Image inpainting – Assignment 2

Michèle Wyss
10-104-123

1 Primal-dual formulation

Our objective function that we want to minimize is given by

$$E(u) = \frac{\lambda}{2}\|u-g\|_\Omega^2 + \|\nabla u\|_2.$$

In General, given an initial problem

$$\min_{x\in X} F(Ku) + G(x),$$

a primal-dual formulation is given by

$$\min_{x\in X}\max_{y\in Y}\langle Kx, y\rangle - F^*(y) + G(x). \tag{1}$$

In the above formulations, F and G are convex and K is a linear operator. Defining $K = \nabla$, $F = \|\cdot\|_2$ and $G(u) = \frac{\lambda}{2}\|u-g\|_\Omega^2$, we can therefore get the following primal-dual formulation of our original problem:

$$\min_u \max_p \langle p, \nabla u\rangle - F^*(p) + \frac{\lambda}{2}\|u-g\|_\Omega^2,$$

where the Legendre-Fenchel transform of F can easily be found as follows:

$$\begin{aligned} F^*(y) = (\|\cdot\|_2)^*(y) &= \sup_x x^T y - \|x\|_2 \\ &= \sup_x x^T y - \max_{\|z\|_2\le 1} x^T z \\ &= \sup_x \min_{\|z\|\le 1} x^T(y-z) \\ &= \begin{cases} 0 \text{ if } \|y\| \le 1, \\ \infty \text{ otherwise.} \end{cases} \end{aligned}$$

2 Primal-Dual steps

Starting again at our general formulation of the saddle-point problem from equation (1), we want to use the following algorithm, proposed by A. Chambolle and T. Pock ("Algorithm 1" in [CP11]):

$$\begin{aligned} y^{n+1} &= \operatorname{prox}_{\sigma F^*}(y^n + \sigma K \bar{x}^n) \\ x^{n+1} &= \operatorname{prox}_{\tau G}(x^n - \tau K^* y^{n+1}) \\ \bar{x}^{n+1} &= x^{n+1} + \theta(x^{n+1} - x^n), \end{aligned}$$

where $\theta \in (0,1]$ and $\tau\sigma\|K\|^2 < 1$.

The operators "prox" are so-called *proximity operators.* They are defined as

$$\operatorname{prox}_{\lambda F}(x) := \arg\min_z \frac{1}{2}\|x - z\|_2^2 + \lambda F(z) \tag{2}$$

2.1 Find y^{n+1}

Finding the proximity operator $\operatorname{prox}_{\sigma(\|\cdot\|)*}$ can be done by e.g. using *Moreau's identity* (sometimes also referred to as *Moreaus's theorem*). This theorem states that

$$\operatorname{prox}_{\lambda F^*}(x) = x - \lambda \operatorname{prox}_{\frac{F}{\lambda}}\left(\frac{x}{\lambda}\right).$$

Applying this, we can calculate the proximity operator of the Legendre-Fenchel transform based on the proximity operator of the l_2-norm and get

$$\begin{aligned} \operatorname{prox}_{\sigma F^*}(y^n + \sigma K\bar{x}^n) &= y^n + \sigma K \bar{x}^n - \sigma \operatorname{prox}_{\frac{F}{\sigma}}\left(\frac{y^n + \sigma K \bar{x}^n}{\sigma}\right) \\ &= (y^n + \sigma K\bar{x}^n) - \sigma \cdot \frac{y^n + \sigma K\bar{x}^n}{\sigma} \cdot \max\left\{0, 1 - \frac{1}{\|y^n + \sigma K \bar{x}^n\|_2}\right\} \\ &= (y^n + \sigma K\bar{x}^n) - (y^n + \sigma K\bar{x}^n) \cdot \max\left\{0, 1 - \frac{1}{\|y^n + \sigma K \bar{x}^n\|_2}\right\} \end{aligned}$$

Here, the second equality follows from the fact that

$$\operatorname{prox}_{\frac{\|\cdot\|_2}{\sigma}}\left(\frac{x}{\sigma}\right) = \frac{x}{\sigma} \cdot \max\left\{0, 1 - \frac{1}{\|x\|_2}\right\}.$$

Let's go step by step and consider the two possible cases of how to compute the wanted proximity operator:

Case 1: $\|y^n + \sigma K\bar{x}^n\| \geq 1$.

