Lecture 4: Stat 238. Winter. 2012

A.L. Yuille

2012-01-20

1 Lecture 4

1. Models with Spatial Context.
2. Mumford and Shah.
3. Rudin, Osher, Fatemi. TV-norm.
4. Convexity. Steepest Descent.
5. Variational Bounding and CCCP.
6. L1 norm versus L2 sparsness.
7. Appendix I: Mumford-Shah and Ambrosio-Tortorelli.
8. Appendix: Functionals and functions of many variables. Calculus of variation.
9. Appendix: Level sets – basic ideas. Split Bregman (basics).

2 Models with Spatial Context

Previous lectures assumed that we could classify, or label, pixels independently. This simplifies the computation. But it is a bad approximation. For example, if a pixel is grass then there is a high probability that nearby pixels are also grass.

The next few lectures will introduce methods for taking this spatial context into account. We will also describe how we can learn spatial context from labeled training data. These methods will require far more computation both for doing inference (interpreting a single image) and for learning the models.

3 The Mumford and Shah model

Mumford and Shah formulated image segmentation of a domain D as the minimization of a functional $E[J, B]$. The input I is an image, The output

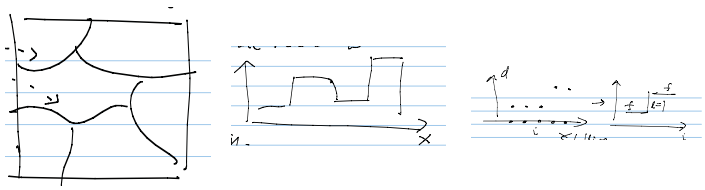

Figure 1: Mumford-Shah tries to decompose the image into subdomains within which the image is smooth (left panel). This assumes a piecewise smooth image (center panel). The energy puts an edge at places where the gradient is too large (right panel).

$(\hat{J}, \hat{B}) = \arg\min E[J, B]$ is a smoothed image $\hat{J}$ and the position $\hat{B}$ of the boundaries that separates D into subdomains $D = \bigcup D_i$, with $D_i \bigcap D_j = 0$ $i \neq j$ (and $B = \bigcup \partial D_i$) (i.e. b specifies the positions of a one-dimensional set of points). $E[J, B]$ is called a functional because the argument is a function $J(\vec{x})$. (See appendix on functionals and calculus of variations):

$$E[J, B] = C \int d\vec{x}(I(\vec{x}) - J(\vec{x}))^2 + A \int_{D/B} \vec{\nabla} J(\vec{x}) \cdot \vec{\nabla} J(\vec{x}) d\vec{x} + B \int_B ds. \quad (1)$$

The first term ensures that the the smoothed image J is similar to the input I. The second term ensures that J has small gradient $|\vec{\nabla} J|$ (i.e. J is smooth) except across boundaries B. The third term penalizes the length of the boundaries ($\int_B ds$ is a one-dimensional integral). Intuitively, it tries to smooth the image I except at places where the image gradient $|\vec{\nabla} I|$ is too high – where it cost less energy to insert a boundary/edge.

This model exploits both edge and regional cues: (i) *edge cues:* it tries to insert boundaries at places where the gradient of the image $I(\vec{x})$ is large, and (ii) *regional cues:* it tries to group pixels which have similar intensities into regions.

The Mumford and Shah model is closely related to the weak membrane models by Geman+Geman and Blake+Zisserman which are formulated on the image lattice (see later section!!).

There are several problems with the energy functional $E[J, B]$: (I) It is difficult to find an algorithm which finds its minimum $(\hat{J}, \hat{B}) = \arg\min E[J, B]$, or even a "good" local minimum – this is because $E[J, B]$ is a function of a continuous variable (function) J and a discrete function B. (II) It only uses a very local form of smoothness based on first derivatives (first derivatives are sufficient to model the smoothness of a soap bubble but second order derivatives are needed to model thin plate splines – discuss why higher order smoothness is needed – show figure). (III) It cannot deal with images which contain texture regions. (But the last two are limitation of all models in this lecture). (IV) It is not obvious that $E[J, B]$ has a well defined minimum (an important issue for mathematicians).

