Neural Data Analysis

Tutorial 3: Gradient Descent

Solving the Least Squares Equation

$$-\ln(L) \propto \frac{1}{2}\sum_{i=1}^{n}(\vec{h}\cdot\vec{s}_i - r_i)^2$$

The least squares equation is a parabola. We want to find its minimum, so we differentiate with respect to h, set equal to 0, and solve for h.

$$\frac{\partial}{\partial h}\left(\frac{1}{2}\sum_{i=1}^{n}(\vec{s}_i\cdot\vec{h} - r_i)^2\right) = \sum_{i=1}^{n}\vec{s}_i^{\,T}(\vec{s}_i\vec{h} - r_n) = \mathbf{S}^{\mathrm{T}}\mathbf{S}\vec{h} - \mathbf{S}^{\mathrm{T}}\mathbf{r}$$

$$\vec{h} = \left(\mathbf{S}^{\mathrm{T}}\mathbf{S}\right)^{-1}\mathbf{S}^{\mathrm{T}}\mathbf{r}$$

Advantages of Gradient Descent

- Will find local minimum of any differentiable function
- For least squares (LSQ) cost function and linear models, gradient descend will find global minimum
- No need to invert large covariance matrix
- Gradient descent with zero – initialization and early stopping provides regularized solution
- Variants can be used for other priors (other starting points and other descent methods)

Gradient Descent

$$\frac{\partial}{\partial h}\left(\frac{1}{2}\sum_{i=1}^{n}(\vec{s}_i \cdot \vec{h} - r_i)^2\right) = \sum_{i=1}^{n}\vec{s}_i^{\,T}(\vec{s}_i\vec{h} - r_n) = \mathbf{S}^{\mathrm{T}}\mathbf{S}\vec{h} - \mathbf{S}^{\mathrm{T}}\mathbf{r}$$

The gradient at a particular value of model parameters h^m is:

$$\vec{g}^{\,m} = \mathbf{S}^{\mathrm{T}}\mathbf{S}\vec{h}^{\,m} - \mathbf{S}^{\mathrm{T}}\mathbf{r}$$

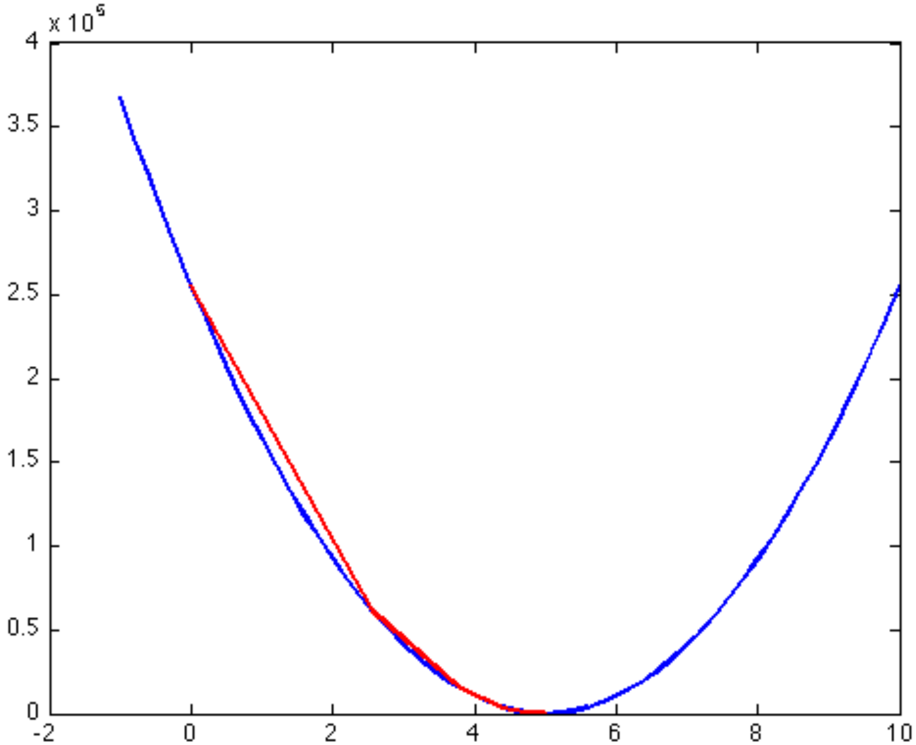
$$\vec{g}^{\,m} = \mathbf{S}^{\mathrm{T}}(\mathbf{S}\vec{h}^{\,m} - \mathbf{r})$$

