Bi-Level Path Following for Cross Validated Solution of Kernel Quantile Regression

Saharon Rosset

Department of Statistics
Tel Aviv University

Predictive modeling

- We are given a learning $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$, with $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$ (or $y_i \in \{0, 1\}$), drawn i.i.d from $p(X, Y)$.

Predictive modeling

- We are given a learning $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$, with $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$ (or $y_i \in \{0, 1\}$), drawn i.i.d from $p(X, Y)$.
- Our goal is to learn about the *discriminative* dependence of Y on X, i.e., $P(Y|X)$.

Predictive modeling

- We are given a learning $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$, with $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$ (or $y_i \in \{0, 1\}$), drawn i.i.d from $p(X, Y)$.
- Our goal is to learn about the *discriminative* dependence of Y on X, i.e., $P(Y|X)$.
- Standard learning problems:

Predictive modeling

- We are given a learning $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$, with $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$ (or $y_i \in \{0, 1\}$), drawn i.i.d from $p(X, Y)$.
- Our goal is to learn about the *discriminative* dependence of Y on X, i.e., $P(Y|X)$.
- Standard learning problems:
  - Classification: estimate $f(\mathbf{x}) = P(Y = 1|X = \mathbf{x})$.

Predictive modeling

- We are given a learning $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$, with $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$ (or $y_i \in \{0, 1\}$), drawn i.i.d from $p(X, Y)$.
- Our goal is to learn about the *discriminative* dependence of Y on X, i.e., $P(Y|X)$.
- Standard learning problems:
  - Classification: estimate $f(\mathbf{x}) = P(Y = 1|X = \mathbf{x})$.
  - Regression: estimate $f(\mathbf{x}) = E(Y|X = \mathbf{x})$.

Predictive modeling

- We are given a learning $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$, with $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$ (or $y_i \in \{0, 1\}$), drawn i.i.d from $p(X, Y)$.
- Our goal is to learn about the *discriminative* dependence of Y on X, i.e., $P(Y|X)$.
- Standard learning problems:
  - Classification: estimate $f(\mathbf{x}) = P(Y = 1|X = \mathbf{x})$.
  - Regression: estimate $f(\mathbf{x}) = E(Y|X = \mathbf{x})$.
- What happens if we want to learn more about $P(Y|X)$ than just these expectations?

Regularized optimization

A general approach to solving predictive modeling problems is through regularized optimization:

Regularized optimization

A general approach to solving predictive modeling problems is through regularized optimization:

- Let $\mathcal{F}$ be a family of *candidate models*.

Regularized optimization

A general approach to solving predictive modeling problems is through regularized optimization:

- Let $\mathcal{F}$ be a family of *candidate models*.
- Define a loss function $L(y, f(\mathbf{x}))$ measuring how well $f(\mathbf{x})$ estimates y.

Regularized optimization

A general approach to solving predictive modeling problems is through regularized optimization:

- Let $\mathcal{F}$ be a family of *candidate models*.
- Define a loss function $L(y, f(\mathbf{x}))$ measuring how well $f(\mathbf{x})$ estimates y.
- Define a model complexity penalty $J(f)$

Regularized optimization

A general approach to solving predictive modeling problems is through regularized optimization:

- Let $\mathcal{F}$ be a family of *candidate models*.
- Define a loss function $L(y, f(\mathbf{x}))$ measuring how well $f(\mathbf{x})$ estimates y.
- Define a model complexity penalty $J(f)$
- Now let the optimal model $\hat{f}(\lambda)$ solve:

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) + \lambda J(f)$$

Example: Kernel machines

$$\min_{f \in \mathcal{H}_K} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) + \lambda \|f\|^2_{\mathcal{H}_K}$$

Example: Kernel machines

$$\min_{f \in \mathcal{H}_K} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) + \lambda \|f\|^2_{\mathcal{H}_K}$$

- The model space $\mathcal{F} = \mathcal{H}_K$ is a Reproducing Kernel Hilbert Space (RKHS) of functions generated by a kernel K.

Example: Kernel machines

$$\min_{f \in \mathcal{H}_K} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) + \lambda \| f \|^2_{\mathcal{H}_K}$$

- The model space $\mathcal{F} = \mathcal{H}_K$ is a Reproducing Kernel Hilbert Space (RKHS) of functions generated by a kernel K.
- The penalty is the norm corresponding to the RKHS.

Example: Kernel machines

$$\min_{f \in \mathcal{H}_K} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) + \lambda \|f\|^2_{\mathcal{H}_K}$$

- The model space $\mathcal{F} = \mathcal{H}_K$ is a Reproducing Kernel Hilbert Space (RKHS) of functions generated by a kernel K.
- The penalty is the norm corresponding to the RKHS.
- For Support Vector Machines we use the Hinge loss $L(y, f(\mathbf{x})) = (1 - yf(\mathbf{x}))_+$.

Example: Kernel machines

$$\min_{f \in \mathcal{H}_K} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) + \lambda \|f\|^2_{\mathcal{H}_K}$$

- The model space $\mathcal{F} = \mathcal{H}_K$ is a Reproducing Kernel Hilbert Space (RKHS) of functions generated by a kernel K.
- The penalty is the norm corresponding to the RKHS.
- For Support Vector Machines we use the Hinge loss $L(y, f(\mathbf{x})) = (1 - yf(\mathbf{x}))_+$.

Can be solved efficiently because of the Representer Theorem.

Example: Kernel machines

$$\min_{f \in \mathcal{H}_K} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) + \lambda \|f\|^2_{\mathcal{H}_K}$$

- The model space $\mathcal{F} = \mathcal{H}_K$ is a Reproducing Kernel Hilbert Space (RKHS) of functions generated by a kernel K.
- The penalty is the norm corresponding to the RKHS.
- For Support Vector Machines we use the Hinge loss $L(y, f(\mathbf{x})) = (1 - yf(\mathbf{x}))_+$.

The optimal solution has the form: $\hat{f}(\lambda)(\mathbf{x}) = \sum_{j=1}^{n} \hat{\alpha}_j K(\mathbf{x}, \mathbf{x}_j)$.

The corresponding penalty is: $\|f\|^2_{\mathcal{H}_K} = \alpha^T K \alpha$.

Model selection

$$\min_{f in \mathcal{F}} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) + \lambda J(f)$$

- The regularization parameter trades off between fitting and complexity control.

Model selection

$$\min_{f in \mathcal{F}} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) + \lambda J(f)$$

- The regularization parameter trades off between fitting and complexity control.
  - We want to select a value λ^* such that $\hat{f}(\lambda^*)$ will do well in prediction.

Model selection

$$\min_{f in \mathcal{F}} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) + \lambda J(f)$$

- The regularization parameter trades off between fitting and complexity control.
  - We want to select a value λ^* such that $\hat{f}(\lambda^*)$ will do well in prediction.
- Approaches for selecting λ^*:

Model selection

$$\min_{f in \mathcal{F}} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) + \lambda J(f)$$

- The regularization parameter trades off between fitting and complexity control.
  - We want to select a value λ^* such that $\hat{f}(\lambda^*)$ will do well in prediction.
- Approaches for selecting λ^*:
  - Based on in-sample estimates of prediction error: Structural Risk Minimization, AIC, BIC,...

Model selection

$$\min_{f in \mathcal{F}} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) + \lambda J(f)$$

- The regularization parameter trades off between fitting and complexity control.
  - We want to select a value λ^* such that $\hat{f}(\lambda^*)$ will do well in prediction.
- Approaches for selecting λ^*:
  - Based on in-sample estimates of prediction error: Structural Risk Minimization, AIC, BIC,...
  - Using hold-out data (cross validation).

Regularization path

- The set of solutions $\hat{f}(\lambda)$ can be thought of as a curve in the space of possible models $\mathcal{F}$.

Regularization path

- The set of solutions $\hat{f}(\lambda)$ can be thought of as a curve in the space of possible models $\mathcal{F}$.
- As it turns out for many choices of L, J this curve has nice geometric structure.

