت تشخیص داده شده است. در سطرهای دوم و سوم سمت راست تغییرات adversarial را نشان داده‌ایم که در سطر دوم با الگوریتم DeepFool و سطر سوم با الگوریتم محاسبه شده است.

الگوریتمهای متفاوتی برای محاسبه میزان مقاوم بودن robustness طبقه‌بند classifier نوشته‌است. در این مقاله الگوریتم DeepFool معرفی شده است. Robustness به صورت زیر تعریف می‌شود.

$$\Delta(\boldsymbol{x}; \hat{k}) := \min_{\boldsymbol{r}} \|\boldsymbol{r}\|_2 \text{ subject to } \hat{k}(\boldsymbol{x} + \boldsymbol{r}) \neq \hat{k}(\boldsymbol{x}),$$

که x یک تصویر k^(x) لیبل تخمین زده شده برای آن و Δ(x;k^) برابر با Robustness برای k^ در نقطه‌ی x می‌باشد. Robustness برای classifier به صورت زیر تعریف می‌شود.

$$\rho_{\text{adv}}(\hat{k}) = \mathbb{E}_{\boldsymbol{x}} \frac{\Delta(\boldsymbol{x}; \hat{k})}{\|\boldsymbol{x}\|_2},$$

که E امید یا میانگین روی توزیع داده‌ها است.

یکی از روش‌های حل مسئله adversarial attack این است که با استفاده از نمونه‌های adversarial شبکه را fine-tune کنیم. در این مقاله با استفاده از الگوریتم DeepFool این عمل fine-tune را انجام می‌دهیم و Robustness شبکه را بالا می‌بریم. مشاهدات نشان داده است که این روش به صورت قابل ملاحظه‌ای Robustness را افزایش می‌دهد مثلا برای داده‌ی MNIST این افزایش Robustness برابر 50 درصد بوده‌است.

الگوریتم DeepFool برای binary classification و multi class classification در زیر آمده است.

Algorithm 1 DeepFool for binary classifiers

input: Image $\boldsymbol{x}$, classifier f.
output: Perturbation $\hat{\boldsymbol{r}}$.
Initialize $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}$, $i \leftarrow 0$.
while $\text{sign}(f(\boldsymbol{x}_i)) = \text{sign}(f(\boldsymbol{x}_0))$ **do**
    $\boldsymbol{r}_i \leftarrow -\frac{f(\boldsymbol{x}_i)}{\|\nabla f(\boldsymbol{x}_i)\|_2^2} \nabla f(\boldsymbol{x}_i)$,
    $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + \boldsymbol{r}_i$,
    $i \leftarrow i + 1$.
end while
return $\hat{\boldsymbol{r}} = \sum_i \boldsymbol{r}_i$.

Algorithm 2 DeepFool: multi-class case

input: Image $\boldsymbol{x}$, classifier f.
output: Perturbation $\hat{\boldsymbol{r}}$.

Initialize $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}$, $i \leftarrow 0$.
while $\hat{k}(\boldsymbol{x}_i) = \hat{k}(\boldsymbol{x}_0)$ **do**
    for $k \neq \hat{k}(\boldsymbol{x}_0)$ **do**
        $\boldsymbol{w}'_k \leftarrow \nabla f_k(\boldsymbol{x}_i) - \nabla f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)$
        $f'_k \leftarrow f_k(\boldsymbol{x}_i) - f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)$
    end for
    $\hat{l} \leftarrow \arg\min_{k \neq \hat{k}(\boldsymbol{x}_0)} \frac{|f'_k|}{\|\boldsymbol{w}'_k\|_2}$
    $\boldsymbol{r}_i \leftarrow \frac{|f'_{\hat{l}}|}{\|\boldsymbol{w}'_{\hat{l}}\|_2^2} \boldsymbol{w}'_{\hat{l}}$
    $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + \boldsymbol{r}_i$
    $i \leftarrow i + 1$
end while
return $\hat{\boldsymbol{r}} = \sum_i \boldsymbol{r}_i$

یکی دیگر از روش‌هایی که در این مقاله به آن اشاره نشده است ولی در کلاس استاد به آن اشاره داشتند این بود که داده‌های adversarial را در هنگام train به همراه لیبل درستشان به شبکه معرفی کنیم.