Taking a step in the right direction:

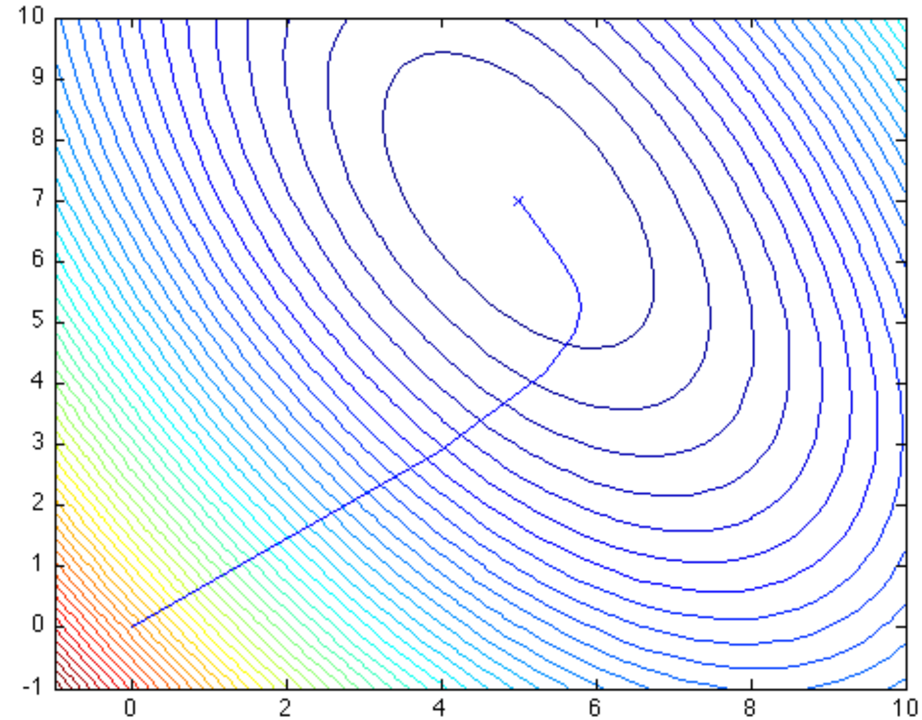
$$\vec{h}^{\,m+1} = \vec{h}^{\,m} - \Delta\vec{g}^{\,m}$$

Gradient Descent in 1D and 2D

1D Error Surface



2D Error Surface



Gradient Descent Pseudocode

Initialize step size

Initialize $h^1 = 0$

for iter = 1:maximum iterations

$$\vec{g}^{m} = \mathbf{S}^{\mathrm{T}}(\mathbf{S}\vec{h}^{m} - \mathbf{r})$$

$$\vec{g}^{m} = \mathbf{S}^{\mathrm{T}}(\mathbf{S}\vec{h}^{m} - \mathbf{r})$$

$$error^{m+1} = \frac{1}{2}\sum_{i=1}^{n}(\vec{s}_i \cdot \vec{h}^{m+1} - r_i)^2$$

stop if error ~ 0

end

Early Stopping Regularization

- Gradient descent makes the largest changes along the dimensions that account for the most variance first
- When the signal is noisy, the LSQ solution may overfit to noise
- Stopping before reaching the LSQ solution will give a solution that accounts for most of the variance without over fitting
- Early stopping corresponds to a Gaussian prior

$$\underbrace{p(h \mid S, r)}_{\text{Posterior}} \propto \underbrace{p(r \mid S, h)}_{\text{Likelihood}} \underbrace{p(h)}_{\text{Prior}}$$

How do you know when to stop?