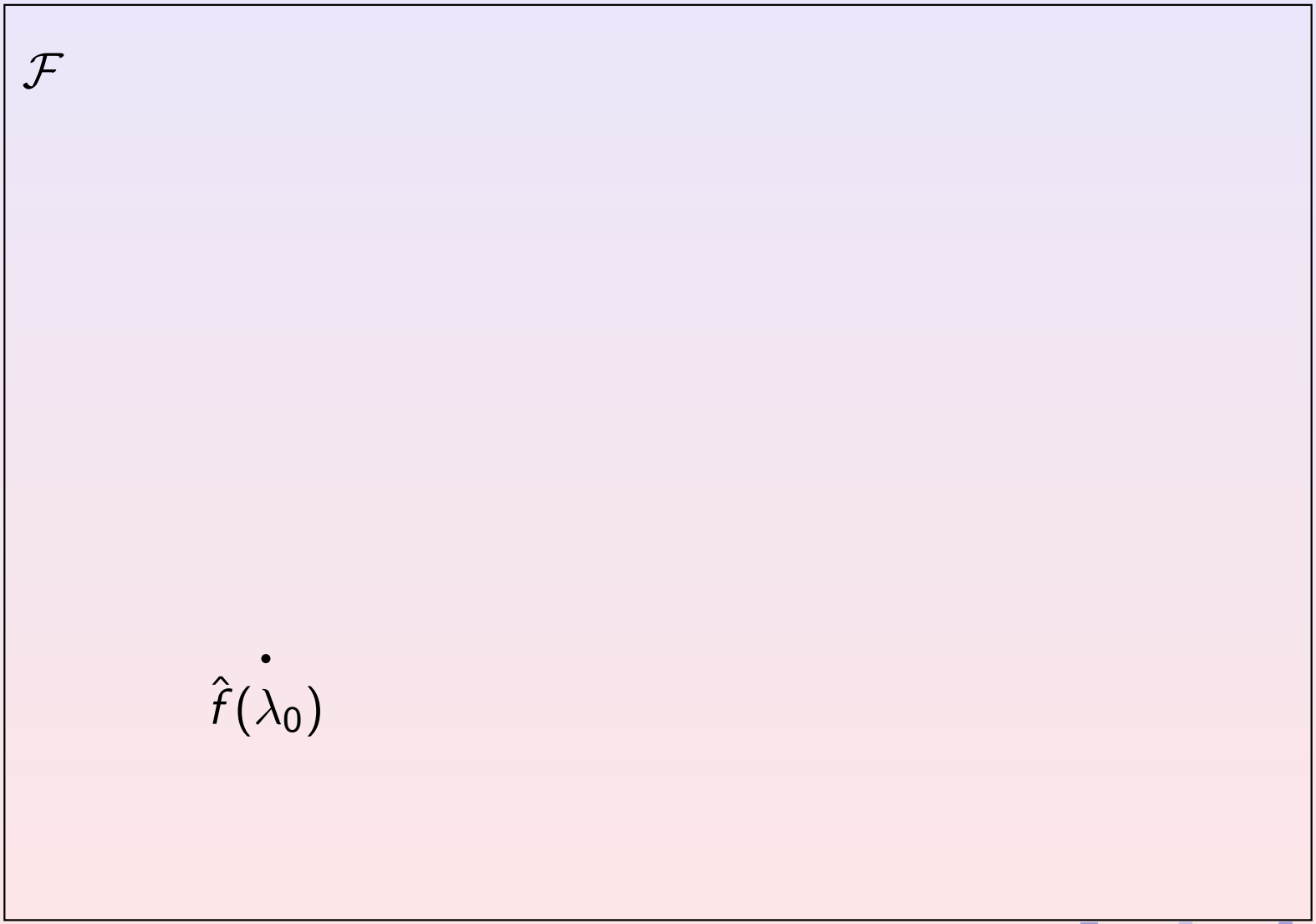
Regularization path

- The set of solutions $\hat{f}(\lambda)$ can be thought of as a curve in the space of possible models $\mathcal{F}$.
- As it turns out for many choices of L, J this curve has nice geometric structure.
- This gives rise to homotopy methods for generating the whole set of solutions $\{\hat{f}(\lambda) : 0 \leq \lambda \leq \infty\}$ incrementally and efficiently.

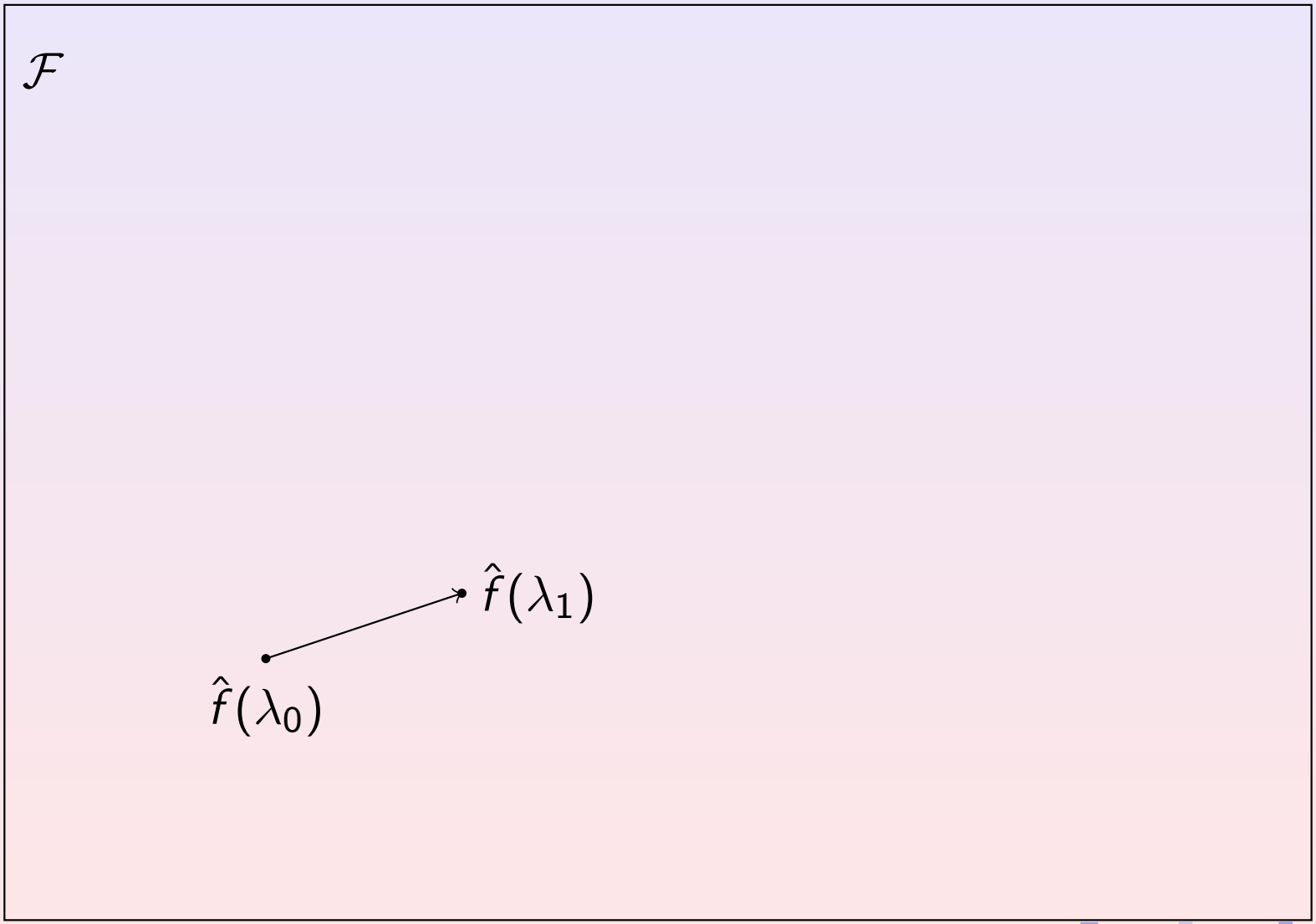
Regularization path

- The set of solutions $\hat{f}(\lambda)$ can be thought of as a curve in the space of possible models $\mathcal{F}$.
- As it turns out for many choices of L, J this curve has nice geometric structure.
- This gives rise to homotopy methods for generating the whole set of solutions $\{\hat{f}(\lambda) : 0 \leq \lambda \leq \infty\}$ incrementally and efficiently.
- Such algorithms have been developed, among others, for:
  - Lasso (Efron et al. 2004)
  - Kernel SVM (Hastie et al. 2004)
  - Kernel quantile regression (Li et al. 2007)