We did not discuss this model any more in the course lectures. It was very influential but is now studied mainly for its mathematical interest (weak smoothness is too simple a model to capture the complexity of real images). We proceeded to the Rudin-Osher-Fatemi model is simpler but which has efficient inference algorithms and was, until recently, at the state of the art for image de-noising).

4 Rudin, Osher, Fatemi: the TV norm model

We introduce another energy functional $E[J;I]$ which is simpler than Mumford-Shah but shares many of its intuitions – its output $\hat{J}$ is a smoothed version of the input I except at boundaries/edges where the image gradient is large. This model has many practical advantages – there are algorithms that can find its global minimum very rapidly (real time) and it used to be state of the art for image denoising.

$$E[J;I] = \int_D |\vec{\nabla} J| d\vec{x} + \frac{\lambda}{2} \int_D (J(\vec{x}) - I(\vec{x}))^2 d\vec{x} \tag{2}$$

This has an L1 norm on the gradient and no explicit edge variable (but see below). Because it is the sum of two *convex* terms – quadratic and linear – it is oonvex (see next section). Since it is also bounded below (clearly $E[J;I] \geq 0$) it has only a global minimum and hence steepest descent methods are guaranteed to perform inference and get to the global minimum.

$$\begin{aligned} \frac{dJ}{dt} &= -\frac{\partial E}{\partial J}, \\ &= \vec{\nabla} \cdot \{ \frac{\vec{\nabla} J}{|\vec{\nabla} J|} \} - \lambda (J - I). \end{aligned} \tag{3}$$

The first term tries to make J have small gradients and the second term tries to keep J close to the input I. Here $\frac{\partial E}{\partial J}$ is the functional derivative (see appendix on calculus of variations!!).

The Rudin-Osher-Fatemi model has no explicit edge variable (unlike Mumford-Shah). But it can be reformulated be introducing a new variable z and an energy function:

$$E[J,z] = \frac{1}{2} \int_D \{ z |\vec{\nabla} J|^2 + z^{-1} \} d\vec{x} + \frac{\lambda}{2} \int_D (J - I)^2 d\vec{x}. \tag{4}$$

We can solve for $z = 1/|\vec{\nabla} J|$ (differentiate $E[J,z]$ with respect to z) and substitute back to obtain the Rudin-Osher-Fatemi model. We can interpret z as a measure of edgeness – $z = 0$ indicates an edge.

Alternative minimization can be used to minimize $E[J,z]$. Minimizing $E[J,z]$ with respect to z yields $z^{t+1} = 1/|\vec{\nabla} J^t|$. Minimizing $E[J,z]$ with respect to z requires solving:

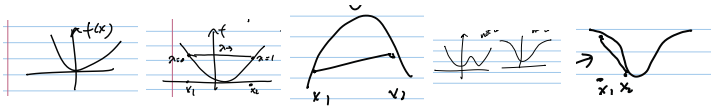

Figure 2: Convex functions (far left panels). Concave function (center panel). Non-convex function (right panel). Unique minimum for non-convex function (far right panel).

$$-2\vec{\nabla}\{z^t\vec{\nabla}J^{t+1}\} + \lambda(J^{t+1} - I) = 0, \tag{5}$$

which has a unique solution (since $E[J, z]$ is a convex function of J if z is fixed).

This alternative minimization reduces to the well-known *lagged-diffusion* model:

$$-\vec{\nabla} \cdot \{|\vec{\nabla}J^t|^{-1}\vec{\nabla}J^{t+1}\} + \lambda(J^{t+1} - I) = 0. \tag{6}$$

NOTE: should mention level sets. State other nice properties of the TV norm.