- Solution 1.
  - Estimate the SNR of your data.
  - Stop when the error is on the order of the noise power (error < Total power/(SNR+1))
- Solution 2. Cross-validate.
  - Hold out part of your training data from the calculation of the gradient
  - On each iteration check the error on this held out set
  - When the error no longer improves on the held out set, stop descent

Coordinate Descent

Calculate Gradient

$$\vec{g}^{m} = \mathbf{S}^{\mathrm{T}}(\mathbf{S}\vec{h}^{m} - \mathbf{r})$$

Update dimension
with largest gradient

$$h_k^{m+1} = h_k^{m} - \Delta \max_k(\vec{g}^{m})$$

Advantages of Coordinate Descent

- Coordinate descent will converge on the LSQ solution, but it’s main benefit is when combined with early stopping
- The early stopped coordinate descent answer will be sparse, aiding interpretation
- Small weights due to noise may be zeroed out, improving predictions
- The early stopped coordinate descent answer is nearly equivalent to a Laplacian (L1) prior

Coordinate Descent Pseudocode

Initialize step size
Initialize $h^1 = 0$
for iter = 1:maximum iterations

$$g^m = \mathbf{S}^{\mathrm{T}}(\mathbf{S}h^m - \mathbf{r})$$

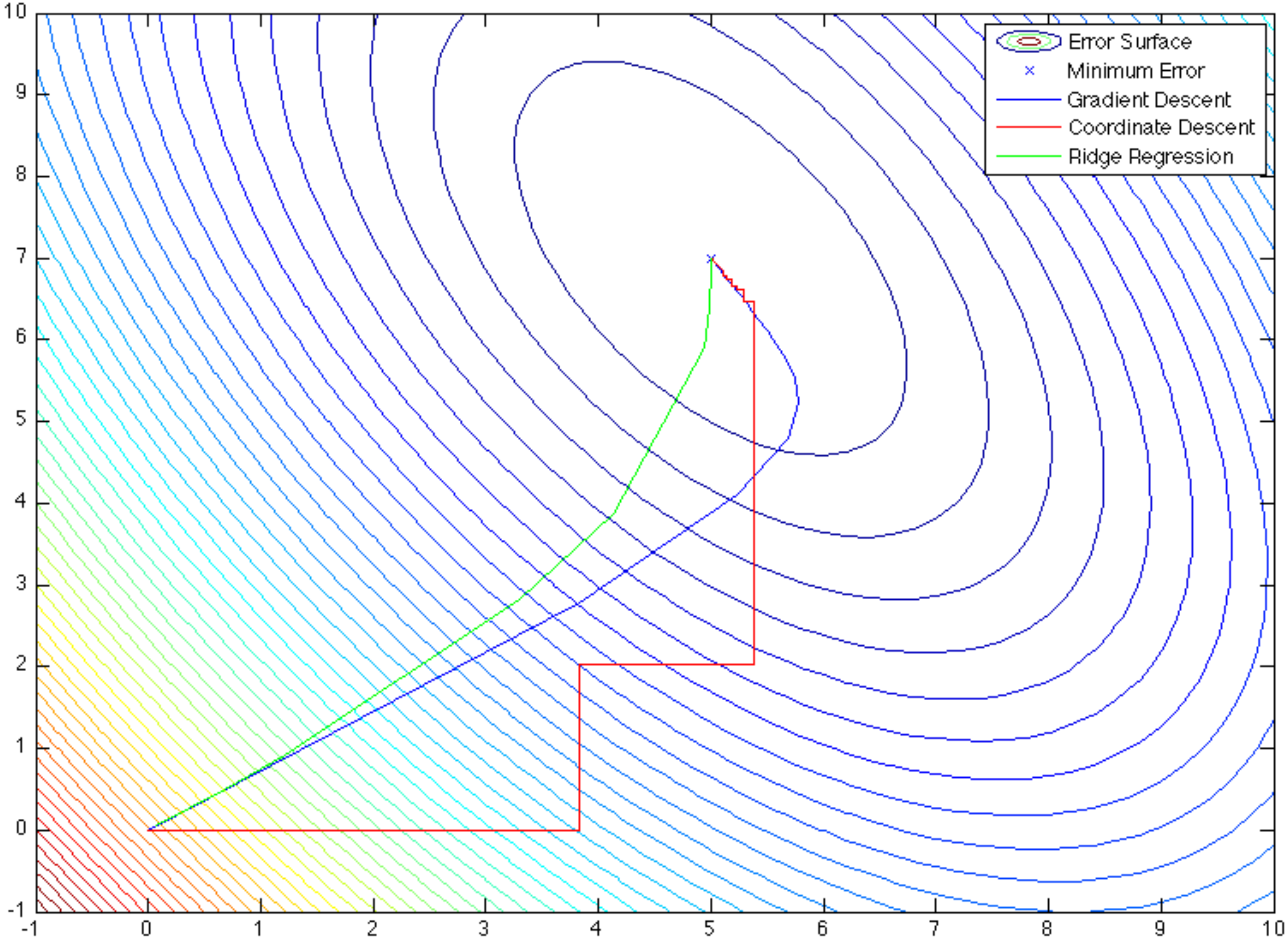
$$h_k^{m+1} = h_k^m - \Delta \max_k (g^m)$$