In this case we have

$$\begin{aligned}
\operatorname{prox}_{\sigma F^*}(y^n + \sigma K\bar{x}^n) &= (y^n + \sigma K\bar{x}^n) - (y^n + \sigma K\bar{x}^n) \cdot \left(1 - \frac{1}{\|y^n + \sigma K\bar{x}^n\|_2}\right) \\
&= y^n + \sigma K\bar{x}^n - (y^n + \sigma K\bar{x}^n) + \frac{y^n + \sigma K\bar{x}^n}{\|y^n + \sigma K\bar{x}^n\|_2} \\
&= \frac{y^n + \sigma K\bar{x}^n}{\|y^n + \sigma K\bar{x}^n\|_2}.
\end{aligned}$$

Case 2: $\|y^n + \sigma K\bar{x}^n\| < 1$.

$$\begin{aligned}
\operatorname{prox}_{\sigma F^*}(y^n + \sigma K\bar{x}^n) &= (y^n + \sigma K\bar{x}^n) - (y^n + \sigma K\bar{x}^n) \cdot \max\{0, 1 - \frac{1}{\|y^n + \sigma K\bar{x}^n\|_2}\} \\
&= (y^n + \sigma K\bar{x}^n) - (y^n + \sigma K\bar{x}^n) \cdot 0 \\
&= (y^n + \sigma K\bar{x}^n).
\end{aligned}$$

Merging the two cases together, we can express the proximity operator in a more compact form:

$$\operatorname{prox}_{\sigma F^*}(y^n + \sigma K\bar{x}^n) = \frac{y^n + \sigma K\bar{x}^n}{\max\{1, \|y^n + \sigma K\bar{x}^n\|_2\}},$$

and remembering that we set $K = \nabla$, we have the following update rule:

$$y^{n+1} = \frac{y^n + \sigma \nabla\bar{x}^n}{\max\{1, \|y^n + \sigma \nabla\bar{x}^n\|_2\}}. \tag{3}$$

2.2 Find x^{n+1}

As opposed to the previous section, we try to find the proximity operator $\operatorname{prox}_{\tau G}(x^n - \tau K^* y^{n+1})$ directly by its definition. Note that K^* is not the Lagrange-Fenchel transform, but simply the *adjoint* operator of K. In our case, K is the gradient, and therefore, K^* is the negative of the divergence, $K^* = -\text{div}$. Starting with the definition of a proximity operator given in equation (2), we get

$$\operatorname{prox}_{\tau G}(x^n - \tau K^* y^{n+1}) = \arg\min_z \underbrace{\frac{1}{2}\|(x^n - \tau K^* y^{n+1}) - z\|_2^2 + \tau \frac{\lambda}{2} \|z - g\|_\Omega^2}_{=:\tilde{E}(z)}.$$

We try to find the minimum by setting the derivative with respect to z to zero:

$$\begin{aligned}
\nabla_z \tilde{E}(z) &= z - (x^n - \tau K^* y^{n+1}) + \tau\lambda\Omega(z-g) \\
&= z - x^n + \tau K^* y^{n+1} + \tau\lambda\Omega(z-g) \\
&= z - x^n + \tau K^* y^{n+1} + \tau\lambda\Omega z - \tau\lambda\Omega g \\
&= z(1+\tau\lambda\Omega) - x^n + \tau K^* y^{n+1} - \tau\lambda\Omega g \overset{!}{=} 0 \\
\iff z &= \frac{x^n - \tau K^* y^{n+1} + \tau\lambda\Omega g}{1+\tau\lambda\Omega}.
\end{aligned}$$

From this we get the update rule

$$x^{n+1} = \frac{x^n + \tau\nabla\cdot y^{n+1} + \tau\lambda\Omega g}{1+\tau\lambda\Omega}, \tag{4}$$

where $\nabla \cdot y^{n+1}$ denotes the divergence of y^{n+1}.

