Chapter 2

Continuous Optimization in Computer Vision

Daniel Cremers and Bastian Goldlücke
Computer Vision Group
Technical University of Munich

Thomas Pock
Institute for Computer Graphics and Vision
Graz University of Technology

Overview

Energy minimization methods

- Typical energies consist of a regularization term and a data term

$$\min_u \{E(u) = \mathcal{R}(u) + \mathcal{D}(u, f)\} \ ,$$

  where f is the input data and u is the unknown solution
- Energy functional is designed such that low-energy states reflect the physical properties of the problem
- Minimizer provides the best (in the sense of the model) solution to the problem

Energy minimization methods

- Discrete MRF setting: Images are represented as graphs $\mathcal{G}(\mathcal{V}, \mathcal{E})$, consisting of a node set $\mathcal{V}$, and an edge set $\mathcal{E}$. Each node $v \in \mathcal{V}$ can take a label from a discrete label set $\mathcal{U} \subset \mathbb{Z}$, i.e. $u(v) \in \mathcal{U}$
- Continuous Variational setting: Images are considered as continuous functions $u : \Omega \to \mathbb{R}$, where $\Omega \subset \mathbb{R}^n$ is the image domain.
- Link to statistics: In a Bayesian setting, the energy relates to the posterior probability via a Gibbs distribution

$$p(u|f) = \frac{1}{Z} exp(-E(u))$$

- Computing the minimizer of $E(u)$ is equivalent to MAP estimation on $p(u|f)$

The posterior distribution

- $p(u|f)$ provides a complete distribution.
- Which sample should we selected?
- Expectation: Compute sample, that minimizes the squared distance to the distribution

$$\bar{u} = \frac{1}{Z} \int_u u\, p(u|f)\, \mathrm{d}u$$

- Needs subtle algorithms to approximate the integral (MCMC)
- MAP: Computing that sample that maximizes the probability

$$u^* = \arg\max_u p(u|f) = \arg\min_u E(u)$$

- Leads to well-defined optimization algorithms

Example: Total variation based image restoration

Image model: $f = k * u + n$, blur kernel $k =$ 

Variational model: [Rudin, Osher, Fatemi '92]

$$\min_u \int_\Omega |Du| + \frac{\lambda}{2} \|k * u - f\|_2^2$$

(a) Degraded image f

Example: Total variation based image restoration

Image model: $f = k * u + n$, blur kernel $k =$ 

Variational model: [Rudin, Osher, Fatemi '92]

$$\min_u \int_\Omega |Du| + \frac{\lambda}{2}\|k * u - f\|_2^2$$

(a) Degraded image f

(b) Reconstructed image u

We need to solve large scale optimization problems

Optimization problems are unsolvable

Consider the following general mathematical optimization problem:

$$\begin{aligned} &\min f_0(x) \\ \text{s.t.} \quad &f_i(x) \leq 0 \,, \quad i = 1 \ldots m \\ &x \in S \,, \end{aligned}$$

where $f_0(x) \ldots f_m(x)$ are real-valued functions, $x = (x_1, \ldots x_n)^T \in \mathbb{R}^n$ is a n-dimensional real-valued vector, and S is a subset of $\mathbb{R}^n$

How to solve this problem?

Optimization problems are unsolvable

Consider the following general mathematical optimization problem:

$$\begin{aligned} &\min f_0(x) \\ \text{s.t.} \quad &f_i(x) \leq 0 \,, \quad i = 1 \ldots m \\ &x \in S \,, \end{aligned}$$

where $f_0(x) \ldots f_m(x)$ are real-valued functions, $x = (x_1, \ldots x_n)^T \in \mathbb{R}^n$ is a n-dimensional real-valued vector, and S is a subset of $\mathbb{R}^n$

How to solve this problem?

- Naive: “Download a commercial package ...”

Optimization problems are unsolvable

Consider the following general mathematical optimization problem:

$$\begin{aligned} &\min f_0(x) \\ \text{s.t.} \quad &f_i(x) \leq 0 \,, \quad i = 1 \ldots m \\ &x \in S \,, \end{aligned}$$

where $f_0(x) \ldots f_m(x)$ are real-valued functions, $x = (x_1, \ldots x_n)^T \in \mathbb{R}^n$ is a n-dimensional real-valued vector, and S is a subset of $\mathbb{R}^n$

How to solve this problem?

- Naive: “Download a commercial package ...”
- Reality: “Finding a solution is far from being trivial!”
- “Optimization problems are unsolvable”

[Nesterov ’04]

Complexity bounds for global optimization (1)

- Consider the problem class $\mathcal{C}_0$ [Nesterov '04]

$$\min_{x \in \mathcal{B}_n} f(x) \ ,$$

where $\mathcal{B}_n$ is the n-dimensional unit box defined by

$$\mathcal{B}_n = \{x \in \mathbb{R}^n : 0 \leq x_i \leq 1, \ i = 1 \ldots n\}$$

Complexity bounds for global optimization (1)

- Consider the problem class $\mathcal{C}_0$ [Nesterov '04]

$$\min_{x \in \mathcal{B}_n} f(x) \ ,$$

where $\mathcal{B}_n$ is the n-dimensional unit box defined by

$$\mathcal{B}_n = \{x \in \mathbb{R}^n : 0 \leq x_i \leq 1, \ i = 1...n\}$$

- What is the lower complexity bound to find an ε - approximate solution?

Complexity bounds for global optimization (2)

A remark on Lipschitz continuity: A function $f(x)$ is called Lipschitz continuous on $\mathcal{B}_n$ if

$$|f(x) - f(y)| \leq L\|x - y\|_\infty, \ \forall x, y \in \mathcal{B}_n \ ,$$

where the constant L is an upper bound to the maximum steepness of $f(x)$

Complexity bounds for global optimization (2)

A remark on Lipschitz continuity: A function $f(x)$ is called Lipschitz continuous on $\mathcal{B}_n$ if

$$|f(x) - f(y)| \leq L\|x - y\|_\infty, \ \forall x, y \in \mathcal{B}_n \ ,$$

where the constant L is an upper bound to the maximum steepness of $f(x)$

- **Theorem 1.1.2** [Nesterov '04]
  For $\frac{1}{2}L > \varepsilon > 0$, the complexity of a zero-order method to find an ε - approximate solution of the problem class $\mathcal{C}_0$ is at least

$$\left(\left\lfloor \frac{L}{2\varepsilon} \right\rfloor\right)^n$$

Example

- Consider a general optimization problem with $n = 10$ unknowns and a moderate Lipschitz constant of $L = 2$
- We require an accuracy of let's say $\varepsilon = 1\%$

Example

- Consider a general optimization problem with $n = 10$ unknowns and a moderate Lipschitz constant of $L = 2$
- We require an accuracy of let's say $\varepsilon = 1\%$
- Results in at least 10^{20} function calls, that is, 31 250 000 years on a workstation!