calculate error on held out set

stop if error increases
end

Regularization paths

- Gradient descent, coordinate descent, ridge regression and other regularized regression algorithms can be seen as creating paths through parameter space
- Points on these paths correspond to different values of regularization parameters/weights on the prior
- The different paths correspond to different types of priors
- The best path/prior will depend on the problem and data


Error Surface
Minimum Error
Gradient Descent
Coordinate Descent
Ridge Regression
10
9
8
7
6
5
4
3
2
1
0
-1
-1
0
1
2
3
4
5
6
7
8
9
10

L2 Regularization = Ridge Regression = Gaussian Isotropic Prior

~ Gradient Descent with zero initialization and early stopping

$$O = \frac{1}{2}\sum_{i=1}^{n}(r_i - \vec{h}\cdot\vec{s}_i)^2 + \frac{\lambda}{2}\left\|\vec{h}\right\|^2$$

$$p(\vec{h}) = \mathcal{N}(0, \lambda^{-1}\mathbf{I})$$

$$p(\vec{h}) = \frac{\lambda^{1/2}}{\sqrt{2\pi}} e^{-\frac{\lambda}{2}h^2}$$

$$\vec{h} = (\mathbf{S}^{\mathrm{T}}\mathbf{S} + \lambda\mathbf{I})^{-1}\mathbf{S}^{\mathrm{T}}\mathbf{r}$$

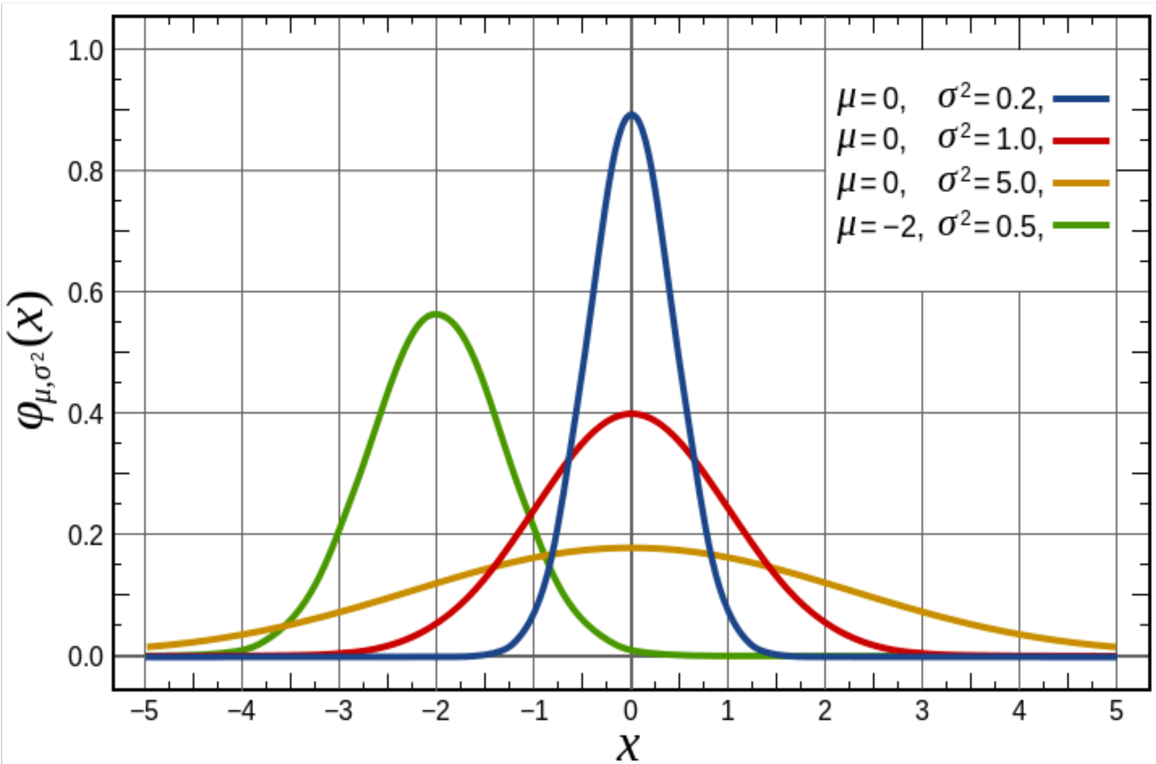
L1 Regularization = Lasso Regression = Laplace Prior