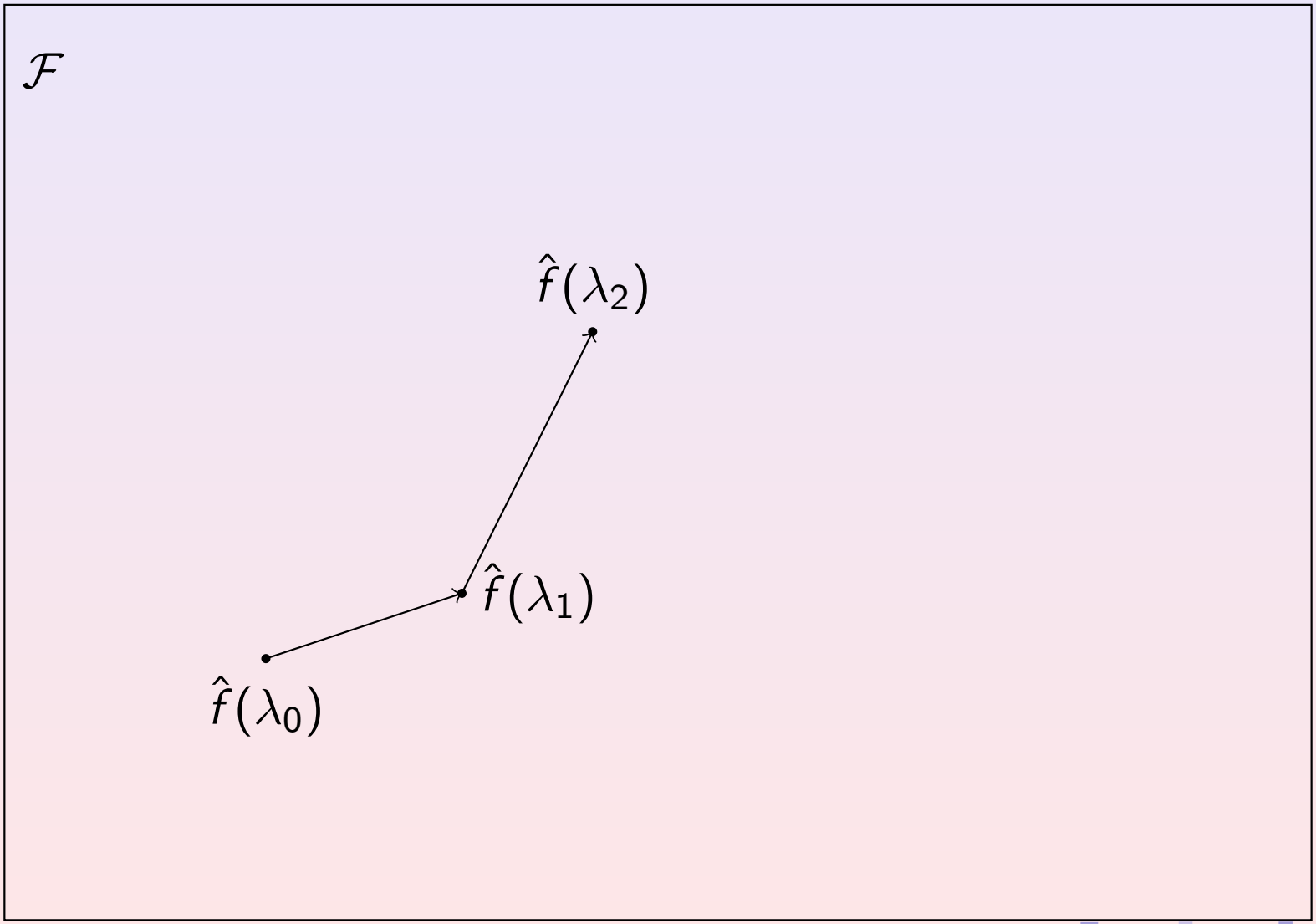
Regularization path schematic



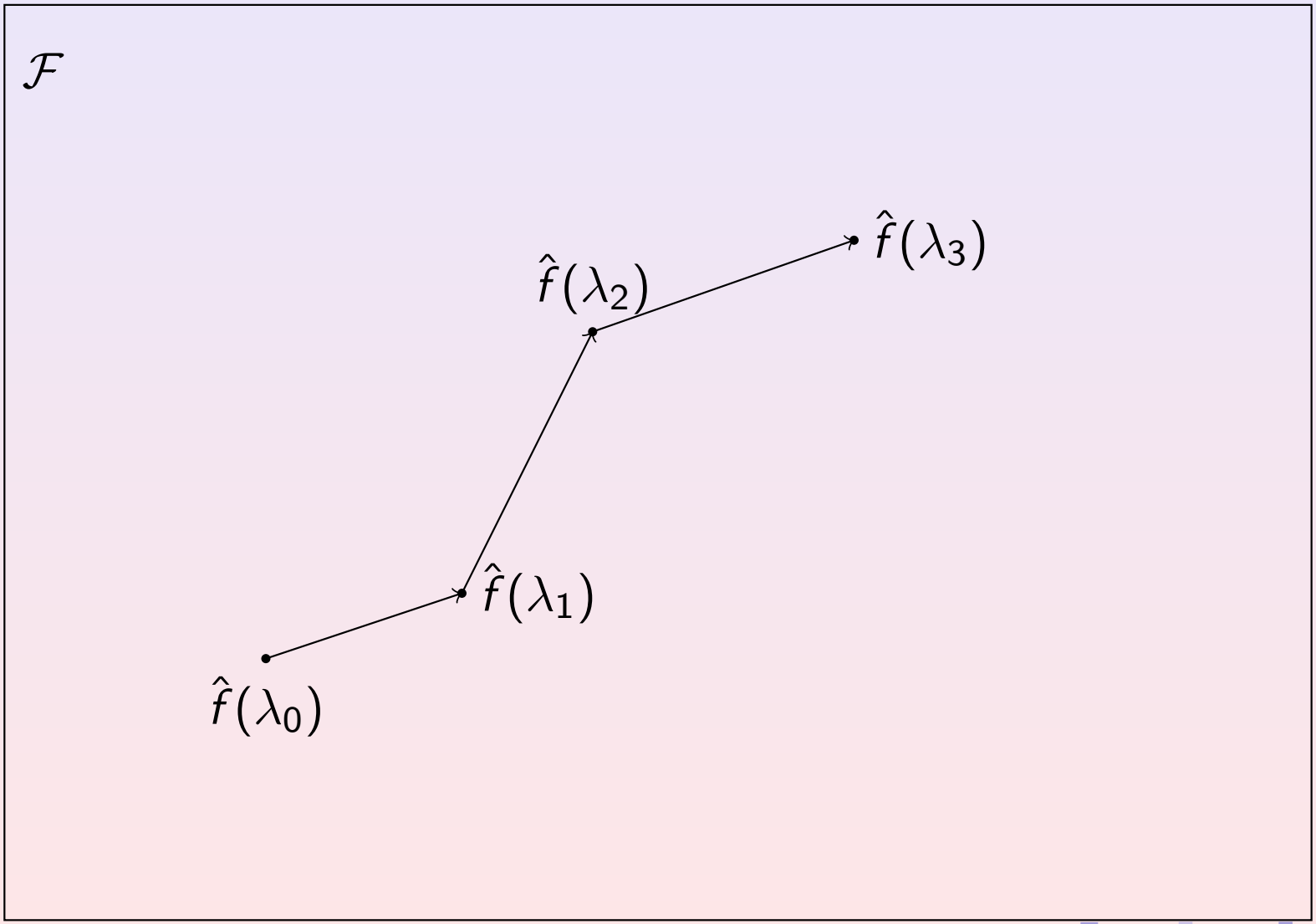
Regularization path schematic



Regularization path schematic



Regularization path schematic



Cross validation as a bi-level programming problem

Assume we want to select a value for λ by comparing the Loss L of $\hat{f}(\lambda)$ for different values of λ on a holdout sample $\{(\tilde{\mathbf{x}}_1, \tilde{y}_1), ..., (\tilde{\mathbf{x}}_N, \tilde{y}_N)\}$:

$$\lambda^* = \arg\min_{\lambda} \sum_{i=1}^{N} L(\tilde{y}_i, \hat{f}(\lambda)(\tilde{\mathbf{x}}_i))$$

Cross validation as a bi-level programming problem

Assume we want to select a value for λ by comparing the Loss L of $\hat{f}(\lambda)$ for different values of λ on a holdout sample $\{(\tilde{\mathbf{x}}_1, \tilde{y}_1), ..., (\tilde{\mathbf{x}}_N, \tilde{y}_N)\}$:

$$\lambda^* = \arg\min_{\lambda} \sum_{i=1}^{N} L(\tilde{y}_i, \hat{f}(\lambda)(\tilde{\mathbf{x}}_i))$$

- This is a bi-level programming problem, since $\hat{f}(\lambda)$ is itself a solution of an inner-level regularized optimization problem!

Cross validation as a bi-level programming problem

Assume we want to select a value for λ by comparing the Loss L of $\hat{f}(\lambda)$ for different values of λ on a holdout sample $\{(\tilde{\mathbf{x}}_1, \tilde{y}_1), ..., (\tilde{\mathbf{x}}_N, \tilde{y}_N)\}$:

$$\lambda^* = \arg\min_{\lambda} \sum_{i=1}^{N} L(\tilde{y}_i, \hat{f}(\lambda)(\tilde{\mathbf{x}}_i))$$

- This is a bi-level programming problem, since $\hat{f}(\lambda)$ is itself a solution of an inner-level regularized optimization problem!
  - Naive solution: find the regularization path, and try many values of $\hat{f}(\lambda)$ on the holdout data.

Cross validation as a bi-level programming problem

Assume we want to select a value for λ by comparing the Loss L of $\hat{f}(\lambda)$ for different values of λ on a holdout sample $\{(\tilde{\mathbf{x}}_1, \tilde{y}_1), ..., (\tilde{\mathbf{x}}_N, \tilde{y}_N)\}$:

$$\lambda^* = \arg\min_{\lambda} \sum_{i=1}^{N} L(\tilde{y}_i, \hat{f}(\lambda)(\tilde{\mathbf{x}}_i))$$

- This is a bi-level programming problem, since $\hat{f}(\lambda)$ is itself a solution of an inner-level regularized optimization problem!
  - Naive solution: find the regularization path, and try many values of $\hat{f}(\lambda)$ on the holdout data.
  - Alternative: find a way to optimize the holdout loss efficiently.

Cross validation as a bi-level programming problem

Assume we want to select a value for λ by comparing the Loss L of $\hat{f}(\lambda)$ for different values of λ on a holdout sample $\{(\tilde{\mathbf{x}}_1, \tilde{y}_1), ..., (\tilde{\mathbf{x}}_N, \tilde{y}_N)\}$:

$$\lambda^* = \arg\min_{\lambda} \sum_{i=1}^{N} L(\tilde{y}_i, \hat{f}(\lambda)(\tilde{\mathbf{x}}_i))$$

- This is a bi-level programming problem, since $\hat{f}(\lambda)$ is itself a solution of an inner-level regularized optimization problem!
  - Naive solution: find the regularization path, and try many values of $\hat{f}(\lambda)$ on the holdout data.
  - Alternative: find a way to optimize the holdout loss efficiently.
  - Problem: even if the inner problem is convex for every λ, the bi-level problem is not convex!

Conditional quantile modeling

If $Y \in \mathbb{R}$, we may want to learn about $P(Y|X = \mathbf{x})$ through its quantiles $f_\tau(\mathbf{x})$ defined by: $P(Y \leq f_\tau(\mathbf{x})|X = \mathbf{x}) = \tau$.

Conditional quantile modeling

If $Y \in \mathbb{R}$, we may want to learn about $P(Y|X = \mathbf{x})$ through its quantiles $f_\tau(\mathbf{x})$ defined by: $P(Y \leq f_\tau(\mathbf{x})|X = \mathbf{x}) = \tau$.
Interested communities:

- Traditionally: Economics and Education, trying to understand the details of how X (e.g. Socio-Economic background) affects Y (e.g. salary).

Conditional quantile modeling

If $Y \in \mathbb{R}$, we may want to learn about $P(Y|X = \mathbf{x})$ through its quantiles $f_\tau(\mathbf{x})$ defined by: $P(Y \leq f_\tau(\mathbf{x})|X = \mathbf{x}) = \tau$.
Interested communities:

- Traditionally: Economics and Education, trying to understand the details of how X (e.g. Socio-Economic background) affects Y (e.g. salary).
- Recently: increasing methodological interest in Machine Learning and Statistics communities.

Conditional quantile modeling

If $Y \in \mathbb{R}$, we may want to learn about $P(Y|X = \mathbf{x})$ through its quantiles $f_\tau(\mathbf{x})$ defined by: $P(Y \leq f_\tau(\mathbf{x})|X = \mathbf{x}) = \tau$.
Interested communities:

- Traditionally: Economics and Education, trying to understand the details of how X (e.g. Socio-Economic background) affects Y (e.g. salary).
- Recently: increasing methodological interest in Machine Learning and Statistics communities.