5 Convexity and Steepest Descent

An energy functional (or function) $E[J; I]$ is convex if for all $0 \leq \alpha \leq 1$ and any J_1, J_2 we have $\alpha E[J_1, I] + (1 - \alpha)E[J_2, I] \geq E[\alpha_1 J_1 + (1 - \alpha_1)J_2]$. This is equivalent to the condition that the Hessian of $E[J; I]$ – the second order derivatives $\frac{\delta^2 E}{\delta J^2}$ – is positive semi-definite.

Steepest descent updates J in the direction of the gradient $-\frac{\partial E}{\partial J}$ (i.e. downhill in $E[J; I]$. This is guaranteed to reduce the energy:

$$\begin{aligned} \frac{dJ}{dt} &= -\frac{\partial E}{\partial J}, \\ \text{Implying} \frac{dE}{dt} &= -\frac{\partial E}{\partial J}\frac{dJ}{dt} = -\frac{\partial E}{\partial J}\frac{\partial E}{\partial J} \leq 0. \end{aligned} \tag{7}$$

If $E[J]$ is convex and bounded below (e.g., $E[J] \geq 0$) then $E[J]$ has a unique minimum (which is global) and steepest descent is guaranteed to find it. There are non-convex functions which only have a singe (global) minimum. But convexity is the only criterion that can be easily checked to guarantee that an energy function has a unique minimum.

Concave functions are the opposite of convex functions. Hence $-E[J]$ is concave if $E[J]$ is convex and vice versa.

Surprisingly a very large number of functions can be decomposed into the sum of a convex and a concave part. This seems counter-intuitive. But it can be shown fairly easily (e.g., see Yuille and Rangarajan).

6 Variational Bounding and CCCP

Note: some repetition at the start of this section.

Steepest descent methods need to be discretized in time to be implemented by computers. We must convert the update equation $\frac{d\vec{x}}{dt} = -\frac{\partial f}{\partial \vec{x}} = -\vec{\nabla} f(\vec{x})$ into a discrete update rule:

$$\vec{x}_{t+1} = \vec{x}_t - \Delta \vec{\nabla} f(\vec{x}(t)), \tag{8}$$

There is a trade-off in the choice of the parameter Δ. To approximate the continuous time update equation requires Δ to be infinitesimally small. But if Δ is too small then the algorithm converges slowly. But, conversely, if Δ is too big then the discrete update is a bad approximation to the continuous time update and may not converge (decrease in energy is not guaranteed – the solution can oscillate or even go chaotic). There are standard methods for dealing with this – see also Newton-Raphson (Refs) – it is usually good to determine Δ adaptively so that it is small when the gradient $|\vec{\nabla} f(\vec{x}(t))|$ is large and big when the gradient is small.

Discrete iterative algorithms are an alternative. They are guaranteed to converge (at least to local minima) and do not require choosing a step size. They are less well-known in general, but occur frequently in vision and machine learning, so we describe them in more detail here. Fortunately, and surprisingly, there are some general principles that underlies almost all discrete iterative methods (despite many of them being developed independently). (Refer to methods – majorization, EM, Legendre transform, etc.).

Consider minimizing $E(\vec{x})$. Variational bounding proceeds by obtaining a sequence of bounding functions $E_B(\vec{x}, \vec{x}_n)$ where $\vec{x}_n$ is the current state, see figure (3). The bounding functions must obey:

$$\begin{aligned} E_B(\vec{x}, \vec{x}_n) &\geq E(\vec{x}), \forall \vec{x}, \vec{x}_n \\ E_B(\vec{x}_n, \vec{x}_n) &= E(\vec{x}_n). \end{aligned} \tag{9}$$

Then the algorithm $\vec{x}_{n+1} = \arg\min_{\vec{x}} E_B(\vec{x}_n)$ is guaranteed to converge to a minimum of $E(\vec{x})$. The algorithm can make large moves from $\vec{x}_n$ to $\vec{x}_{n+1}$.