Example

- Consider a general optimization problem with $n = 10$ unknowns and a moderate Lipschitz constant of $L = 2$
- We require an accuracy of let's say $\varepsilon = 1\%$
- Results in at least 10^{20} function calls, that is, 31 250 000 years on a workstation!
- If we change n to $n + 1$ the estimate is multiplied by $1/\varepsilon = 100$ and hence would take much longer

Example

- Consider a general optimization problem with $n = 10$ unknowns and a moderate Lipschitz constant of $L = 2$
- We require an accuracy of let's say $\varepsilon = 1\%$
- Results in at least 10^{20} function calls, that is, 31 250 000 years on a workstation!
- If we change n to $n + 1$ the estimate is multiplied by $1/\varepsilon = 100$ and hence would take much longer
- Contrary, if we change ε to 8% we would need only two weeks

Example

- Consider a general optimization problem with $n = 10$ unknowns and a moderate Lipschitz constant of $L = 2$
- We require an accuracy of let's say $\varepsilon = 1\%$
- Results in at least 10^{20} function calls, that is, 31 250 000 years on a workstation!
- If we change n to $n + 1$ the estimate is multiplied by $1/\varepsilon = 100$ and hence would take much longer
- Contrary, if we change ε to 8% we would need only two weeks
- Complexity bounds for higher order methods (gradient descend, Newton, ...) are not much better

Example

- Consider a general optimization problem with $n = 10$ unknowns and a moderate Lipschitz constant of $L = 2$
- We require an accuracy of let's say $\varepsilon = 1\%$
- Results in at least 10^{20} function calls, that is, 31 250 000 years on a workstation!
- If we change n to $n + 1$ the estimate is multiplied by $1/\varepsilon = 100$ and hence would take much longer
- Contrary, if we change ε to 8% we would need only two weeks
- Complexity bounds for higher order methods (gradient descend, Newton, ...) are not much better
- Comparison to NP-hard problems: Hard combinatorial problems need 2^n arithmetic operations (only)!

Convex versus non-convex

- Non-convex problems
  - Often give more accurate models
  - In general no chance to find the global minimizer
  - Result strongly depends on the initialization
  - Dilemma: Wrong model or wrong algorithm?

Convex versus non-convex

- Non-convex problems
  - Often give more accurate models
  - In general no chance to find the global minimizer
  - Result strongly depends on the initialization
  - Dilemma: Wrong model or wrong algorithm?

- Convex problems
  - Convex models often inferior
  - Any local minimizer is a global minimizer
  - Result is independent of the initialization
  - Note: Convex does not mean easy!

Convex versus non-convex

- Non-convex problems
  - Often give more accurate models
  - In general no chance to find the global minimizer
  - Result strongly depends on the initialization
  - Dilemma: Wrong model or wrong algorithm?

- Convex problems
  - Convex models often inferior
  - Any local minimizer is a global minimizer
  - Result is independent of the initialization
  - Note: Convex does not mean easy!

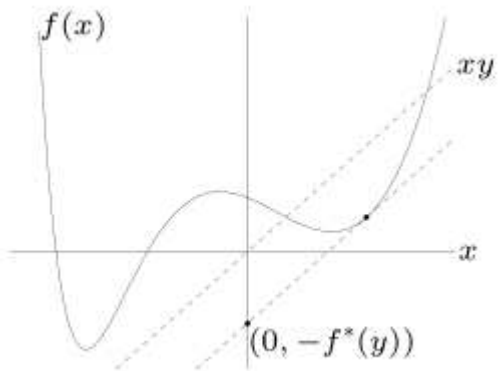
- Current research: Bridging the gap between convex and non-convex optimization
  - Convex approximations of non-convex models
  - New models
  - Algorithms
  - Bounds

Overview

The convex conjugate

- The convex conjugate $f^*(y)$ of a function $f(x)$ is defined through the Legendre-Fenchel transform

$$f^*(y) = \sup_{x \in \mathrm{dom}\ f} \langle x, y \rangle - f(x)$$



- $f^*(y)$ is a convex function (pointwise supremum over linear functions)

Examples

- $f(x) = |x|$:

$$f^*(y) = \sup_x \langle x, y \rangle - |x| = \begin{cases} 0 & \text{if} \quad |y| \leq 1 \\ \infty & \text{else} \end{cases}$$

- $f(x) = \frac{1}{2} x^T Q x$, Q, positive definite

$$f^*(y) = \sup_x \langle x, y \rangle - \frac{1}{2} x^T Q x = \frac{1}{2} y^T Q^{-1} y$$

The convex envelope

- The biconjugate function is defined by twice application of the Legendere-Fenchel transform

$$f^{**}(x) = \sup_{y \in \mathrm{dom}\, f^*} \langle x, y \rangle - f^*(y)$$

The convex envelope

- The biconjugate function is defined by twice application of the Legendere-Fenchel transform

$$f^{**}(x) = \sup_{y \in \mathrm{dom}\, f^*} \langle x, y \rangle - f^*(y)$$

- The biconjugate function $f^{**}(x)$ is the largest convex l.s.c. function below $f(x)$
- In case of uniqueness, $f(x)$ and $f^{**}(x)$ have the same minimum
- If $f(x)$ is a convex, l.s.c. function, $f^{**}(x) = f(x)$

$f(x)$ $\qquad$ $f^{**}(x)$

- Computing the biconjugate function allows to “convexify” a non-convex problem
- Unfortunately, computing $f^{**}(x)$ is not tractable for most problems

Approximate envelopes

- It is not possible to compute f^{**} in general (there are exceptions, though)
- What about approximate envelopes?
- Many problems can be decomposed as

$$f(x) = \sum_{i=1}^{n} f_i(x) \ ,$$

  where $f_i^{**}(x)$ is often much easier to compute than $f^{**}(x)$.
- It can be shown that

$$f(x) > f^{**}(x) = \left(\sum_i f_i \right)^{**} (x) > \left(\sum_i f_i^{**} \right) (x)$$

- This is exactly, what current research is about
- State-of-the art techniques (LP, QBPO, graph cuts, convex relaxations ...) exactly try to approximate the convex envelope in this way