~ Coordinate Descent with zero initialization and early stopping

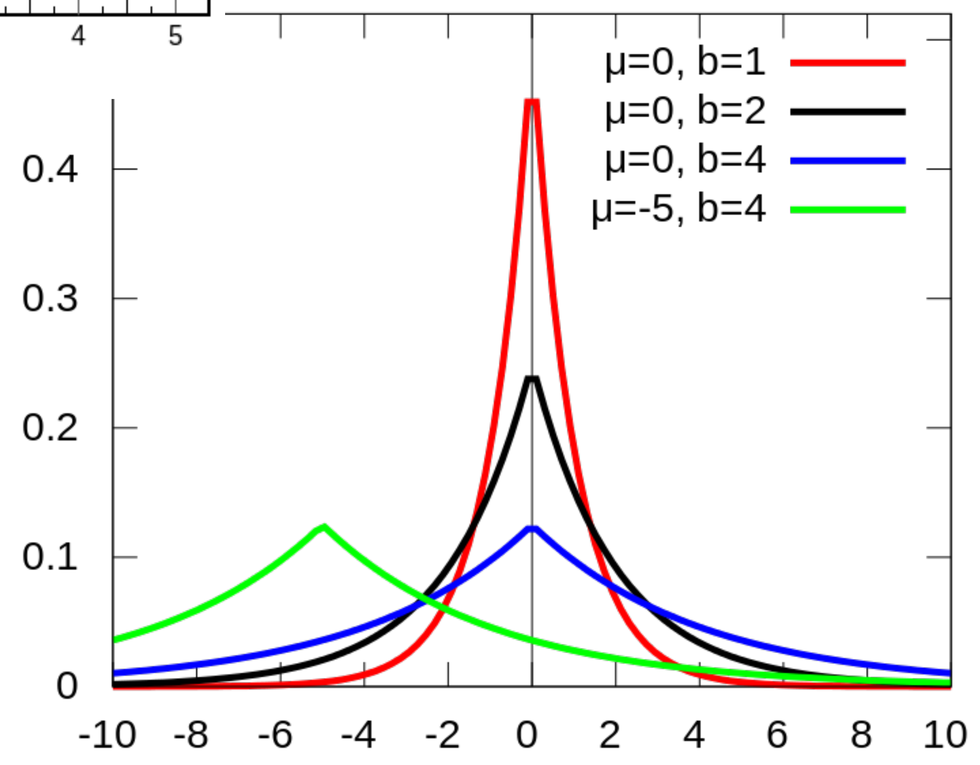
$$O = \frac{1}{2}\sum_{i=1}^{n}(r_i - \vec{h}\cdot\vec{s}_i)^2 + \lambda\left\|\vec{h}\right\|$$

$$\left\|\vec{h}\right\| = \sum_{i=1}^{k}\left|h_i\right|$$

$$p(\vec{h}) = \prod_{i=1}^{k}\frac{\lambda}{2\sigma}e^{-\frac{\lambda}{\sigma}\left|h_i\right|}$$