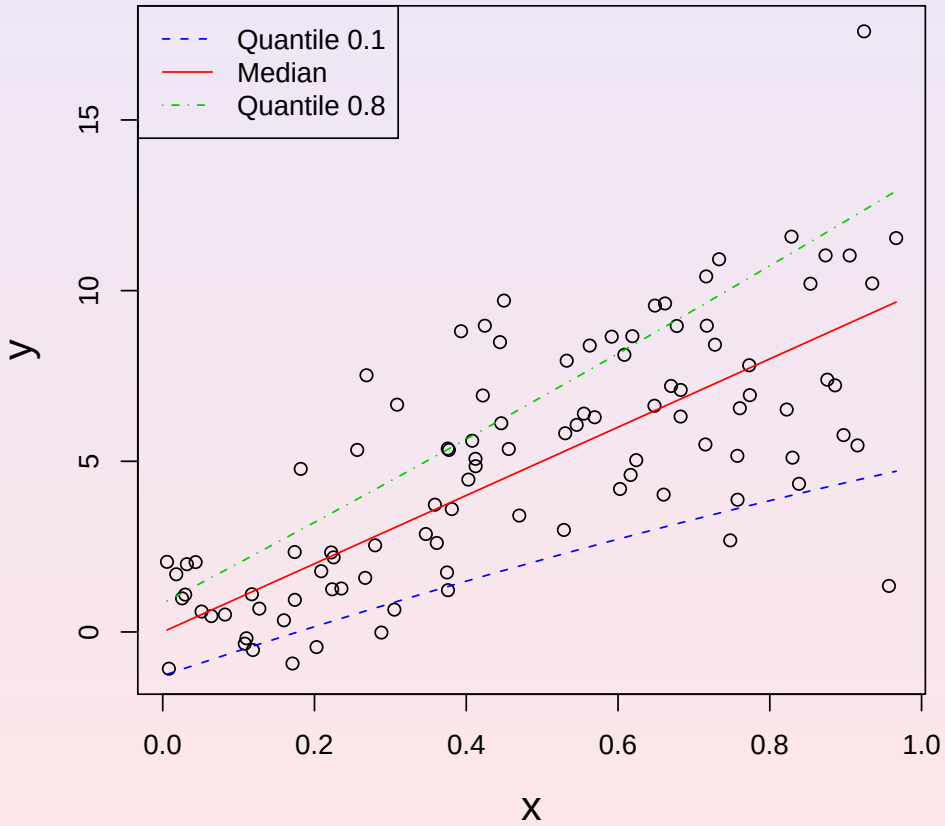
Conditional quantile modeling

If $Y \in \mathbb{R}$, we may want to learn about $P(Y|X = \mathbf{x})$ through its quantiles $f_\tau(\mathbf{x})$ defined by: $P(Y \leq f_\tau(\mathbf{x})|X = \mathbf{x}) = \tau$.
Interested communities:

- Traditionally: Economics and Education, trying to understand the details of how X (e.g. Socio-Economic background) affects Y (e.g. salary).
- Recently: increasing methodological interest in Machine Learning and Statistics communities.

In simple cases, such as the classic model $Y = f(\mathbf{x}) + \epsilon$, ϵ i.i.d, modeling $f_\tau(\mathbf{x})$ for multiple values of τ is redundant.

Non-uniform noise ⇒ Non parallel quantiles

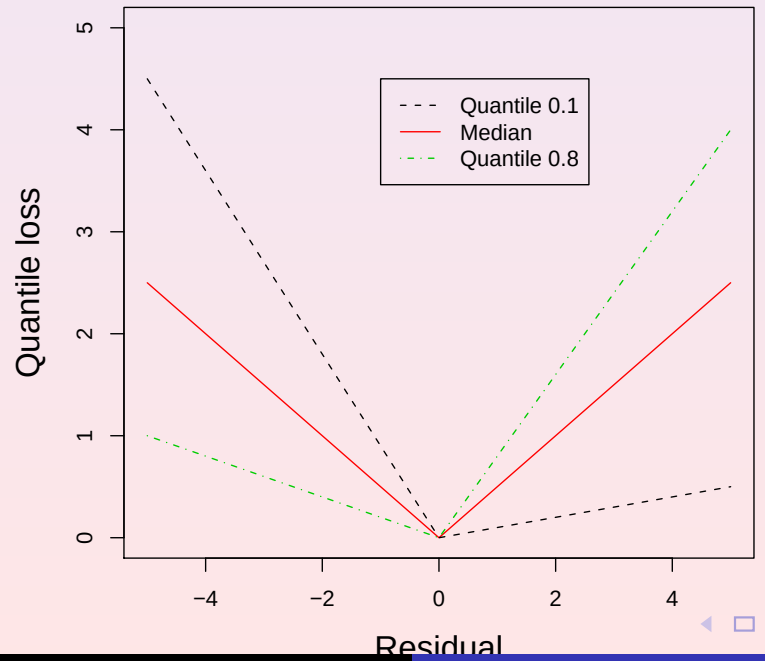


Quantile regression

The quantile loss function is:

$$L_\tau(r) = \begin{cases} r\tau & r \geq 0 \\ -r(1-\tau) & r < 0 \end{cases}$$

(where $r = y - f(\mathbf{x})$ is the residual)



Quantile regression and kernel quantile regression

This quantile loss function has the appropriate population optimizer:

$$\arg\min_{f(\mathbf{x})} E(L_\tau(y - f(\mathbf{x}))) = f_\tau(\mathbf{x})$$

Quantile regression and kernel quantile regression

This quantile loss function has the appropriate population optimizer:

$$\arg\min_{f(\mathbf{x})} E(L_\tau(y - f(\mathbf{x}))) = f_\tau(\mathbf{x})$$

We can use this loss function to create regularized quantile regression problems:

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{n} L_\tau(y_i - f(\mathbf{x}_i)) + \lambda J(f)$$

Quantile regression and kernel quantile regression

This quantile loss function has the appropriate population optimizer:

$$\arg\min_{f(\mathbf{x})} E(L_\tau(y - f(\mathbf{x}))) = f_\tau(\mathbf{x})$$

We can use this loss function to create regularized quantile regression problems:

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{n} L_\tau(y_i - f(\mathbf{x}_i)) + \lambda J(f)$$

We are going to concentrate on the kernel quantile regression (KQR) version:

$$\min_{\alpha \in \mathbb{R}^n} \sum_{i=1}^{n} L_\tau(y_i, -\sum_{j=1}^{n} \alpha_j K(\mathbf{x}_i, \mathbf{x}_j)) + \lambda \alpha^{\mathsf{T}} K \alpha$$

Bi-level programming with a parameterized loss function

KQR in fact has two parameters which can be modified: the quantile τ and the regularization parameter λ.

Bi-level programming with a parameterized loss function

KQR in fact has two parameters which can be modified: the quantile τ and the regularization parameter λ.
However the two parameters have vert different roles:

- Every value of τ defined a loss function L_τ and a corresponding quantile modeling problem.
- λ balances loss and penalty to optimize predictive performance.

Bi-level programming with a parameterized loss function

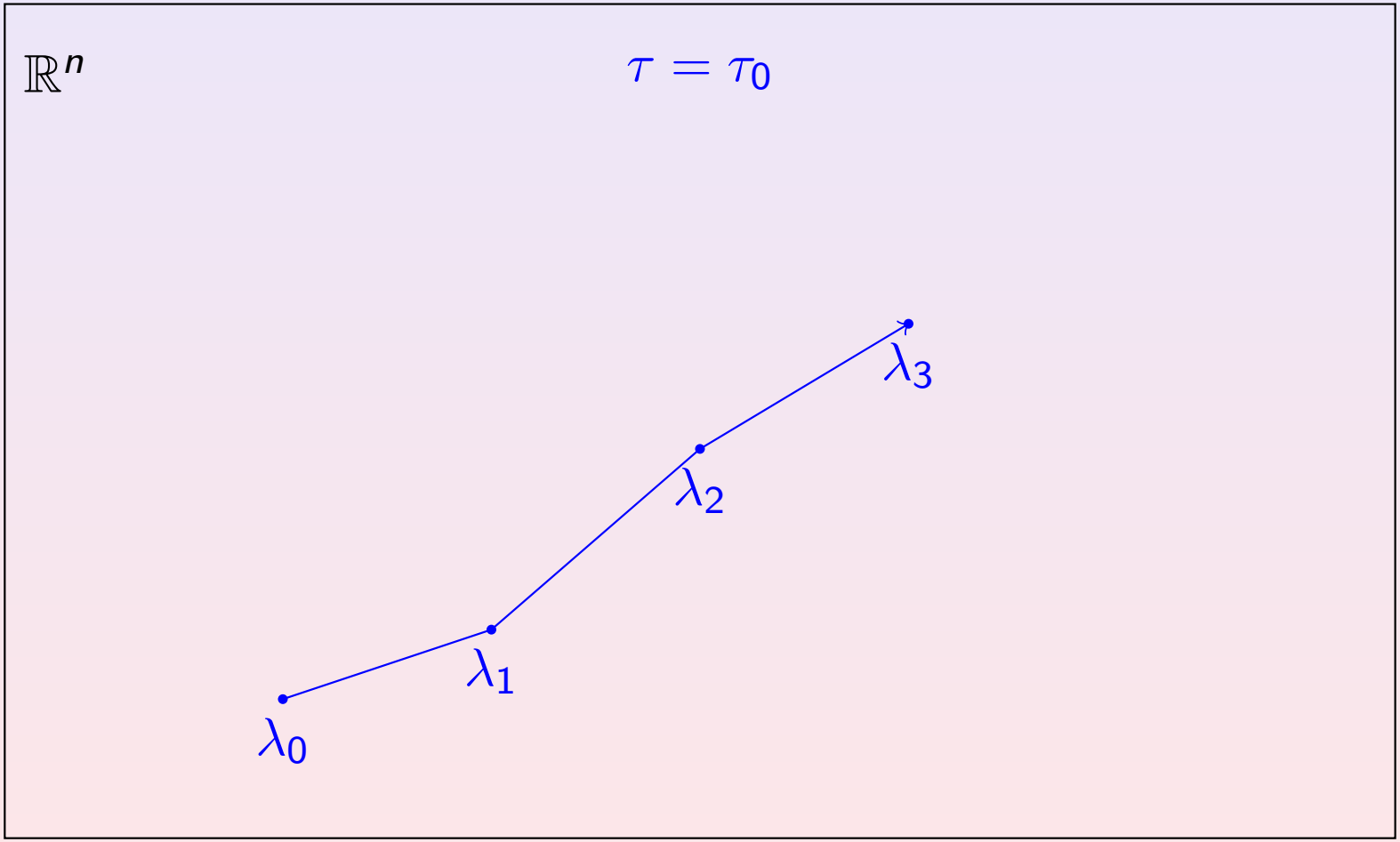
KQR in fact has two parameters which can be modified: the quantile τ and the regularization parameter λ.
However the two parameters have vert different roles:

- Every value of τ defined a loss function L_τ and a corresponding quantile modeling problem.
- λ balances loss and penalty to optimize predictive performance.