Fenchel-Duality

- Consider the problem

$$\min_x f_1(x) - f_2(x)\,,$$

  where $f_{1,2}$ are real-valued functions
- Classical form of duality developed by W. Fenchel in the 40s

Fenchel-Duality

- Consider the problem

$$\min_x f_1(x) - f_2(x)\,,$$

  where $f_{1,2}$ are real-valued functions
- Classical form of duality developed by W. Fenchel in the 40s

- Consider the equivalent constrained problem

$$\min_{x_1,x_2} f_1(x_1) - f_2(x_2)\,, \quad \text{s.t. } x_1 = x_2$$

Fenchel-Duality

- Consider the problem

$$\min_x f_1(x) - f_2(x)\,,$$

  where $f_{1,2}$ are real-valued functions
- Classical form of duality developed by W. Fenchel in the 40s

- Consider the equivalent constrained problem

$$\min_{x_1,x_2} f_1(x_1) - f_2(x_2)\,, \quad \text{s.t. } x_1 = x_2$$

- Introducing Lagrange multipliers y for the constraint $x_1 = x_2$ yields

$$\min_{x_1,x_2} \max_y f_1(x_1) - f_2(x_2) + \langle x_1 - x_2, y \rangle$$

Fenchel-Duality

- Consider the problem

$$\min_x f_1(x) - f_2(x)\,,$$

  where $f_{1,2}$ are real-valued functions
- Classical form of duality developed by W. Fenchel in the 40s

- Consider the equivalent constrained problem

$$\min_{x_1,x_2} f_1(x_1) - f_2(x_2)\,, \quad \text{s.t. } x_1 = x_2$$

- Introducing Lagrange multipliers y for the constraint $x_1 = x_2$ yields

$$\min_{x_1,x_2} \max_y f_1(x_1) - f_2(x_2) + \langle x_1 - x_2, y \rangle$$

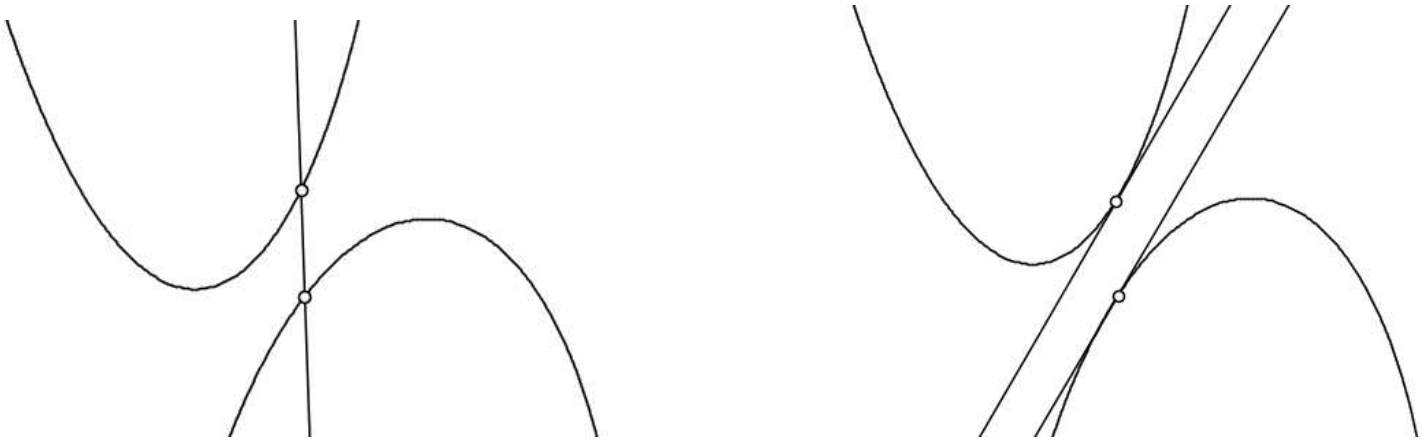
- We can recognize the definition of the convex conjugate functions, leads to the dual problem

$$\max_y f_2^*(y) - f_1^*(y)$$

Fenchel's duality theorem

If f_1 convex and f_2 concave, then

$$\min_x f_1(x) - f_2(x) = \max_y f_2^*(y) - f_1^*(y)$$



Interpretation: Minimizing the vertical distance between two points on the curves is equivalent to maximizing the distance between two parallel tangents on the curves

Overview

Convex optimization

- A general convex optimization problem is defined as

$$\begin{aligned} \min \quad & f_0(x) \\ \text{s.t.} \quad & f_i(x) \leq 0 \,, \quad i = 1 \dots m \\ & Ax = b \end{aligned}$$

where $f_0(x) \dots f_p(x)$ are real-valued convex functions,
$x = (x_1, \dots x_n)^T \in \mathbb{R}^n$ is a m-dimensional real-valued vector,
$Ax = b$ are affine equality constraints

Convex optimization

- A general convex optimization problem is defined as

$$\begin{aligned} \min \quad & f_0(x) \\ \text{s.t.} \quad & f_i(x) \leq 0 \,, \quad i = 1 \dots m \\ & Ax = b \end{aligned}$$

  where $f_0(x) \dots f_p(x)$ are real-valued convex functions,
  $x = (x_1, \dots x_n)^T \in \mathbb{R}^n$ is a m-dimensional real-valued vector,
  $Ax = b$ are affine equality constraints

- Convex optimization problems are considered as solvable!

Convex optimization

- A general convex optimization problem is defined as

$$\begin{aligned} \min \quad & f_0(x) \\ \text{s.t.} \quad & f_i(x) \leq 0 \,, \quad i = 1 \dots m \\ & Ax = b \end{aligned}$$

  where $f_0(x) \dots f_p(x)$ are real-valued convex functions,
  $x = (x_1, \dots x_n)^T \in \mathbb{R}^n$ is a m-dimensional real-valued vector,
  $Ax = b$ are affine equality constraints

- Convex optimization problems are considered as solvable!