Normal Distributions



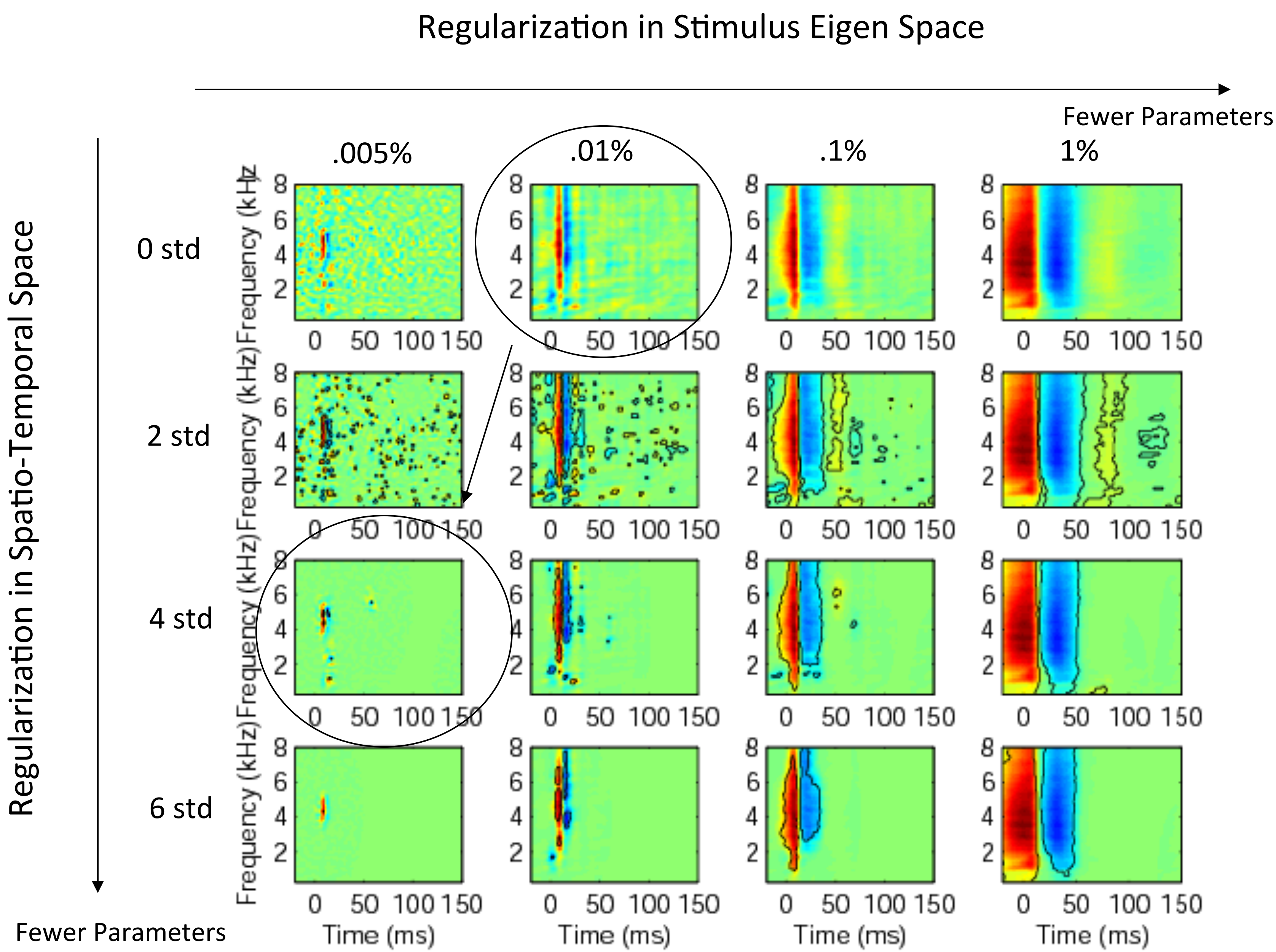
Laplace Distributions

Elastic Net combinesL1 and L2 the two with two hyper paramenters

$$O = \frac{1}{2n}\sum_{i=1}^{n}(r_i - \vec{h}\cdot\vec{s}_i)^2 + \lambda\left(\frac{1-\alpha}{2}\left\|\vec{h}\right\|^2 + \alpha\left\|\vec{h}\right\|\right)$$