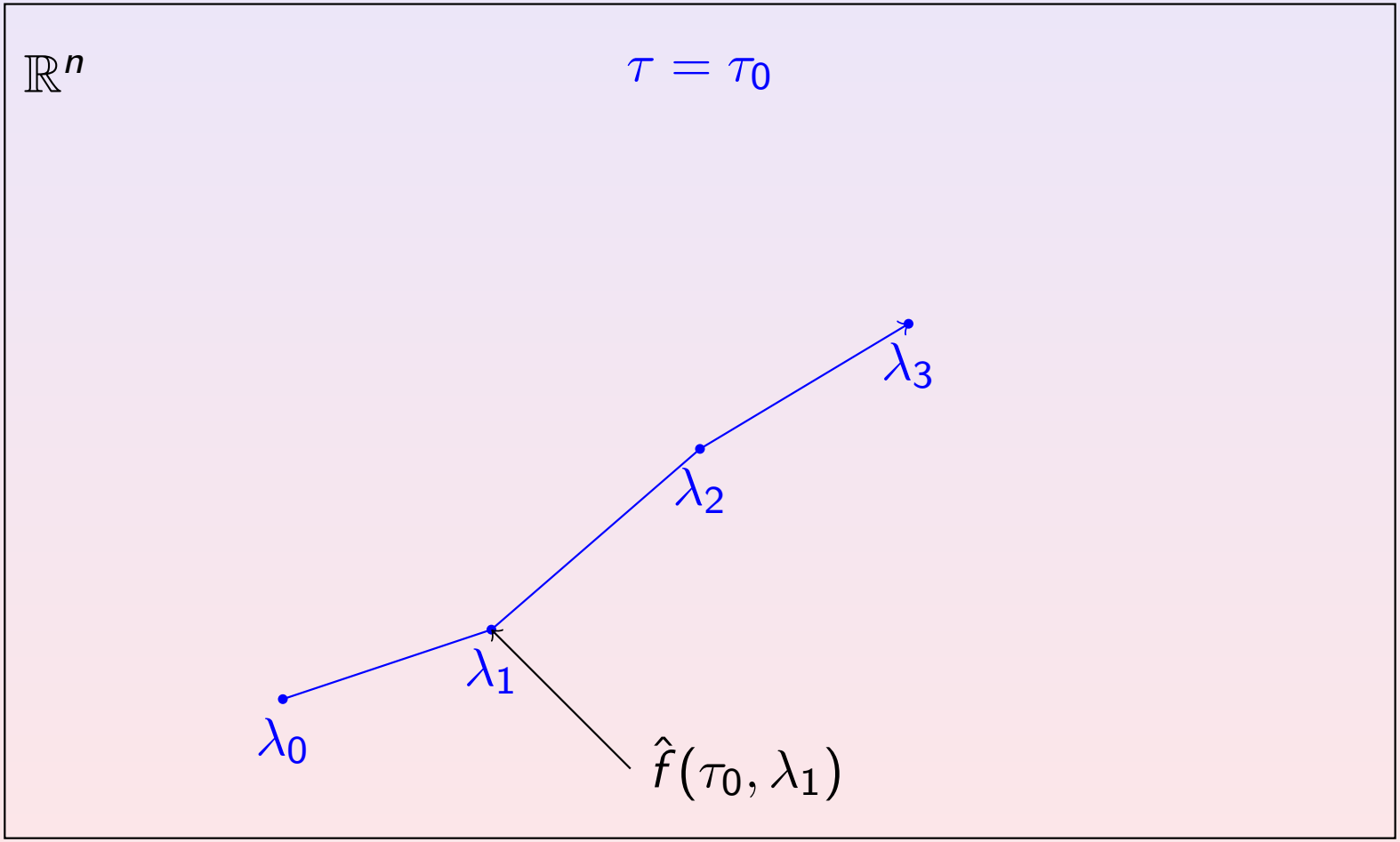
We would typically want to solve a bi-level problem to select an optimal regularizer for every quantile:

$$\lambda^*(\tau) = \arg\min_{\lambda} \sum_{i=1}^{N} L_\tau(\tilde{y}_i - \hat{f}(\lambda)(\tilde{\mathbf{x}}_i))$$