-4

در ابتدا learning rateهای متفاتی برای آپتیمایزر adam تست شد از 1e-5,1e-4,1e-3,0.01,0.1 که 1e-4 بهترین نتیجه را روی عملکرد مدل classification داشت و باعث شد آن را انتخاب کنیم. ضمن اینکه باید اشاره شود که در epoch 2 این مسئله اصلا overfit نشده دقت train برابر 80 و دقت test برابر 76 درصد بوده و انواع روش‌های regularization تاثیر قابل مشاهده‌ای روی جواب نداشتند بنابراین از epoch را به 10 افزایش دادیم تا بتوانیم تحلیل جامع‌تری داشته باشیم.

model	Train accuracy	Test accuracy
without regularization	95%	81%
with data augmentation	88	84
with dropout=0.5,0.05	89	81
with dropout= 0.5,0.1	89	80
with dropout= 0.4,0.2	91	81
with L2-regularization= 0.0001	96	81
with L2-regularization= 0.001	93	82
with L2-regularization= 0.005	86	80

در train بدون regularization همانطور که در جدول درج شده دقت train برابر 95% و دقت test برابر 81 درصد شده و به مشکل overfit برخوردیم. حال به تاثیر انواع روش‌های regularization می‌پردازیم.

زمانی که از data augmentation استفاده کردیم انواع augmentation تست شد اما از آنجا که سایز عکس‌ها در دیتاست cifar 10 برابر 32×32 هست وکوچک می‌باشد و مثلا تصویر از وضوح کمتری برخوردار هست augmentation زیاد و پیچیده باعث تاثیر منفی بر روی دقت روی داده‌های test شد بنابراین از انجام آن‌ها

خودداری گردید و به دو augmentation ساده اکتفا کردیم. دقت train از 95 به 88 رسید و دقت test از 81 به 84 که بسیار خوب است.

زمانی که از dropout=0.5 برای لایه‌های fully connected و از dropout=0.05 برای لایه‌های convolution استفاده شد دقت داده‌های test اضافه نشد اما دقت داده‌های train از 95 به 89 کاهش یافت که کمی از overfit شدن جلوگیری کرد.

زمانی که از dropout=0.5 برای لایه‌های fully connected و از dropout=0.1 برای لایه‌های convolution استفاده شد دقت داده‌های test یک درصد کاهش یافت اما دقت داده‌های train از 95 به 89 کاهش یافت که کمی از overfit شدن جلوگیری کرد.

زمانی که از dropout=0.4 برای لایه‌های fully connected و از dropout=0.2 برای لایه‌های convolution استفاده شد دقت داده‌های test اضافه نشد اما دقت داده‌های train از 95 به 91 کاهش یافت که کمی از overfit شدن جلوگیری کرد.

با مشاهده تنظیمات فوق در جدول و بسیاری از تنظیمات دیگری که برای dropout پیاده‌سازی گردید ولی در جدول درج نشد مشاهده شد که dropout تاثیر کمی در افزایش دقت داده‌های تست داشت اما از overfit شدن جلوگیری می‌کرد.

قبل از توضیح L2-regularization باید اضافه شود که برای پیاده‌سازی آن از weight decay در optimizer استفاده شده است.

زمانی که از L2-regularization= 0.0001 استفاده شد دقت داده‌های test اضافه نشد اما دقت داده‌های train از 95 به 96 افزایش یافت که تاثیری در جلوگیری از overfit نداشت و بدتر هم شد.

زمانی که از L2-regularization= 0.001 استفاده شد دقت داده‌های test یک درصد افزایش یافت اما دقت داده‌های train از 95 به 93 کاهش یافت که کمی از overfit شدن جلوگیری کرد اما هنوز هم overfit هست.

زمانی که از L2-regularization= 0.005 استفاده شد دقت داده‌های test یک درصد کاهش یافت اما دقت داده‌های train از 95 به 86 کاهش یافت که تا حد خوبی از overfit شدن جلوگیری کرد.

با مشاهده تنظیمات فوق در جدول و بسیاری از تنظیمات دیگری که برای L2-regularization پیاده‌سازی گردید ولی در جدول درج نشد مشاهده شد که L2-regularization تاثیر کمی در افزایش دقت داده‌های تست داشت اما از overfit شدن جلوگیری می‌کرد.

در کد نهایی از dropout=0.5 برای لایه‌های fully connected و از dropout=0.05 برای لایه‌های convolution و L2-regularization= 0.005 و data augmentation دقت روی داده‌های تست از 81 به 79 رسیده و روی داده‌های train از 95 به 81 که مشاهده می‌شود مسئله overfit شدن کاملا حل شده است.

در کد نهایی از dropout=0.5 برای لایه‌های fully connected و از dropout=0.05 برای لایه‌های convolution و L2-regularization= 0.001 و data augmentation دقت روی داده‌های تست از 81 به 80 رسیده و روی داده‌های train از 95 به 83 که مشاهده می‌شود مسئله overfit شدن کاملا حل شده است.

در کل در تمام موارد فوق نتیجه روی داده‌ی تست فقط در data augmentation پیشرفت خوبی داشته و overfit نسبتا کمی داشته اما هنگامی که تمامی موارد regularization اعمال شده دقت روی داده تست افت کرده اما مسئله overfit حل شده است.