A special case of this approach is called CCCP (it can be shown that many discrete iterative algorithms – including many instances of variational bounding – can be derived from CCCP – Yuille and Rangarajan). Decompose the function $E(\vec{x})$ into a concave $E_c(\vec{x})$ and a convex part $E_v(\vec{x})$ so that: $E(\vec{x}) = E_c(\vec{x}) + E_v(\vec{x})$. Then define the update rule:

$$\vec{\nabla} E_v(\vec{x}_{n+1}) = -\vec{\nabla} E_c(\vec{x}_n). \tag{10}$$

By properties of convex and concave functions, this CCCP update rule is guaranteed to decrease the energy at each time-step.

It is easy to check (FIGURE) that the CCCP is a special case of majorization/variational bounding. Because it corresponds to the bound $E_B(\vec{x}, \vec{x}_n) = E_v(\vec{x}) + E_c(\vec{x}_n) + (\vec{x} - \vec{x}_n) \cdot \vec{\nabla} E_c(\vec{x}_n)$.

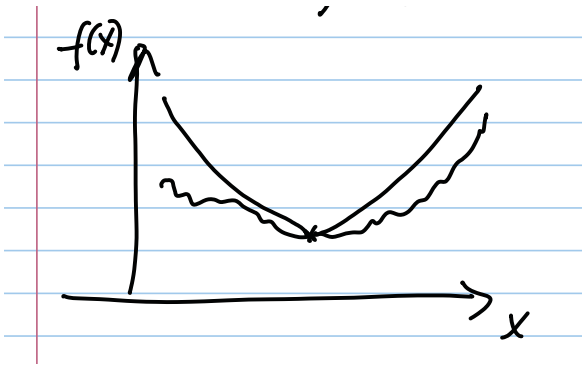

Figure 3: Put a variational bound on the function you want to minimize. Then minimize this bound. Then replace with a new bound and repeat.

It can be shown that several well-known algorithms used in this course – EM, GIS, Sinkhorn – can be obtained as special cases of CCCP and hence of majorization/variational bounding (Yuille and Rangarajan – though the proofs may require changes of variables).

Here is a simple example $E(x) = x \log x + (1-x)\log(1-x) - (1/2)x^2$ for $0 \leq x \leq 1$. We can decompose $E(x)$ into a convex function $E_v(x) = x \log x + (1-x)\log(1-x)$ and a concave function $E_c(x) = -(1/2)x^2$. This decomposition gives a discrete update rule:

$$x_{n+1} = \frac{1}{1 + \exp\{-x_n\}}. \tag{11}$$

6.1 Relations of Discrete Iterative to Steepest Descent

Note: this is optional.

Why should you care? It shows that methods for discretezing steepest descent can be derived directly from discrete iterative algorithms.

First we show how to derive the discretizing parameter Δ for steepest descent from the perspective of discrete iterative algorithms (in particular CCCP). The goal is to minimize $f(x)$. Re-express this as $f(x) = (1/2)Kx^2 + \{f(x) - 1/2Kx^2\}$ where K is chosen so $f(x) - 1/2Kx^2$ is concave (use Hessian argument – choose K so that the Hessian is negative semi-definite). Then CCCP gives the update equation $x^{t+1} = x^t - (1/K)\frac{\partial f}{\partial x}$. Hence the step $\Delta = 1/K$ can be derived from requiring that $f(x) - 1/2Kx^2$ is concave.

Some popular methods for stabilizing gradient/steepest descent can be explained as an example of this discrete iterative framework. For example, Eyre studies gradient flow $d\vec{x}/dt = -\vec{\nabla}E(\vec{x})$ and proposes to decompose $E(\vec{x}) = E_1(\vec{x}) - E_2(\vec{x})$ where $E_1(.)$ and $E_2(.)$ are convex functions (i.e., $-E_2(\vec{x})$ is a concave function – so this is like a CCCP decomposition). His update rule is:

$$\vec{x}_{t+1} - \vec{x}_t = \Delta\{\vec{\nabla} E_2(\vec{x}_t) - \vec{\nabla} E_1(\vec{x}_{t+1})\}. \tag{12}$$

This is just CCCP with the decomposition $E_v(\vec{x}) = E_1(\vec{x}) + (1/2\Delta)\vec{x} \cdot \vec{x}$ and $E_c(\vec{x}) = -E_2(\vec{x}) - (1/2\Delta)\vec{x} \cdot \vec{x}$. Observe that Eyre's method looks like a small modification of the basic gradient/steepest descent $\vec{x}_{t+1} - \vec{x}_t = -\Delta\{\vec{\nabla} E(\vec{x}_t)$.