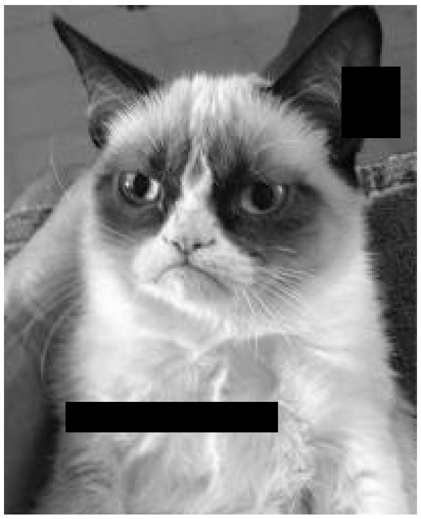
3 Results

3.1 Setting the parameters

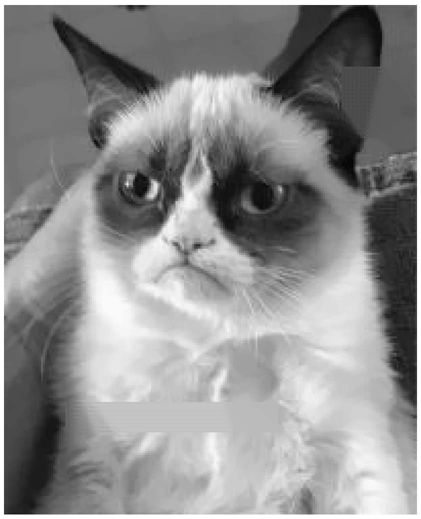
As proposed for the case of a denoising application on p.24 in [CP11], I used the value $\tau = 0.01$. It was already mentioned in Section 2, that $\tau\sigma\|K\|^2 < 1$ has to be fulfilled. I chose $\sigma = \frac{1}{0.09}$. This ensures that $\tau\sigma\|K\|^2 < 1$ since $\|K\|^2 \leq 8$ (for a proof of this, please consider [Cha04]).

3.2 Obtained images

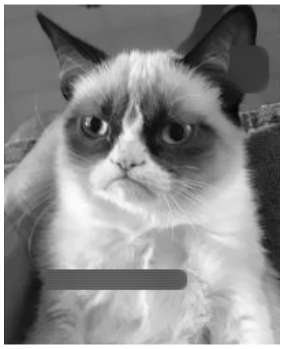
In Figure 1, there is an example of the obtained results. It's worth mentioning that there's not much of a difference between the image obtained with only $1'000$ iterations and the final output. We can also see that the method already converges very soon from the plot of the objective function E (see Figure 2), as opposed to the previously implemented gradient descent method that only converged after up to $10'000$ or even $100'000$ iterations, depending on the learning rate.



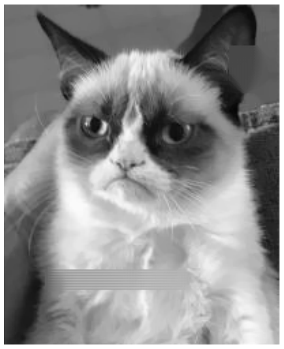
Input



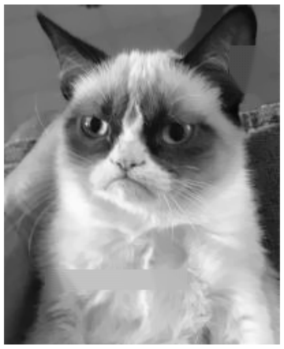
Output (3′000 iterations)



250 iterations



500 iterations



1′000 iterations

Fig. 1: The different states of the proceeding of the algorithm. Note that there's not much of a difference between the image obtained with only 1′000 iterations and the final output. I chose $\theta = 0.5$ and $\lambda = 100$. The other parameters were chosen as mentioned before.