- "The great watershed in optimization isn't between linearity and non-linearity, but convexity and non-convexity" [Rockafellar '93]

Black box convex optimization

Generic iterative algorithm for convex optimization:

1. Pick any initial vector $x^0 \in \mathbb{R}^n$, set $k = 0$
2. Compute search direction $d^k \in \mathbb{R}^n$
3. Choose step size τ^k such that $f(x^k + \tau^k d^k) < f(x^k)$
4. Set $x^{k+1} = x^k + \tau^k d^k$, $k = k + 1$
5. Stop if converged, else goto 2

Black box convex optimization

Generic iterative algorithm for convex optimization:

1. Pick any initial vector $x^0 \in \mathbb{R}^n$, set $k = 0$
2. Compute search direction $d^k \in \mathbb{R}^n$
3. Choose step size τ^k such that $f(x^k + \tau^k d^k) < f(x^k)$
4. Set $x^{k+1} = x^k + \tau^k d^k$, $k = k + 1$
5. Stop if converged, else goto 2

Different methods to determine the search direction d^k

- steepest descend
- conjugate gradients
- Newton, quasi-Newton
- Working-horse for the lazy: Limited memory BFGS quasi-Newton method [Nocedal '80]

Black box convex optimization

Generic iterative algorithm for convex optimization:

1. Pick any initial vector $x^0 \in \mathbb{R}^n$, set $k = 0$
2. Compute search direction $d^k \in \mathbb{R}^n$
3. Choose step size τ^k such that $f(x^k + \tau^k d^k) < f(x^k)$
4. Set $x^{k+1} = x^k + \tau^k d^k$, $k = k + 1$
5. Stop if converged, else goto 2

Different methods to determine the search direction d^k

- steepest descend
- conjugate gradients
- Newton, quasi-Newton
- Working-horse for the lazy: Limited memory BFGS quasi-Newton method [Nocedal '80]

Black box methods do not exploit the structure of the problem and hence are often less effective

Structured convex optimization

There are a number of efficient solvers available for important types of structured convex optimization problems [Boyd, Vandenberghe '04]:

Structured convex optimization

There are a number of efficient solvers available for important types of structured convex optimization problems [Boyd, Vandenberghe '04]:

- Least squares problems:

$$\min_x \|Ax - b\|_2^2$$

Structured convex optimization

There are a number of efficient solvers available for important types of structured convex optimization problems [Boyd, Vandenberghe '04]:

- Least squares problems:

$$\min_x \|Ax - b\|_2^2$$

- Linear program (LP)

$$\begin{aligned} &\min c^T x, \\ &\text{s.t. } Ax = b,\ x \geq 0 \end{aligned}$$

Structured convex optimization

There are a number of efficient solvers available for important types of structured convex optimization problems [Boyd, Vandenberghe '04]:

- Least squares problems:

$$\min_x \|Ax - b\|_2^2$$

- Linear program (LP)

$$\begin{aligned} &\min c^T x, \\ &\text{s.t. } Ax = b,\ x \geq 0 \end{aligned}$$

- Quadratic program (QP)

$$\begin{aligned} &\min \tfrac{1}{2} x^T Q x + c^T x,\ Q \succeq 0 \\ &\text{s.t. } Ax = b,\ l \leq x \leq u \end{aligned}$$

Structured convex optimization

There are a number of efficient solvers available for important types of structured convex optimization problems [Boyd, Vandenberghe '04]:

- Least squares problems:

$$\min_x \|Ax - b\|_2^2$$

- Linear program (LP)

$$\begin{aligned} &\min c^T x, \\ &\text{s.t. } Ax = b,\ x \geq 0 \end{aligned}$$

- Quadratic program (QP)

$$\begin{aligned} &\min \tfrac{1}{2} x^T Q x + c^T x,\ Q \succeq 0 \\ &\text{s.t. } Ax = b,\ l \leq x \leq u \end{aligned}$$

- ...
- Many problems can be formulated in these frameworks
- Fast methods available (simplex, interior point, ...)
- Sometimes ineffective for large-scale problems

Overview

A class of problems

Let us consider the following class of structured convex optimization problems

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

- $K : X \to Y$ is a linear and continuous operator from a Hilbert space X to a Hilbert space Y.
- $F : Y \to \mathbb{R} \cup \{\infty\}$, $G : X \to \mathbb{R} \cup \{\infty\}$ are "simple" convex, proper, l.s.c. functions, and hence have an easy to compute prox operator:

$$\text{prox}_G(z) = (I + \partial G)^{-1}(z) = \arg\min_x \frac{\|x - z\|^2}{2} + G(x)$$

A class of problems

Let us consider the following class of structured convex optimization problems

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

- $K : X \to Y$ is a linear and continuous operator from a Hilbert space X to a Hilbert space Y.
- $F : Y \to \mathbb{R} \cup \{\infty\}$, $G : X \to \mathbb{R} \cup \{\infty\}$ are "simple" convex, proper, l.s.c. functions, and hence have an easy to compute prox operator:

$$\text{prox}_G(z) = (I + \partial G)^{-1}(z) = \arg\min_x \frac{\|x - z\|^2}{2} + G(x)$$

- It turns out that many standard problems can be cast in this framework.
- There exists a vast literature of numerical algorithms to solve this class of problems

Examples

- Image restoration: The ROF model

$$\min_u \|\nabla u\|_1 + \frac{\lambda}{2}\|k * u - f\|_2^2 \,,$$

- Compressed sensing: Basis pursuit problem (LASSO)

$$\min_x \|x\|_1 + \frac{\lambda}{2}\|Ax - b\|_2^2$$

Examples

- Image restoration: The ROF model

$$\min_u \|\nabla u\|_1 + \frac{\lambda}{2}\|k * u - f\|_2^2 \,,$$

- Compressed sensing: Basis pursuit problem (LASSO)

$$\min_x \|x\|_1 + \frac{\lambda}{2}\|Ax - b\|_2^2$$

- Machine learning: Linear support vector machine

$$\min_{w,b} \frac{\lambda}{2}\|w\|_2^2 + \sum_{i=1}^{n} \max\left(0, 1 - y_i\left(\langle w, x_i\rangle + b\right)\right)$$

Examples

- Image restoration: The ROF model

$$\min_u \|\nabla u\|_1 + \frac{\lambda}{2}\|k * u - f\|_2^2 \,,$$

- Compressed sensing: Basis pursuit problem (LASSO)

$$\min_x \|x\|_1 + \frac{\lambda}{2}\|Ax - b\|_2^2$$

- Machine learning: Linear support vector machine

$$\min_{w,b} \frac{\lambda}{2}\|w\|_2^2 + \sum_{i=1}^{n} \max\left(0, 1 - y_i\left(\langle w, x_i\rangle + b\right)\right)$$

- General linear programming problems

$$\min_x \langle c, x\rangle \,, \text{ s.t. } \begin{cases} Ax &= b \\ x &\geq 0 \end{cases}$$