7 L1 norm versus L2 – sparsity and robustness

The Rudin-Osher-Fatemi model uses an L1 penalty for the gradient $|\vec{J}|$. One advantage of this penalty is that it encourages sparsity – it will bias the solution to precisely 0. This relates to *sparsity* (which will re-occur throughout the course) and is illustrated by the following example (here we replace $|\vec{J}|$ by J to simplify the argument).

$$f(\omega; I) = (\omega - I)^2 + \lambda|\omega| \tag{13}$$

What is $\hat{\omega}(I) = \arg\min_\omega f(\omega; I)$?

$$f_+(\omega; I) = (\omega - I)^2 + \lambda\omega. \quad \text{For } \omega \geq 0 \tag{14}$$
$$f_-(\omega; I) = (\omega - I)^2 - \lambda\omega. \quad \text{For } \omega \leq 0 \tag{15}$$

$$\frac{df_+}{d\omega} = 2(\omega - I) + \lambda = 0 \Rightarrow \hat{\omega}(I) = \frac{2I - \lambda}{2}, \forall\omega \geq 0 \tag{16}$$
$$\frac{df_-}{d\omega} = 2(\omega - I) - \lambda = 0 \Rightarrow \hat{\omega}(I) = \frac{2I + \lambda}{2}, \forall\omega \leq 0 \tag{17}$$

Hence, we have

$$\hat{w}(I) = \frac{2I - \lambda}{2}, \quad I \geq \tfrac{\lambda}{2} \tag{18}$$
$$\hat{w}(I) = 0, \quad |I| \leq \tfrac{\lambda}{2} \tag{19}$$
$$\hat{w}(I) = \frac{2I + \lambda}{2}, \quad I \leq \tfrac{-\lambda}{2} \tag{20}$$

As we show in Figure 7, the use of the $L1$ norm $|\omega|$ biases the solution to $\hat{\omega}(I) = 0$ for small I.

By contrast, $f_2(\omega; I) = (w - I)^2 + \lambda\omega^2$ has minimum at $\omega - I + \lambda\omega = 0$. Therefore,

$$\hat{\omega}(I) = \frac{I}{1 + \lambda} \tag{21}$$

which always smooths I, but doesn't force it to 0.

Note: $L1$ norm corresponds to an exponential distribution and $L2$ norm to a Gaussian distribution. This will be described in the next lecture. Gaussian distribution are non-robust and very sensitive to outliers.

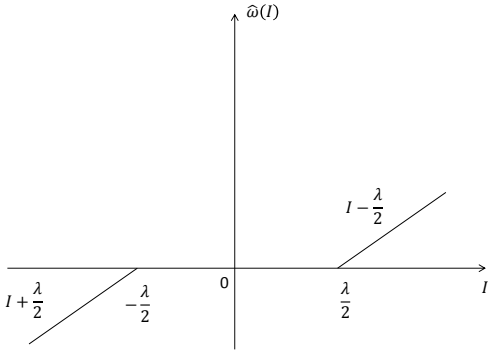

Figure 4: L1 norm is better

7.1 Appendix I: Mumford and Shah – Ambrosio-Tortorelli

Advanced topic: not required for the course.