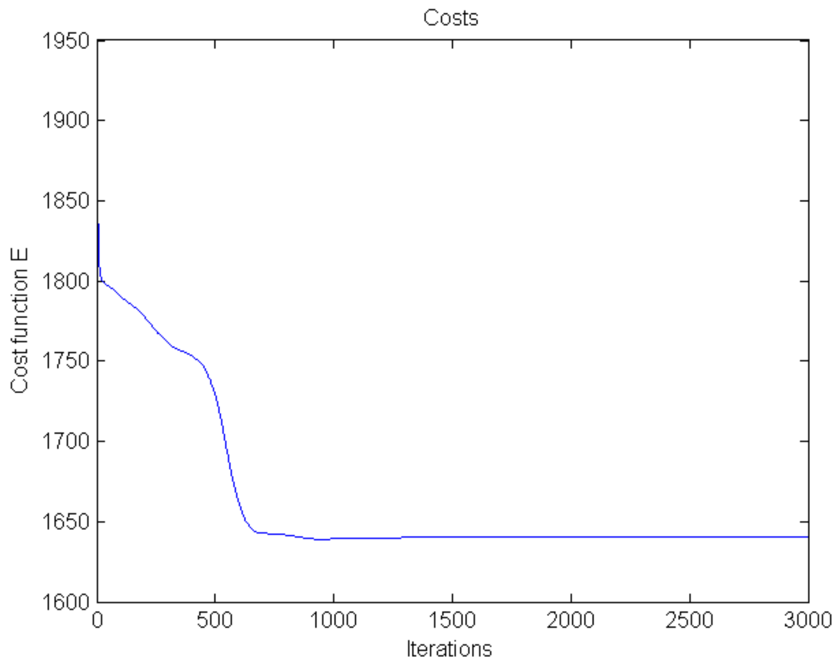


Fig. 2: Plot of the cost function E.

4 The effect of λ

In the objective function

$$E(u) = \frac{\lambda}{2}\|u - g\|_\Omega^2 + \|\nabla u\|_2$$

the parameter λ is a *regularization parameter*. It controls the tradeoff between the two constraints of fitting the image and keeping the result smooth. A very low λ, e.g. $\lambda = 0.1$ kind of allows the whole image to change. The smoothness contraint is in this case much more important. This will make the transitions between the originally available and the inpainted image regions nearly or completely invisible, but the overall image might get significantly oversmoothed. A very high λ-value may fail to end up in a nice-looking smooth image – in other words, it is likely that there are aprupt changes in the output image. Some examples obtained with different values for λ are shown in Figure 3. Note that λ does not heavily influence the number of iterations (as long as the values are not exremely small, the behaviour will stay more or less the same) that is necessary for the algorithm to converge. This was not the case in the gradient descent method, because due to large values it may have been necessary to set a lower learning rate.