Primal, dual, primal-dual

The real power of convex optimization comes through duality

Recall the convex conjugate:

$$F^*(y) = \sup_{x \in X} \langle x, y \rangle - F(x) \,, \quad F^{**}(x) = \sup_{y \in Y} \langle x, y \rangle - F^*(y)$$

If $F(x)$ convex, l.s.c. than $F^{**}(x) = F(x)$

Primal, dual, primal-dual

The real power of convex optimization comes through duality

Recall the convex conjugate:

$$F^*(y) = \sup_{x \in X} \langle x, y \rangle - F(x) \,, \quad F^{**}(x) = \sup_{y \in Y} \langle x, y \rangle - F^*(y)$$

If $F(x)$ convex, l.s.c. than $F^{**}(x) = F(x)$

$$\min_{x \in X} F(Kx) + G(x) \quad \text{(Primal)}$$

Primal, dual, primal-dual

The real power of convex optimization comes through duality

Recall the convex conjugate:

$$F^*(y) = \sup_{x \in X} \langle x, y \rangle - F(x) \,, \quad F^{**}(x) = \sup_{y \in Y} \langle x, y \rangle - F^*(y)$$

If $F(x)$ convex, l.s.c. than $F^{**}(x) = F(x)$

$$\min_{x \in X} F(Kx) + G(x) \quad \text{(Primal)}$$

$$\min_{x \in X} \max_{y \in Y} \langle Kx, y \rangle + G(x) - F^*(y) \quad \text{(Primal-Dual)}$$

Primal, dual, primal-dual

The real power of convex optimization comes through duality

Recall the convex conjugate:

$$F^*(y) = \sup_{x \in X} \langle x, y \rangle - F(x) \,, \quad F^{**}(x) = \sup_{y \in Y} \langle x, y \rangle - F^*(y)$$

If $F(x)$ convex, l.s.c. than $F^{**}(x) = F(x)$

$$\min_{x \in X} F(Kx) + G(x) \quad \text{(Primal)}$$

$$\min_{x \in X} \max_{y \in Y} \langle Kx, y \rangle + G(x) - F^*(y) \quad \text{(Primal-Dual)}$$

$$\max_{y \in Y} -(F^*(y) + G^*(-K^* y)) \quad \text{(Dual)}$$

Allows to compute the duality gap, which is a measure of optimality

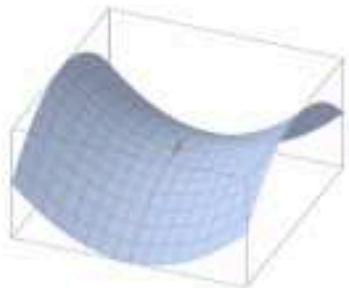
Optimality conditions

We focus on the primal-dual formulation:

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

We assume, there exists a saddle-point $(\hat{x}, \hat{y}) \in X \times Y$ which satisfies the Euler-Lagrange equations

$$\begin{cases} K\hat{x} - \partial F^*(\hat{y}) & \ni 0 \\ K^*\hat{y} + \partial G(\hat{x}) & \ni 0 \end{cases}$$



A note on Forward-backward splitting

- Consider the problem $\min_x f_1(x) + f_2(x)$, where $f_1(x)$ is a convex function and $f_2(x)$ is a convex function with L-Lipschitz continuous gradient ∇f_2, i.e.

$$\|\nabla f_2(x) - \nabla f_2(y)\| \leq L\|x - y\| \ , \forall x, y \in \mathrm{dom}(f_2)$$

- It can be shown that a minimizer can be characterized by the fixed point equation[Combettes, Pesquet '05]

$$x = (I + \tau \partial f_1)^{-1}(x - \tau \nabla f_2(x))$$

- This suggests the forward-backward iterations

$$x^{n+1} = \underbrace{(I + \tau \partial f_1)^{-1}}_{\text{backward step}} \underbrace{(x^n - \tau \nabla f_2(x^n))}_{\text{forward step}}$$

- **Note:** This is exactly the case for our saddle-point formulation

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

Arrow-Hurwicz algorithm

- Classical method for finding saddle-points [Arrow, Hurwicz, Uzawa '58]

$$\begin{cases} x^{n+1} = (I + \tau \partial G)^{-1}(x^n - \tau K^* y^n) \\ y^{n+1} = (I + \tau \partial F^*)^{-1}(y^n + \tau K x^{n+1}) \end{cases}$$

- Alternating forward-backward step in the primal variable and the dual variable
- Convergence under quite restrictive assumptions on τ
- For some problems very fast, e.g. ROF problem using adaptive time steps [Zhu, Chan, '08]

Proximal point algorithm

- Very general algorithm with well-studied properties [Martinet '70], [Rockafellar '76]
- Searches the zero of an operator, i.e. $0 \in T$
- General algorithm

$$p^{n+1} = (I + \tau^n T)^{-1} p^n$$

- In our setting:

$$p = \begin{pmatrix} x \\ y \end{pmatrix}, \quad T = \begin{pmatrix} \partial G & K^* \\ -K & \partial F^* \end{pmatrix}$$

- "Not implementable" since in most interesting cases, T is very hard to invert
- Many "implementable" algorithms can be shown to be a special case of the proximal point algorithm

Other interesting algorithms

- Extragradient methods [Korpelevich '76], [Popov '80]
  - Similar to Arrow-Hurwicz, but twice evaluation of the proximal operators
- Algorithms proposed by Yurii Nesterov
  - Many different algorithms, with guaranteed convergence rates, e.g. [Nesterov '03]
  - Recently, an increased interest in signal processing, computer vision and machine learning
- Douglas-Rachford splitting (DRS): [Mercier, Lions '79]
  - Can be applied if the problem can be decomposed into simpler problems
  - Special case of the proximal-point algorithm [Eckstein, Bertsekas '89]
  - Equivalent to an alternating minimization (ADMM) on the augmented Lagrangian formulation [Eckstein, Bertsekas '89]
  - Equivalent to the split-Bregman algorithm [Goldstein, Osher '09]

Other interesting algorithms

- Iterative shrinkage thresholding algorithm (ISTA) [Daubechies, Defrise, De Mol '04]
  - Developed for sparse recovery
  - Fast iterative shrinkage thresholding algorithm (FISTA) [Beck, Teboulle '09]
- Proximal splitting algorithms, e.g. [Combettes, Pesquet, '08]
  - Many different algorithms spezialized to interesting problems in image processing and sparse recovery

A simple primal-dual algorithm

Proposed in a series of papers: [Pock, Cremers, Bischof, Chambolle, '09], [Chambolle, Pock, '10], [Pock, Chambolle, '11]