Why should you care? It gives examples of optimization methods which can be applied to a range of vision problems (and other problems). (I) It illustrates a continuation method – want to minimize a function $E[f]$, define a parameterized family of functions $E[f,\epsilon]$ so that $\lim_{\epsilon\mapsto 0} E[f,\epsilon] = E[f]$ and where $E[f,\epsilon]$ becomes smoother (has less minima) as ϵ increases (like annealing), minimize $E[f,\epsilon]$ for large ϵ, use this as initial conditions to minimize $E[f,\epsilon]$ for smaller ϵ (continuation methods are a heuristic – usually impossible to prove convergence to global minimum – but often work well). (II) It is an example of a method for converting a discrete valued variable (e.g., B) to a continuous-valued one, which enables the use of gradient descent techniques (because we can compute gradients of continuous variables). (III) It is an example of alternative minimization where we minimize a function $E(f,z)$ by fixing z, minimizing $E(f,z)$ with respect to f, the fixing f and minimizing with respect to z, and repeating.

Ambrosio-Tortorelli showed that Mumford-Shah functional $E[J,B]$ can be obtained as the limit of a family of energy functionals $E[J,z:\epsilon]$ where the boundary B is replaced by continuous function z whose magnitude indicates the presence of a boundary. Their result (see also Shen) applies to functionals:

$$E[J,z;\epsilon] = C\int d\vec{x}(I(\vec{x})-J(\vec{x}))^2 + A\int d\vec{x} z(\vec{x})|\vec{\nabla}J(\vec{x})|^2 + B\int d\vec{x}\{\epsilon|\vec{\nabla}J(\vec{x})|^2 + \epsilon^{-1}\phi^2(z(\vec{x}))\}, \tag{22}$$

where $\epsilon > 0$ is a (small) parameter and $\phi(z)$ is a potential function. Two forms of $\phi(z)$ are $\phi_1(z) = (1-z)/2$ and $\phi_2(z) = 3z(1-z)$ (where $z \in [0,1]$). For $\phi_1(.)$

the edge set B is associated with the set of points such that $\phi_1(z) \approx 0$, for $\phi_2(.)$ the edge set is associated with points such that $\phi_2(z) \approx 1/2$. As $\epsilon \mapsto 0$, the last two terms of the energy function converge to the edge set integral $\int_B ds$ (this is not-trivial).

This energy $E[J, z; \epsilon]$ can be minimized (local minima) by gradient descent methods – calculate the gradient E with respect to J and z and set $J^{t+1} = J^t - \delta \frac{\partial E}{\partial J}$ and $z^{t+1} = z^t - \delta \frac{\partial E}{\partial z}$ (see gradient descent section). Alternatively, we can minimize $E[J, z; \epsilon]$ with respect to J and z in alteration – alternative minimization. (Note: this is particularly easy since $E[J, z; \epsilon]$ is a convex function of J (for fixed z) and a convex function of z (for each J).

8 Appendix II: Functional Analysis and Calculus of Variations

Background material: optional.

Why should you care? This lets you differential functionals (where the arguments are functions). It is analogous to functions of multiple variables – where the number of variables becomes infinite. Full understanding of calculus of variations requires a textbook (e.g., Google Peter Olver calculus of variations).

$E[J]$ is a functional if its argument $\{J(x) : x \in D\}$ is a function (by contrast, the the argument of a function is a scalar or a vector). The calculus of variations describes how to take *functional derivatives* of functionals. Here we introduce the basic concepts and discuss the relations to function of multiple variables.

We start by defining a product $< J, W >= \int J(x)W(x)dx$ between two functions. This is analogous to the dot product between two vectors – e.g. $\vec{J} \cdot \vec{W} = \sum_i J_i W_i$. The product gives an L^2 norm for the space $|J|^2 =< J, J >$. The integrals are defined over the domain D.

The functional derivative $\frac{\delta E}{\delta J}$ of $E[J]$ with respect to J is defined by the relation:

$$< \frac{\delta E}{\delta J}, W >= \frac{d}{dt} E[J + tW], \text{ for any function } W. \tag{23}$$

Here the right hand side is the derivative of a scalar t. It is the derivative of E in the direction W (e.g., like taking the derivative of a multivariate function $f(\vec{x})$ in direction $\vec{n}$ – $\vec{n} \cdot \nabla f = \frac{d}{dt} f(\vec{x} + t\vec{n})$).