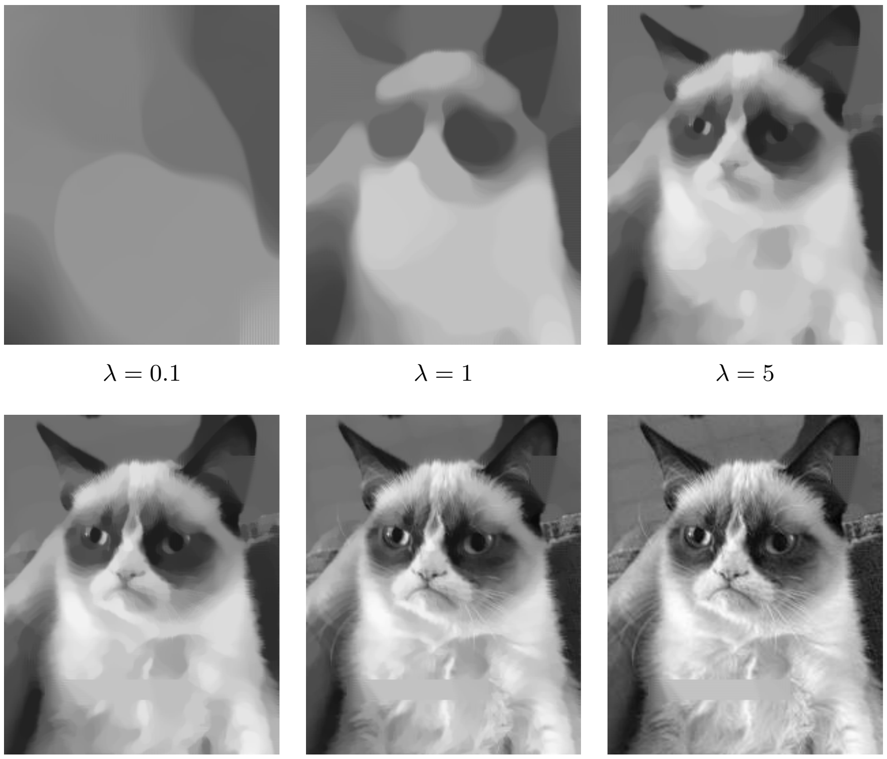


Fig. 3: Results with different values for λ. All the above images were obtained after $1'500$ iterations.

I was not able to find an optimal value for λ. As can be seen from Figure 4, the higher the value for λ, the smaller the error gets. However, first of all we see that after reaching a value of $\lambda \approx 100$, the result does not significantly improve with growing λ-values. Second, it's worth to mention that we can see from the update-rule (4), that the value of λ is in a way meaningless within inpainted regions, since Ω will be zero in those regions and therefore all the terms containing λ will cancel out (however, λ of course influences those regions indirectly through the surrounding image content, due to gradients and divergence). This is a possible explanation of why a really huge value for λ does not cause very bad results (as it was the case in gradient descent, where a big value could cause the whole algorithm to diverge) – the image regions that are originally available will be fitted as good as possible, whereas the inpainting regions do not further progress.

However, it is evident that those high values cause some gradient inconsistencies at the borders of the inpainting regions. It might be that the sum of squared distances it not the very best measure to capture this problem.

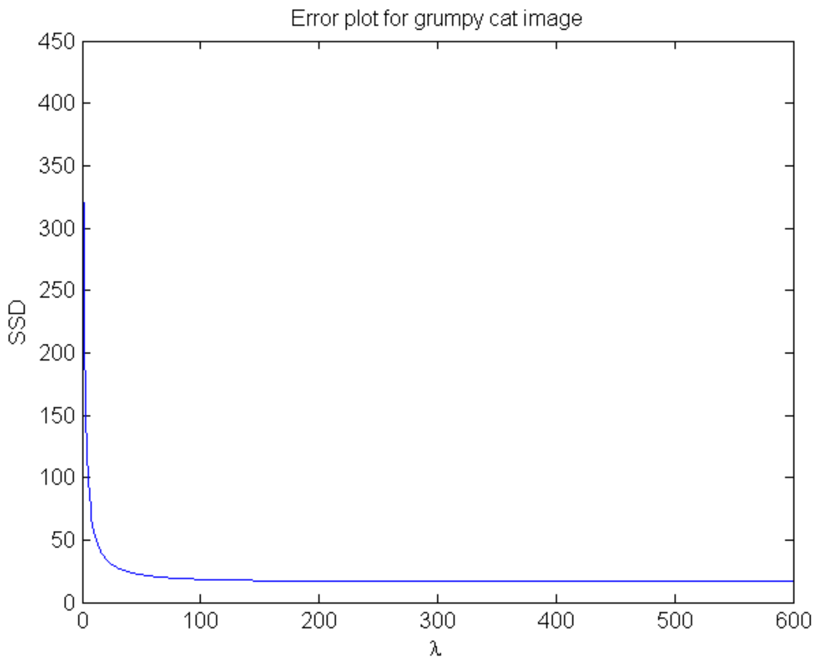


Fig. 4: λ vs. SSD plot. Once λ reached the value of 100, the error is not further reduced in a remarkable way.