λ is the hyperparameter for how much shrinkage
α determines whether shrinkage is Lasso (α=1) or Ridge (α=0)

Regularization with Two Hyper Parameters



Proof that early stopping Gradient descent corresponds to a Gaussian Prior

$$\mathbf{h}^{n+1} = \mathbf{h}^n - \varepsilon \mathbf{S}^T\left(\mathbf{S}\mathbf{h}^n - \mathbf{r}\right)$$

$$\frac{\mathbf{h}^{n+1} - \mathbf{h}^n}{\varepsilon} = -\mathbf{S}^T\mathbf{S}\mathbf{h}^n + \mathbf{S}^T\mathbf{r}$$

$$\frac{\mathbf{h}^{n+1} - \mathbf{h}^n}{\varepsilon} = -\mathbf{U}\Lambda\mathbf{U}^T\mathbf{h}^n + \mathbf{S}^T\mathbf{r}$$

$$\frac{\mathbf{U}^T\mathbf{h}^{n+1} - \mathbf{U}^T\mathbf{h}^n}{\varepsilon} = -\Lambda\mathbf{U}^T\mathbf{h}^n + \mathbf{U}^T\mathbf{S}^T\mathbf{r}$$

$$\mathbf{U}^T\mathbf{h}^n = \mathbf{f}^n \qquad \mathbf{U}^T\mathbf{S}^T\mathbf{r} = \mathbf{q}$$

$$\frac{\mathbf{f}^{n+1} - \mathbf{f}^n}{\varepsilon} = -\Lambda\mathbf{f}^n + \mathbf{q}$$

$$\frac{f_i^{n+1} - f_i^n}{\varepsilon} = -\lambda_i f_i^n + q_i$$

$$\frac{df_i(n)}{dn} = -\lambda_i f_i(n) + q_i$$

$$f_i(n) = Ce^{-\lambda_i n} + \frac{q_i}{\lambda_i}$$

$$f_i(0) = 0$$

$$C = -\frac{q_i}{\lambda_i}$$

$$f_i(n) = \left(1 - e^{-\lambda_i n}\right)\frac{q_i}{\lambda_i}$$

$$f_i(n) = \left(\frac{\lambda_i}{\left(1 - e^{-\lambda_i n}\right)}\right)^{-1} q_i$$

$$f_i(n) = \left(\lambda_i + \frac{\lambda_i}{\left(e^{\lambda_i n} - 1\right)}\right)^{-1} q_i$$

$$\mathbf{f}(n) = \left(\Lambda + \left(e^{n\Lambda} - \mathbf{I}\right)^{-1}\Lambda\right)^{-1}\mathbf{q}$$

$$\mathbf{U}^T\mathbf{h}(n) = \left(\Lambda + \left(e^{n\Lambda} - \mathbf{I}\right)^{-1}\Lambda\right)^{-1}\mathbf{U}^T\mathbf{S}^T\mathbf{r}$$

$$\mathbf{h}(n) = \left(\mathbf{U}^T\Lambda\mathbf{U} + \mathbf{U}^T\left(e^{n\Lambda} - \mathbf{I}\right)^{-1}\Lambda\mathbf{U}\right)^{-1}\mathbf{S}^T\mathbf{r}$$

$$\mathbf{h}(n) = \left(\mathbf{S}^T\mathbf{S} + \mathbf{A}\right)^{-1}\mathbf{S}^T\mathbf{r}$$

$$\mathbf{A} = \mathbf{U}\left(e^{n\Lambda} - \mathbf{I}\right)^{-1}\Lambda\mathbf{U}^T$$

A is the covariance matrix of a Gaussian Prior