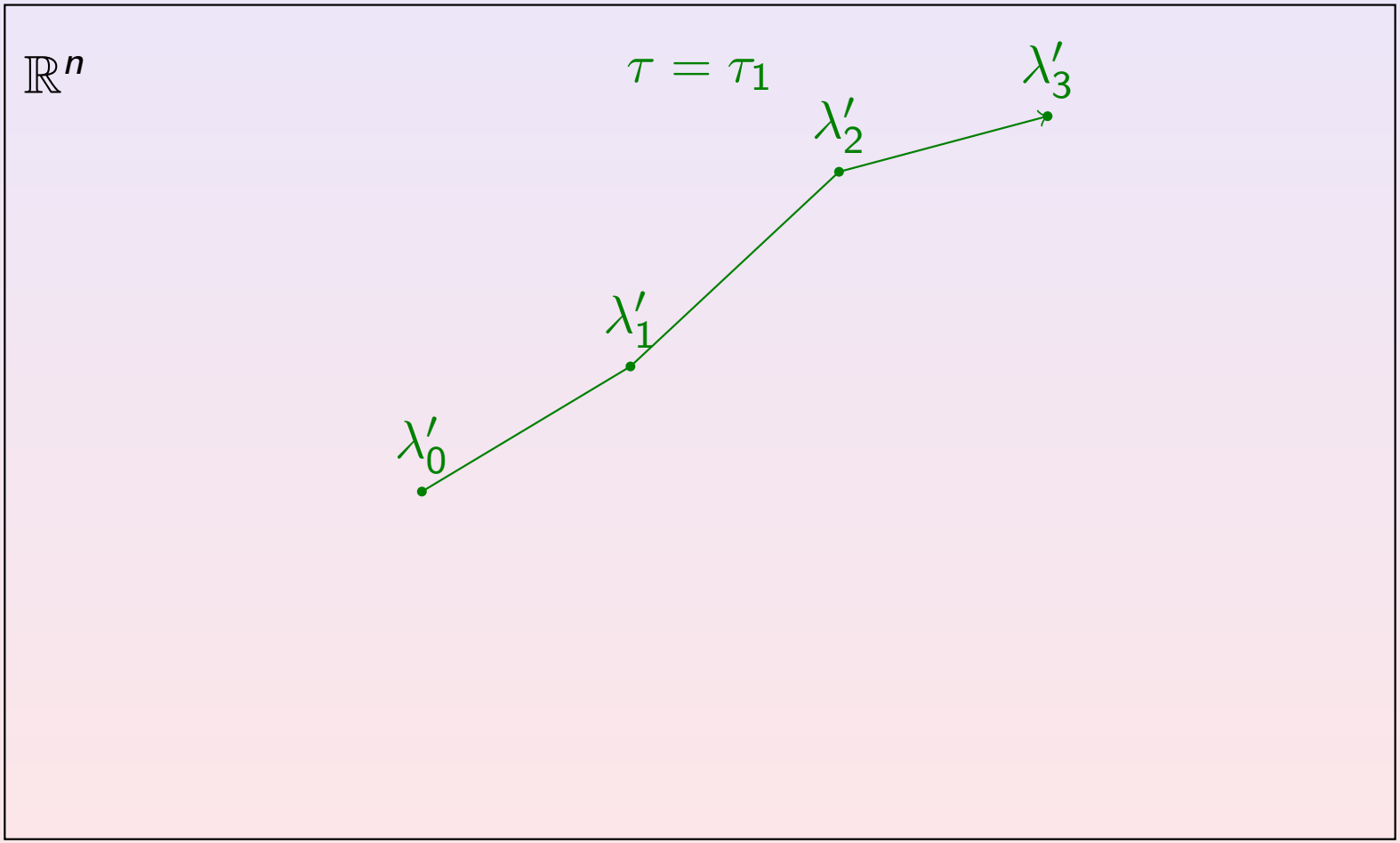
Regularization paths as τ and λ vary



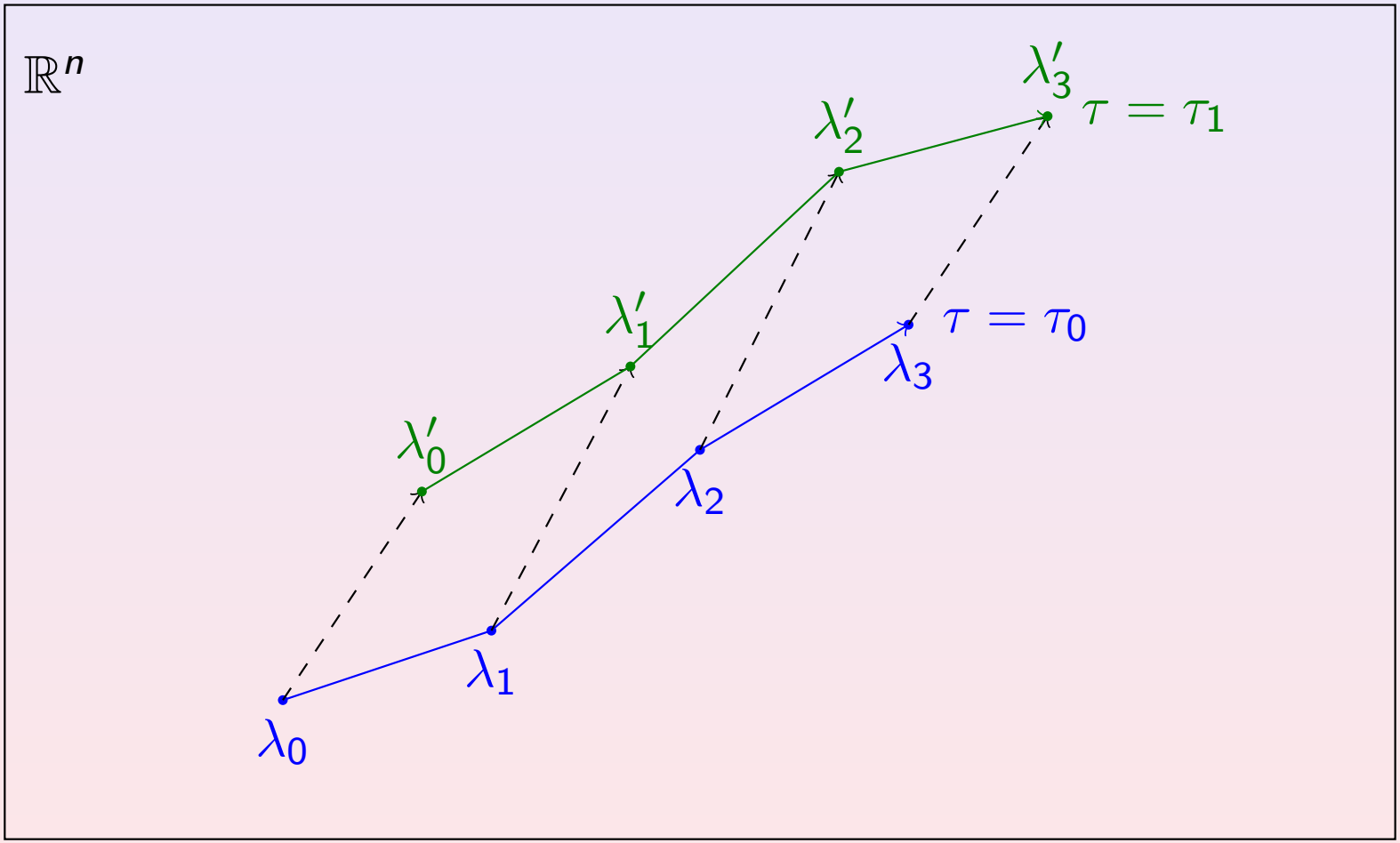
Regularization paths as τ and λ vary



Regularization paths as τ and λ vary



Regularization paths as τ and λ vary



Mapping $\hat{f}(\tau, \lambda)$

Input: a starting solution path for τ_0.

Mapping $\hat{f}(\tau, \lambda)$

Input: a starting solution path for τ_0.
Output: generation of solutions paths for all values of τ.

Mapping $\hat{f}(\tau, \lambda)$

Input: a starting solution path for τ_0.
Output: generation of solutions paths for all values of τ.
Method:

- Identify all knots in the path for τ_0.

Mapping $\hat{f}(\tau, \lambda)$

Input: a starting solution path for τ_0.
Output: generation of solutions paths for all values of τ.
Method:

- Identify all knots in the path for τ_0.
- Follow each knot as τ changes.

Mapping $\hat{f}(\tau, \lambda)$

Input: a starting solution path for τ_0.
Output: generation of solutions paths for all values of τ.
Method:

- Identify all knots in the path for τ_0.
- Follow each knot as τ changes.
- Identify events:

Mapping $\hat{f}(\tau, \lambda)$

Input: a starting solution path for τ_0.
Output: generation of solutions paths for all values of τ.
Method:

- Identify all knots in the path for τ_0.
- Follow each knot as τ changes.
- Identify events:
  - Knot crossings.

Mapping $\hat{f}(\tau, \lambda)$

Input: a starting solution path for τ_0.
Output: generation of solutions paths for all values of τ.
Method:

- Identify all knots in the path for τ_0.
- Follow each knot as τ changes.
- Identify events:
  - Knot crossings.
  - Knot merge.

Mapping $\hat{f}(\tau, \lambda)$

Input: a starting solution path for τ_0.
Output: generation of solutions paths for all values of τ.
Method:

- Identify all knots in the path for τ_0.
- Follow each knot as τ changes.
- Identify events:
  - Knot crossings.
  - Knot merge.
  - Knot split.

Mapping $\hat{f}(\tau, \lambda)$

Input: a starting solution path for τ_0.
Output: generation of solutions paths for all values of τ.
Method:

- Identify all knots in the path for τ_0.
- Follow each knot as τ changes.
- Identify events:
  - Knot crossings.
  - Knot merge.
  - Knot split.
- When each event happens, update directions of knots.

Knot evolution: some details

The KKT conditions for the KQR problem give us:

- Any knot λ_k moves linearly as τ changes.
- The regularization path at every τ can be fully described by the knots (in the appropriate representation).
- The fit at the moving knot:

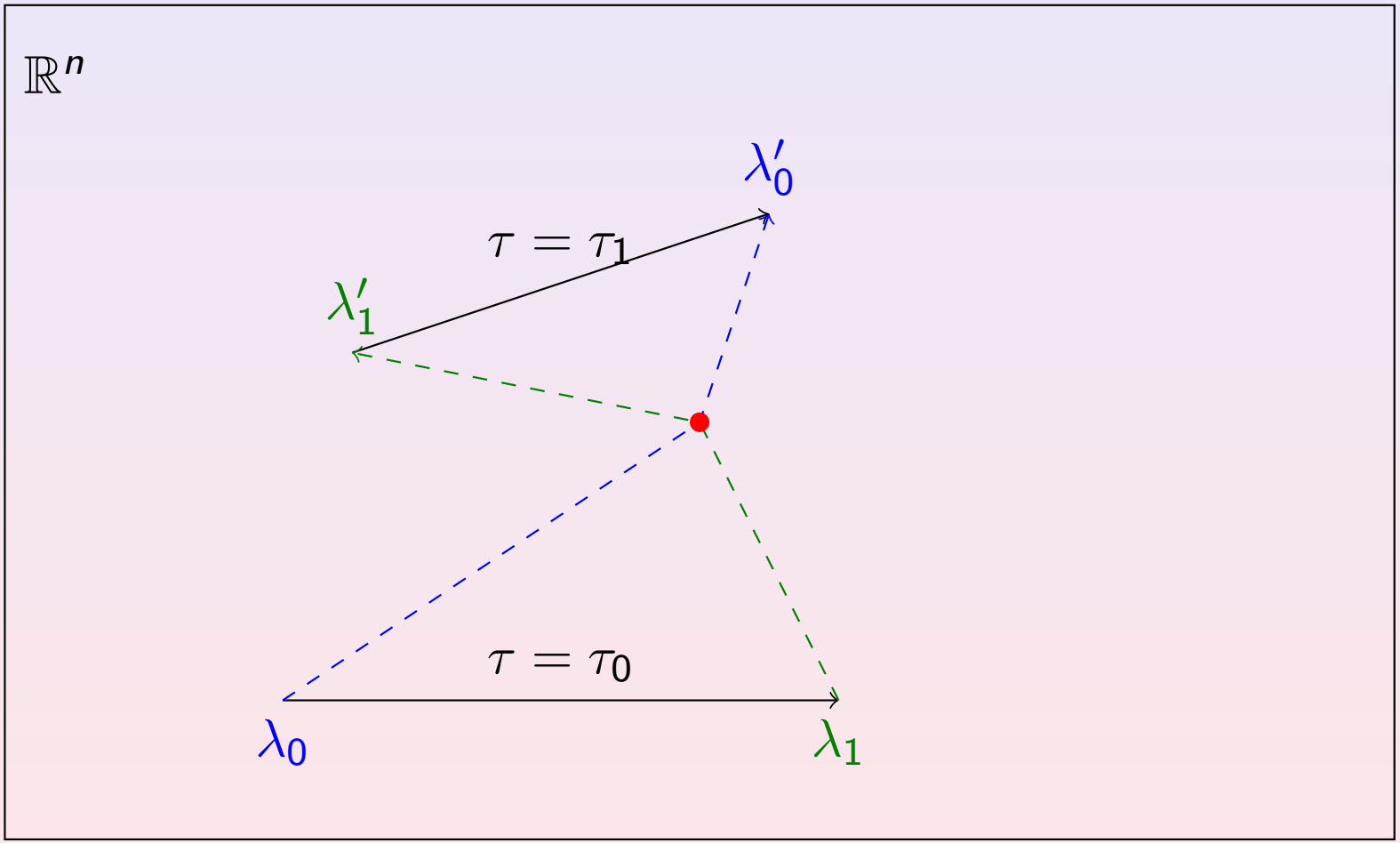
$$\hat{f}(\tau_0 + \delta, \lambda_k + c_k\delta) = \frac{1}{\lambda_k + c_k\delta}\left(\lambda_k \hat{f}(\lambda_k, \tau_0)(\mathbf{x}) + \delta h_k(\mathbf{x})\right)$$

$$h_k(\mathbf{x}) = \tilde{b}_0^k + \sum_{i \in \mathcal{E}_k - i_k} \tilde{b}_i^k K(\mathbf{x}, \mathbf{x}_i) + \sum_{i \in \mathcal{L} \cup \mathcal{R} \cup \{i_k\}} K(\mathbf{x}, \mathbf{x}_i)$$

- The direction $\tilde{\mathbf{b}}_k$ and rate c_k are determined by solving a set of linear equations:

$$\mathbf{B}^k \begin{pmatrix} \tilde{\mathbf{b}}^k \\ c_k \end{pmatrix} = \begin{pmatrix} -(|\mathcal{R}| + |\mathcal{L}| + 1) \\ -\mathbf{s}_{\mathcal{E}_k} \end{pmatrix}$$

Knot crossing example



Properties of the bi-level solutions

- The bi-level problems are not convex.

Properties of the bi-level solutions

- The bi-level problems are not convex.
- However, optimal bi-level solutions can only occur in non-differentiability points of cross-validated objective (candidates):

Properties of the bi-level solutions

- The bi-level problems are not convex.
- However, optimal bi-level solutions can only occur in non-differentiability points of cross-validated objective (candidates):
  - Knots in $\hat{f}$.

Properties of the bi-level solutions

- The bi-level problems are not convex.
- However, optimal bi-level solutions can only occur in non-differentiability points of cross-validated objective (candidates):
  - Knots in $\hat{f}$.
  - Cross validation observations crossing 0.

Properties of the bi-level solutions

- The bi-level problems are not convex.
- However, optimal bi-level solutions can only occur in non-differentiability points of cross-validated objective (candidates):
  - Knots in $\hat{f}$.
  - Cross validation observations crossing 0.
- As τ changes they stay in the same candidate for a while, then jump to another.