- Initialization: Choose s.p.d. T, Σ, $\theta \in [0, 1]$, $(x^0, y^0) \in X \times Y$.
- Iterations ($n \geq 0$): Update x^n, y^n as follows:

$$\begin{cases} x^{n+1} &= (I + \mathrm{T}\partial G)^{-1}(x^n - \mathrm{T}K^* y^n) \\ y^{n+1} &= (I + \Sigma\partial F^*)^{-1}(y^n + \Sigma K(x^{n+1} + \theta(x^{n+1} - x^n))) \end{cases}$$

- Alternates gradient descend in x and gradient ascend in y
- Linear extrapolation of iterates of x in the y step
- T, Σ can be seen as preconditioning matrices

A simple primal-dual algorithm

Proposed in a series of papers: [Pock, Cremers, Bischof, Chambolle, '09], [Chambolle, Pock, '10], [Pock, Chambolle, '11]

- Initialization: Choose s.p.d. T, Σ, $\theta \in [0, 1]$, $(x^0, y^0) \in X \times Y$.
- Iterations ($n \geq 0$): Update x^n, y^n as follows:

$$\begin{cases} x^{n+1} &= (I + \mathrm{T}\partial G)^{-1}(x^n - \mathrm{T}K^* y^n) \\ y^{n+1} &= (I + \Sigma\partial F^*)^{-1}(y^n + \Sigma K(x^{n+1} + \theta(x^{n+1} - x^n))) \end{cases}$$

- Alternates gradient descend in x and gradient ascend in y
- Linear extrapolation of iterates of x in the y step
- T, Σ can be seen as preconditioning matrices

- Can be derived from a pre-conditioned ADMM algorithm
- Can be seen as a relaxed Arrow-Hurwicz scheme
- Can be seen as an approximate extragradient scheme

Relations to the proximal-point algorithm

- The iterations of PD can be written as the variational inequality [He,Yuan '10]

$$\left\langle \begin{pmatrix} x - x^{n+1} \\ y - y^{n+1} \end{pmatrix}, \begin{pmatrix} \partial G(x^{n+1}) + K^* y^{n+1} \\ \partial F^*(y^{n+1}) - K x^{n+1} \end{pmatrix} + M \begin{pmatrix} x^{n+1} - x^n \\ y^{n+1} - y^n \end{pmatrix} \right\rangle \geq 0 \,,$$

$$M = \begin{bmatrix} \mathrm{T}^{-1} & -K^* \\ -\theta K & \Sigma^{-1} \end{bmatrix}$$

Relations to the proximal-point algorithm

- The iterations of PD can be written as the variational inequality [He,Yuan '10]

$$\left\langle \begin{pmatrix} x - x^{n+1} \\ y - y^{n+1} \end{pmatrix}, \begin{pmatrix} \partial G(x^{n+1}) + K^* y^{n+1} \\ \partial F^*(y^{n+1}) - K x^{n+1} \end{pmatrix} + M \begin{pmatrix} x^{n+1} - x^n \\ y^{n+1} - y^n \end{pmatrix} \right\rangle \geq 0 \,,$$

$$M = \begin{bmatrix} \mathrm{T}^{-1} & -K^* \\ -\theta K & \Sigma^{-1} \end{bmatrix}$$

- This is exactly the proximal-point algorithm, but with a norm in M.

Relations to the proximal-point algorithm

- The iterations of PD can be written as the variational inequality [He,Yuan '10]

$$\left\langle \begin{pmatrix} x - x^{n+1} \\ y - y^{n+1} \end{pmatrix}, \begin{pmatrix} \partial G(x^{n+1}) + K^* y^{n+1} \\ \partial F^*(y^{n+1}) - K x^{n+1} \end{pmatrix} + M \begin{pmatrix} x^{n+1} - x^n \\ y^{n+1} - y^n \end{pmatrix} \right\rangle \geq 0 \,,$$

$$M = \begin{bmatrix} \mathrm{T}^{-1} & -K^* \\ -\theta K & \Sigma^{-1} \end{bmatrix}$$

- This is exactly the proximal-point algorithm, but with a norm in M.
- Convergence is ensured if M is symmetric and positive definite

Relations to the proximal-point algorithm

- The iterations of PD can be written as the variational inequality [He,Yuan '10]

$$\left\langle \begin{pmatrix} x - x^{n+1} \\ y - y^{n+1} \end{pmatrix}, \begin{pmatrix} \partial G(x^{n+1}) + K^* y^{n+1} \\ \partial F^*(y^{n+1}) - K x^{n+1} \end{pmatrix} + M \begin{pmatrix} x^{n+1} - x^n \\ y^{n+1} - y^n \end{pmatrix} \right\rangle \geq 0 \,,$$

$$M = \begin{bmatrix} \mathrm{T}^{-1} & -K^* \\ -\theta K & \Sigma^{-1} \end{bmatrix}$$

- This is exactly the proximal-point algorithm, but with a norm in M.
- Convergence is ensured if M is symmetric and positive definite
- This is the case if $\theta = 1$ and the assertion $\| \Sigma^{\frac{1}{2}} K \mathrm{T}^{\frac{1}{2}} \|^2 < 1$ is fulfilled.

A family of diagonal preconditioners

- It is important to choose T, Σ such that the prox-operators are still easy to compute
- Restrict the preconditioning matrices to diagonal matrices
- It turns out that: Let $K \in \mathbb{R}^{m \times n}$, $\mathrm{T} = \mathrm{diag}(\boldsymbol{\tau})$ and $\Sigma = \mathrm{diag}(\boldsymbol{\sigma})$ such that

$$\tau_j = \frac{1}{\sum_{i=1}^{m} |K_{i,j}|^{2-\alpha}}, \quad \sigma_i = \frac{1}{\sum_{j=1}^{n} |K_{i,j}|^{\alpha}}$$

then for any $\alpha \in [0, 2]$

$$\|\Sigma^{\frac{1}{2}} K \mathrm{T}^{\frac{1}{2}}\|^2 \leq 1 \,.$$

- Gives an automatic problem-dependent choice of the primal and dual steps
- Allows to apply the algorithm in a plug-and-play fashion
- For $\alpha = 0$, equivalent to the alternating step method [Eckstein, Bertsekas, '90]

Convergence rates

The algorithm gives different convergence rates on different problem classes [Chambolle, Pock, '10]

- F^* and G nonsmooth: $O(1/N)$

Convergence rates

The algorithm gives different convergence rates on different problem classes [Chambolle, Pock, '10]

- F^* and G nonsmooth: $O(1/N)$
- F^* or G uniformly convex: $O(1/N^2)$

Convergence rates

The algorithm gives different convergence rates on different problem classes [Chambolle, Pock, '10]

- F^* and G nonsmooth: $O(1/N)$
- F^* or G uniformly convex: $O(1/N^2)$
- F^* and G uniformly convex: $O(\omega^N)$, $\omega < 1$

Convergence rates

The algorithm gives different convergence rates on different problem classes [Chambolle, Pock, '10]

- F^* and G nonsmooth: $O(1/N)$
- F^* or G uniformly convex: $O(1/N^2)$
- F^* and G uniformly convex: $O(\omega^N)$, $\omega < 1$

- Coincides with so far best known rates of first-order methods

Parallel computing?