Now suppose that $E[J] = \int L(J, J']dx$ where $J'(x) = \frac{d}{dx} J(x)$ (e.g., $L(J, J') = J(x)^2 + \frac{dJ}{dx}^2$)). Then it follows that

$$< \frac{\delta E}{\delta J}, W >= \int \{W \frac{\partial L}{\partial J} + W' \frac{\partial L}{\partial J'}. \tag{24}$$

(e.g., $\frac{\partial L}{\partial J} = 2L(x)$ and $\frac{\partial L}{\partial J'} = 2J'(x)$).

Next we use integration by parts to convert the term:

$$\int W' \frac{\partial L}{\partial J'} dx = -\int W \frac{d}{dx} \frac{\partial L}{\partial J'} dx + \int \frac{d}{dx} \{W \frac{\partial L}{\partial J'}\} dx \tag{25}$$

We require that $W(x) = 0$ for $x \in \partial D$ – the boundary of D. Then it follows that:

$$< \frac{\delta E}{\delta J}, W >= \int W\{\frac{\partial L}{\partial J} - \frac{d}{dx}\frac{\partial L}{\partial J'}\}dx \tag{26}$$

This condition must hold for all functions $W(x)$ (which vanish on ∂D). Hence we have:

$$\frac{\delta E}{\delta J} = \frac{\partial L}{\partial J} - \frac{d}{dx}\frac{\partial L}{\partial J'} \tag{27}$$

These are the Euler-Lagrange equations. (E.g., $2J(x) - 2J''(x)$).

This can be extended to other functionals – e.g., $L(W, W', W'')$ – by repeating this procedure performing integration by parts as often as needed.

So there are only two things to know – the definition in equation (!!) and the use of integration by parts.

It is interesting to compare this to differentiation of functions of multiple variables. Suppose we discrete the functional $E[J] = \int J(x)^2 + \frac{dJ}{dx}^2$ to obtain:

$$E_D[J] = \sum_i J_i^2 + \sum_i (J_{i+1} - J_i)^2. \tag{28}$$

We compute:

$$\frac{\partial E_D}{\partial J_j} = 2J_i + 2(2J_i - J_{i+1} - J_{i-1}). \tag{29}$$

which is the same as the discretized version of $2J(x) - 2J''(x)$.

Appendix III: Legendre Transforms

Note: another optional section.

The Legendre transform can be used to reformulate optimization problems in terms by introducing auxiliary variables. Let $E(\vec{x})$ be a convex function, define its Legendre transform to be $E^*(\vec{y}) = \min_{\vec{x}}\{E(\vec{x}) + \vec{x} \cdot \vec{y}\}$. It can be verified that $E^*(\vec{y})$ is concave. We recover $E(\vec{x})$ by the inverse transform: $E(\vec{x}) = \max_{\vec{y}}\{E^*(\vec{y}) - \vec{y} \cdot \vec{x}\}$.

Now suppose an energy function is expressed in CCCP form as $E(\vec{x}) = E_v(\vec{x}) + E_c(\vec{x})$ where $E_v(.)$ is convex and $E_c(.)$ concave. Minimizing $E(\vec{x})$ wrt $\vec{x}$ is equivalent to minimizing the function $E(\vec{x}, \vec{y}) = E_v(\vec{x}) + \vec{x} \cdot \vec{y} + E_c^{-1*}(\vec{y})$ wrt $\vec{x}, \vec{y}$, where E_c^{-1*} is the inverse Legendre transform of $E_c(.)$. Minimizing $E(\vec{x}, \vec{y})$ by coordinate descent (i.e. wrt $\vec{x}$ and $\vec{y}$ alternatively) is equivalent to performing CCCP on $E(.)$.