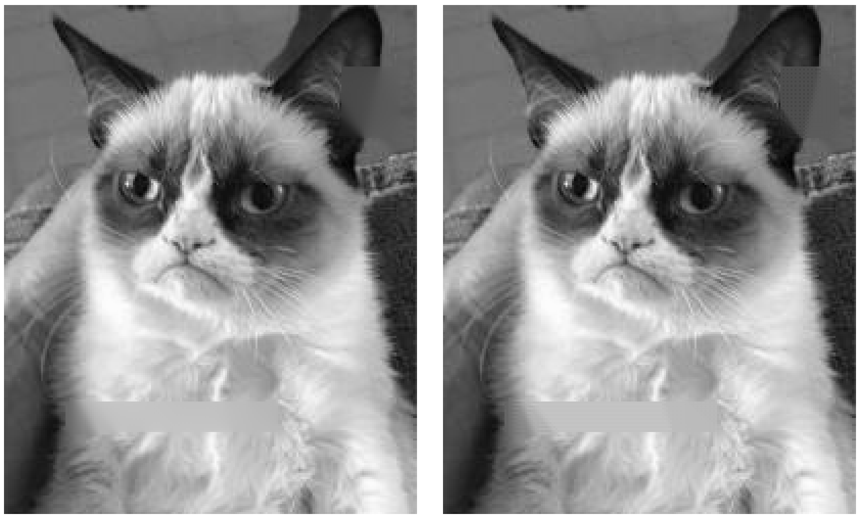
5 Conclusions

Whereas the gradient descent method is simple to implement, it might require a huge amount of iterations and a low learning rate to get to a reasonable result. The primal-dual method introduces some additional variables, the dual variables, that made us able to remove coupling of the variable due to the gradient. Thanks to that, the method converges remarkably faster than gradient descent, and, moreover, independent from the choice of λ.

The primal-dual algorithm also contains some tricky aspects like e.g. finding useful parameters τ, σ and θ. As was also mentioned in [CP11], it is not clear how to deal with operators with an unknown norm or even unbounded ones, since the parameters τ and σ do depend on that. The gradient descent method is much simpler to handle since the only thing that may have to be changed is the learning rate (and, possibly, depending on that, the number of iterations).

Also, the theoretical background needed to understand the primal-dual formulations and to understand why the proposed algorithm works, is way more advanced than the theory behind gradient descent, which may make it harder to reason about results and problems.

From the point of view of performance, the two methods are at a comparable level. In my cases, the primal-dual method was able to capture a bit more of the structures, as can be seen in Figure 5. This results may not be very representative since in both algorithms the choice of the parameters is probably not the optimal one.



Gradient descent (SSD = 15.0042)
100'000 iterations

Primal-dual (SSD = 16.4611)
1'500 iterations

Fig. 5: Comparison between the results obtained with gradient descent and the primal-dual algorithm, both applied with $\lambda = 1'000$. The ear of the cat is visibly more pleasing in the primal-dual result than gradient descent. The SSD to the ground truth, however, is lower with gradient descent.

References

[Cha04] Chambolle, Antonin: An algorithm for total variation minimization and applications. In: *Journal of Mathematical imaging and vision* 20 (2004), Nr. 1-2, S. 89–97

[CP11] Chambolle, Antonin ; Pock, Thomas: A first-order primal-dual algorithm for convex problems with applications to imaging. In: *Journal of Mathematical Imaging and Vision* 40 (2011), Nr. 1, S. 120–145