Properties of the bi-level solutions

- The bi-level problems are not convex.
- However, optimal bi-level solutions can only occur in non-differentiability points of cross-validated objective (candidates):
  - Knots in $\hat{f}$.
  - Cross validation observations crossing 0.
- As τ changes they stay in the same candidate for a while, then jump to another.

Properties of the bi-level solutions

- The bi-level problems are not convex.
- However, optimal bi-level solutions can only occur in non-differentiability points of cross-validated objective (candidates):
  - Knots in $\hat{f}$.
  - Cross validation observations crossing 0.
- As τ changes they stay in the same candidate for a while, then jump to another.

These properties lead to an algorithm for following the complete set of cross-validated solutions $\hat{f}(\tau, \lambda^*(\tau))$, with two components:

1. Following the events as τ changes.

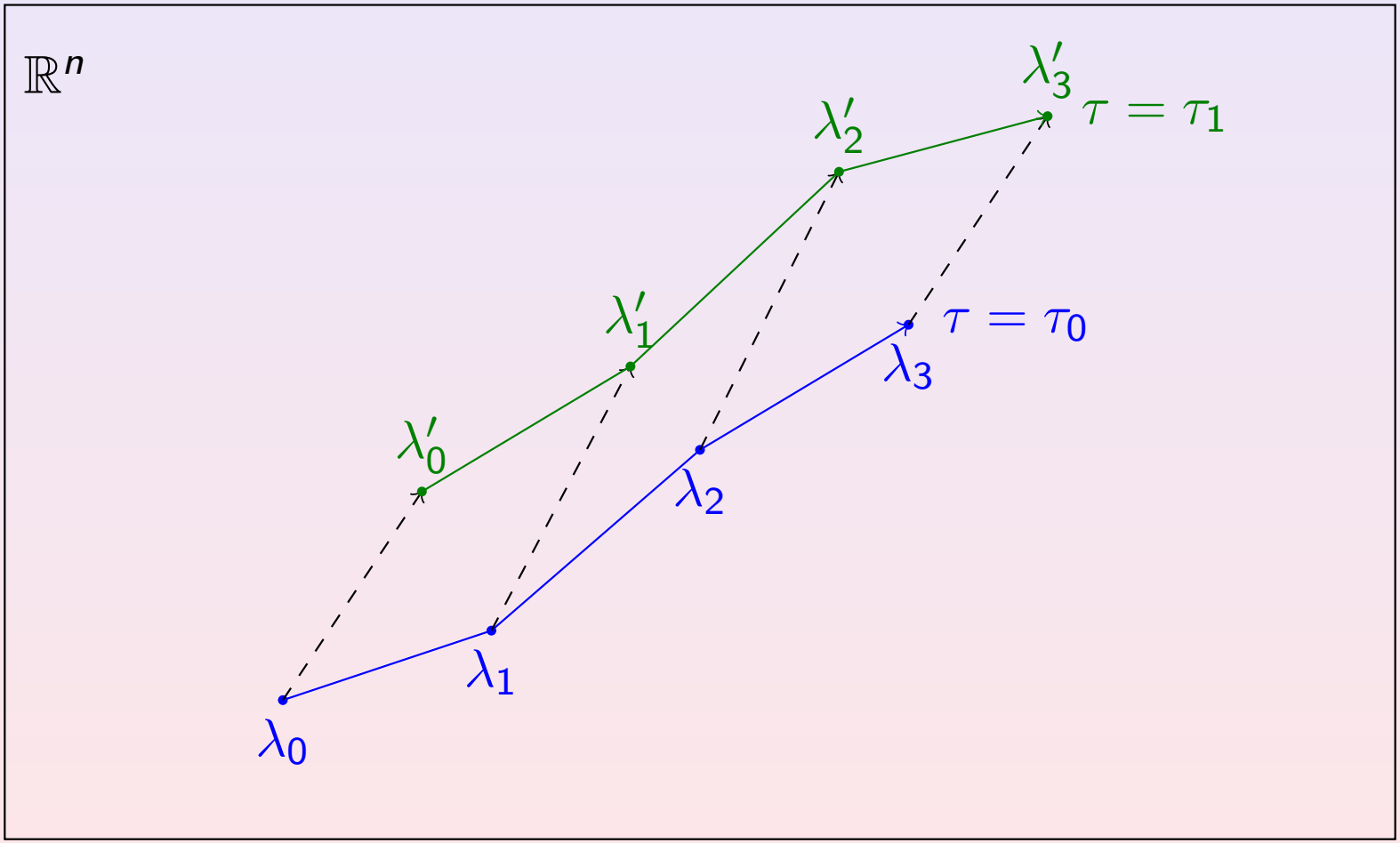
Properties of the bi-level solutions

- The bi-level problems are not convex.
- However, optimal bi-level solutions can only occur in non-differentiability points of cross-validated objective (candidates):
  - Knots in $\hat{f}$.
  - Cross validation observations crossing 0.
- As τ changes they stay in the same candidate for a while, then jump to another.

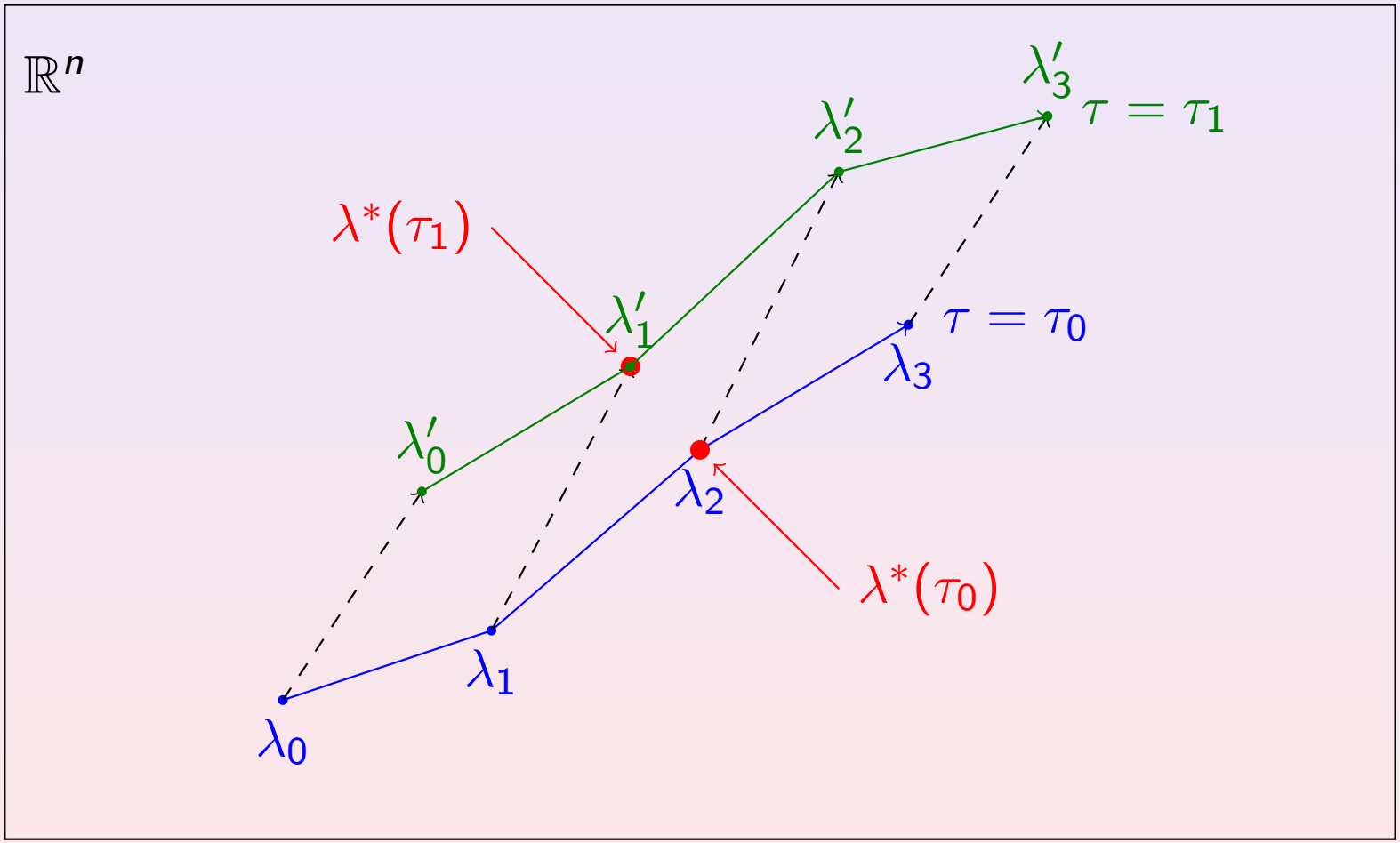
These properties lead to an algorithm for following the complete set of cross-validated solutions $\hat{f}(\tau, \lambda^*(\tau))$, with two components:

1. Following the events as τ changes.
2. Keeping track of the bi-level candidates and their cross-validated scores.

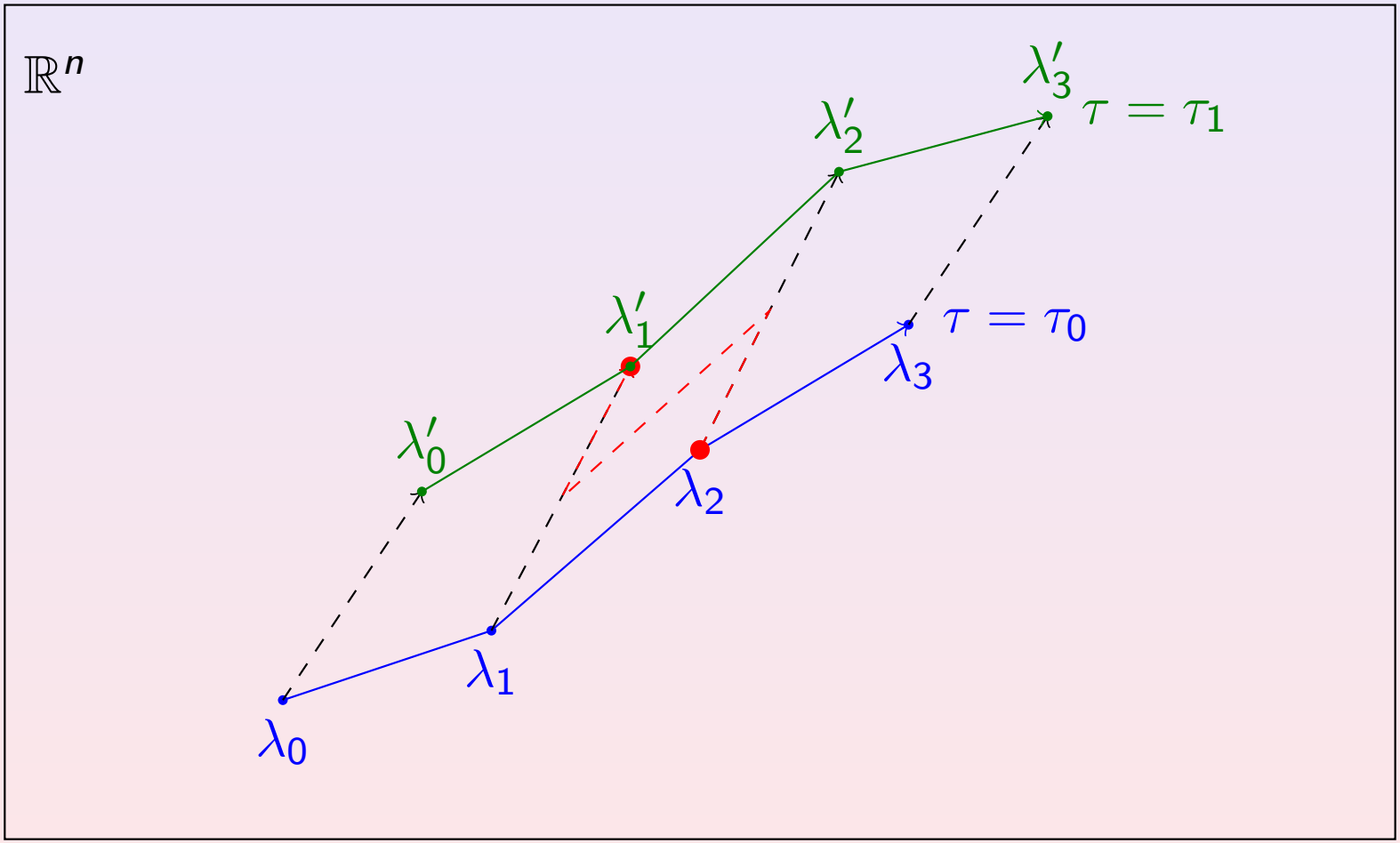
Evolution of bi-level solution



Evolution of bi-level solution



Evolution of bi-level solution



Schematic of complete algorithm

As before, start from a solution path for τ_0.

Schematic of complete algorithm

As before, start from a solution path for τ_0.

- Identify all knots in the path for τ_0.

Schematic of complete algorithm

As before, start from a solution path for τ_0.

- Identify all knots in the path for τ_0.
- Follow each knot as τ changes.

Schematic of complete algorithm

As before, start from a solution path for τ_0.

- Identify all knots in the path for τ_0.
- Follow each knot as τ changes.
- Identify events.

Schematic of complete algorithm

As before, start from a solution path for τ_0.

- Identify all knots in the path for τ_0.
- Follow each knot as τ changes.
- Identify events.
- Between events:

Schematic of complete algorithm

As before, start from a solution path for τ_0.

- Identify all knots in the path for τ_0.
- Follow each knot as τ changes.
- Identify events.
- Between events:
  - Track all candidates for bi-level solution, and their cross validated objective.

Schematic of complete algorithm

As before, start from a solution path for τ_0.

- Identify all knots in the path for τ_0.
- Follow each knot as τ changes.
- Identify events.
- Between events:
  - Track all candidates for bi-level solution, and their cross validated objective.
  - Identify points where the optimum jumps between candidates.

Schematic of complete algorithm

As before, start from a solution path for τ_0.

- Identify all knots in the path for τ_0.
- Follow each knot as τ changes.
- Identify events.
- Between events:
  - Track all candidates for bi-level solution, and their cross validated objective.
  - Identify points where the optimum jumps between candidates.

Schematic of complete algorithm

As before, start from a solution path for τ_0.

- Identify all knots in the path for τ_0.
- Follow each knot as τ changes.
- Identify events.
- Between events:
  - Track all candidates for bi-level solution, and their cross validated objective.
  - Identify points where the optimum jumps between candidates.

Approximate computational complexity: between $O(n^2 max(n, N))$ and $O(n^3 max(n, N))$

- Based on various assumptions and approximations.
- Depends on number of support observations at non-differentiability of loss.

Schematic of complete algorithm

As before, start from a solution path for τ_0.

- Identify all knots in the path for τ_0.
- Follow each knot as τ changes.
- Identify events.
- Between events:
  - Track all candidates for bi-level solution, and their cross validated objective.
  - Identify points where the optimum jumps between candidates.

Approximate computational complexity: between $O(n^2 max(n, N))$ and $O(n^3 max(n, N))$

- Based on various assumptions and approximations.
- Depends on number of support observations at non-differentiability of loss.

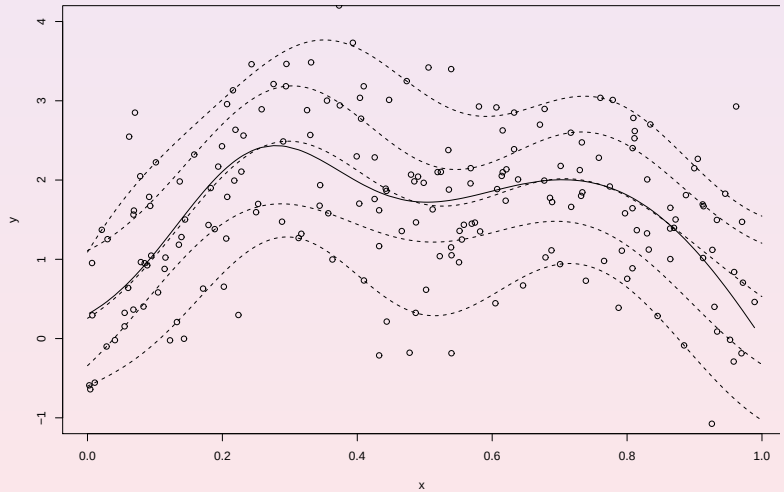
Experiments show good computational performance relative to alternatives.

Example 1: parallel quantiles

- Take $x \in [0, 1]$ uniform.
- Let $f(x) = 2 \cdot \left(\exp(-30 \cdot (x - 0.25)^2) + \sin(\pi \cdot x^2)\right)$.
- Let $Y = f(x) + \epsilon$, with $\epsilon \sim N(0, 1)$ i.i.d.

Example 1: parallel quantiles

- Take $x \in [0, 1]$ uniform.
- Let $f(x) = 2 \cdot \left(\exp(-30 \cdot (x - 0.25)^2) + \sin(\pi \cdot x^2)\right)$.
- Let $Y = f(x) + \epsilon$, with $\epsilon \sim N(0, 1)$ i.i.d.



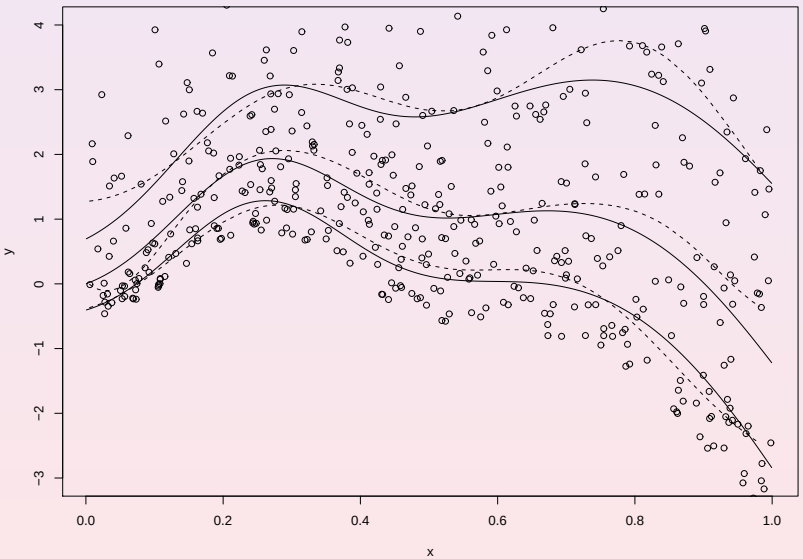
The plot shows the true mean/median $f(x)$ (solid) and the bi-level solutions for quantiles $0.1, 0.25, 0.5, 0.75, 0.9$.

Example 2: non-parallel quantiles

- Same $x, f(x)$.
- Now let $Y = f(x) + \epsilon$, with $\epsilon + (x+1)^2 \sim \exp(1/(x+1)^2)$.
- Errors are highly skewed and non-homogeneous.

Example 2: non-parallel quantiles

- Same $x, f(x)$.
- Now let $Y = f(x) + \epsilon$, with $\epsilon + (x+1)^2 \sim \exp(1/(x+1)^2)$.
- Errors are highly skewed and non-homogeneous.



The plot shows the true (solid) and the bi-level solutions (dashed) for quantiles $0.25, 0.5, 0.75$.

Summary

Our work brings together traditional and emerging concepts in (Statistical) Machine Learning:

- Regularized optimization for predictive modeling, with kernel machines as an example.
- The importance of model selection.
- Path following for efficient solution.
- Quantile estimation as a flexible approach to regression modeling.

Summary

Our work brings together traditional and emerging concepts in (Statistical) Machine Learning:

- Regularized optimization for predictive modeling, with kernel machines as an example.
- The importance of model selection.
- Path following for efficient solution.
- Quantile estimation as a flexible approach to regression modeling.

Main contributions:

- Mapping the whole solution surface $\hat{f}(\tau, \lambda)$.
- Path following algorithm for the bi-level solution $\lambda^*(\tau)$.

Thanks!

Saharon@post.tau.ac.il