The algorithm basically computes matrix-vector products
The matrices are usually very sparse

Parallel computing?

The algorithm basically computes matrix-vector products
The matrices are usually very sparse

- One simple processor for each unknown
- Each processor has a small amount of local memory
- According to the structure of K, each processor can inter-change data with its neighboring processors

Parallel computing?

The algorithm basically computes matrix-vector products
The matrices are usually very sparse

- One simple processor for each unknown
- Each processor has a small amount of local memory
- According to the structure of K, each processor can inter-change data with its neighboring processors

Recent GPUs already go into this direction

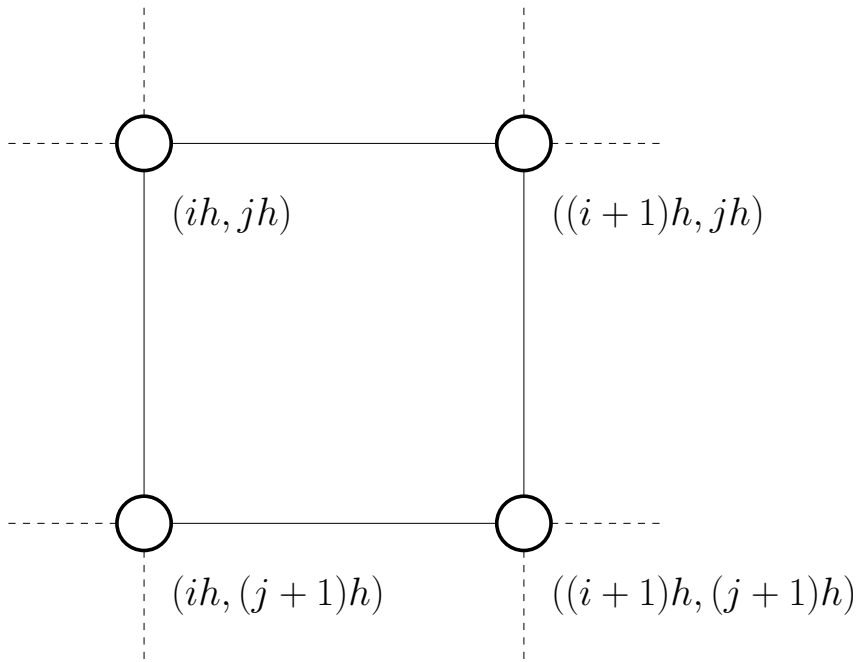
Overview

Discrete grid

We consider a regular Cartesian grid of size $M \times N$:

$$\Omega_h = \{(ih, jh) \,:\, 1 \leq i \leq M, 1 \leq j \leq N\}\,,$$

where h denotes the size of the spacing and (i, j) denote the indices of the discrete locations (ih, jh) in the image domain.



Discrete gradient

The gradient ∇u is a vector in the vector space $Y = X \times X$. For discretization of $\nabla : X \to Y$, we use standard finite differences with Neumann boundary conditions

$$(\nabla u)_{i,j} = \begin{pmatrix} (\nabla u)^1_{i,j} \\ (\nabla u)^2_{i,j} \end{pmatrix} ,$$

where

$$(\nabla u)^1_{i,j} = \begin{cases} \frac{u_{i+1,j} - u_{i,j}}{h} & \text{if } i < M \\ 0 & \text{if } i = M \end{cases}, \quad (\nabla u)^2_{i,j} = \begin{cases} \frac{u_{i,j+1} - u_{i,j}}{h} & \text{if } j < N \\ 0 & \text{if } j = N \end{cases} .$$

Furthermore we will also need the discrete divergence operator $\operatorname{div} p : Y \to X$, which is chosen to be adjoint to the discrete gradient operator. In particular, one has $-\operatorname{div} = \nabla^*$ which is defined through the identity

$$\langle \nabla u, p \rangle_Y = - \langle u, \operatorname{div} p \rangle_X .$$

Preconditioner: We easily find that $\mathrm{T} = \operatorname{diag}(\tau)$, where $\tau = h/4$ and $\Sigma = \operatorname{diag}(\sigma)$, where $\sigma = h/2$

The discrete ROF model

In the continuous setting, the ROF model is defined as

$$\min_u \int_\Omega |Du| + \frac{\lambda}{2} \|u - g\|_2^2$$

Using the discrete setting, we obtain

$$h^2 \min_{u \in X} \|\nabla u\|_1 + \frac{\lambda}{2} \|u - g\|_2^2 \,,$$

where the discrete total variation is given by

$$\|\nabla u\|_1 = \sum_{i,j} |(\nabla u)_{i,j}| \,, \quad |(\nabla u)_{i,j}| = \sqrt{((\nabla u)_{i,j}^1)^2 + ((\nabla u)_{i,j}^2)^2}$$

and

$$\|u - g\|_2^2 = \sum_{i,j} (u_{i,j} - g_{i,j})^2$$

The parameter λ is used to control the influence of regularization

The primal-dual ROF model

Casting the discrete ROF model in our framework, we obtain the following primal-dual ROF model

$$\min_{u \in X} \max_{p \in Y} -\langle u, \operatorname{div} p \rangle_X + \frac{\lambda}{2} \|u - g\|_2^2 - \delta_P(p) \,,$$

where the convex set P is given by

$$P = \{p \in Y : \|p\|_\infty \leq 1\} \,,$$

and $\|p\|_\infty$ denotes the discrete maximum norm defined as

$$\|p\|_\infty = \max_{i,j} |p_{i,j}| \,, \quad |p_{i,j}| = \sqrt{(p_{i,j}^1)^2 + (p_{i,j}^2)^2} \,.$$

Primal-dual algorithm to solve the ROF model

ROF-Primal-Dual

- Initialization: Choose $\tau = h/4$, $\sigma = h/2$, set $u^0 = \bar{u}^0 = f$, $p^0 = 0$, $\theta = 1$
- Iterations ($n \geq 0$)

$$\begin{cases} u^{n+1} &= (I + \tau \partial G)^{-1}(u^n - \tau(-\operatorname{div} p^n)) \\ p^{n+1} &= (I + \sigma \partial F^*)^{-1}(p^n + \sigma \nabla(2u^{n+1} - u^n)) \end{cases}$$

Computing the prox operators

Finally, we need to find explicit expressions for $(I + \sigma\partial F^*)^{-1}$ and $(I + \tau\partial G)^{-1}$. We have that $F^*(p) = \delta_P(p)$ and $G(u) = \frac{\lambda}{2}\|u - g\|_2^2$. Since F^* is the indicator function of a convex set, the resolvent operator reduces to pointwise Euclidean projectors onto L^2 balls

$$p = (I + \sigma\partial F^*)^{-1}(\tilde{p}) \Longleftrightarrow p_{i,j} = \frac{\tilde{p}_{i,j}}{\max(1, |\tilde{p}_{i,j}|)} \; .$$

The resolvent operator with respect to G poses simple pointwise quadratic problems. The solution is trivially given by

$$u = (I + \tau\partial G)^{-1}(\tilde{u}) \Longleftrightarrow u_{i,j} = \frac{\tilde{u}_{i,j} + \tau\lambda g_{i,j}}{1 + \tau\lambda} \; .$$

Practical implementation

- Matlab implementation → **Online-Demo**
  - Uses Matlab vector operations
  - Takes a few seconds for typical images
  - About 100 lines of code
  - Files can be downloaded from the Tutorial homepage
- Parallel implementation → **Online-Demo**
  - GPU implementation using Nvidia CUDA
  - Matlab fronted
  - About 1000 lines of code
  - Depending on the GPU 100-1000 times faster

(a) Clean (b) Noisy (c) Denoised

ROF-denoising on the GPU

Linear programming

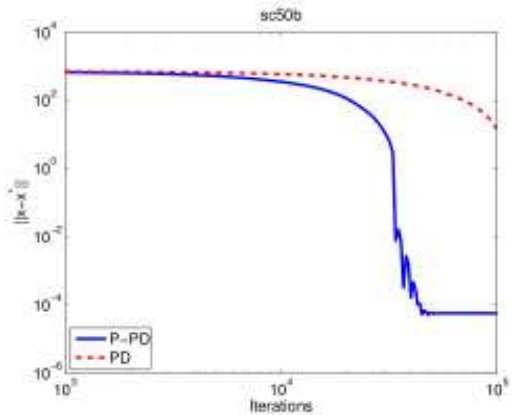
- Many applications of LP relaxations in computer vision and machine learning
- LP program in equality form

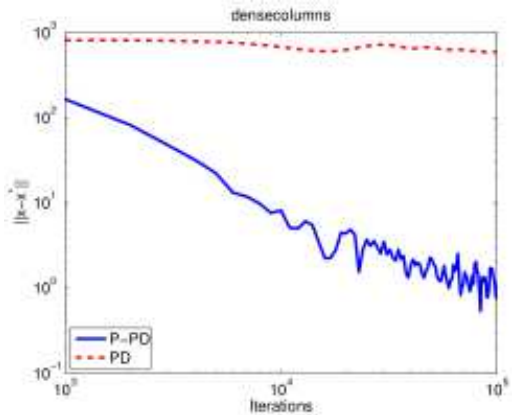
$$\min c^T x \quad \text{s.t.} \quad Ax = b\,,\; x \geq 0\,, \tag{1}$$

- Preconditioned primal-dual algorithm

$$\begin{cases} x^{k+1} = \text{proj}_{[0,\infty)}\left(x^k - \text{T}(A^T y^k + c)\right) \\ y^{k+1} = y^k + \Sigma(A(2x^{k+1} - x^k) - b)\,, \end{cases} \tag{2}$$

- Comparison





Graph cuts

- Graph cuts are widely used in computer vision [3]
- Can be written as a weighted total variation energy

$$\min_u \|D_w u\|_{\ell_1} + \langle u, w^u \rangle \ , \ \text{s.t. } 0 \leq u \leq 1 \ , \tag{3}$$

- Preconditioned primal-dual algorithm

$$\begin{cases} u^{k+1} = \text{proj}_{[0,1]} \left(u^k + \text{T}(D_w^T y^k - w^u) \right) \\ y^{k+1} = \text{proj}_{[-1,1]} \left(y^k + \Sigma(D_w(2u^{k+1} - u^k)) \right) \ , \end{cases} \tag{4}$$

- Comparison

MAXFLOW	PD	P-PD	P-PD-GPU
0.160s	15.75s	8.56s	0.045s

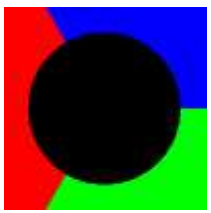
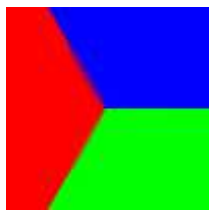
Continuous Potts model

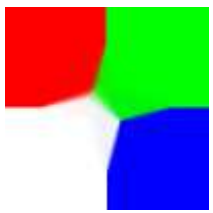
- The Continuous Potts model [4] with k phases can be written as a convex problem

$$\min_{u \in S} \max_{q \in B} \sum_{l=1}^{k} \langle Du_l, q_l \rangle + \langle u_l, f_l \rangle \quad , \tag{5}$$

where S is the simplex constraint and B is an interection of non-local ℓ_2 balls.

- Comparison



	PD	P-PD	Speedup
Synthetic (3 phases)	221.71s	75.65s	2.9
Synthetic (4 phases)	1392.02s	538.83s	2.6
Natural (8 phases)	592.85s	113.76s	5.2

Summary and open questions

- Energy minimization methods for computer vision
- Leads to large scale optimization problems
- Introduction into convex analysis and convex optimization
- Flexible primal-dual algorithm for a class of structured convex optimization problems

Summary and open questions

- Energy minimization methods for computer vision
- Leads to large scale optimization problems
- Introduction into convex analysis and convex optimization
- Flexible primal-dual algorithm for a class of structured convex optimization problems

- Can first-order methods go beyond the shown convergence rates?
- Combining first- and second-order methods?
- Discrete versus continuous optimization methods?