Review of generalized linear point-process models

Neurostats club 2015

September 21, 2015

This will be quick
No formal derivation
No proofs

Truccolo et al. 2005

- Derivation of point process GLM
- Intrinsic and ensemble history models
- Maximum likelihood estimation
- Non-GLM conditional intensity model
- KS test and time rescaling
- Residual analysis
- Model selection
- Decoding

Truccolo et al. 2005

- Derivation of point process GLM
- **Intrinsic and ensemble history models**
- **Maximum likelihood estimation**
- Non-GLM conditional intensity model
- KS test and time rescaling
- Residual analysis
- Model selection
- Decoding

Truccolo et al. 2005

- Derivation of point process GLM
- **Intrinsic and ensemble history models**
- **Maximum likelihood estimation**
- Non-GLM conditional intensity model
- KS test and time rescaling
- Residual analysis
- Model selection
- Decoding
- **Take NEUR2110 with Wilson Truccolo in the spring!**

Truccolo et al. 2005

- Derivation of point process GLM
- **Intrinsic and ensemble history models**
- **Maximum likelihood estimation**
- Non-GLM conditional intensity model
- KS test and time rescaling
- Residual analysis
- Model selection
- Decoding
- **Take NEUR2110 with Wilson Truccolo in the spring!**

If there is time:

- Closed form approximation, double crossvalidation, regularization, and regularization paths

Linear model / multiple linear regression

$$y = \beta_o + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + ... = \beta_o + \sum_{i=1}^{N} x_i \beta_i = XB + \beta_o$$

- y: response/dependent variable being modeled (when multivariate, often a column vector by convention)
- X: "design matrix" matrix of observations. Conventionally, each row is a realization and each column is a feature/variable
- B: Coefficient or parameter vector
- β_o: Constant term.

Nonlinear features OK

- Features $(x_1, x_2, ...)$ can be any function of recorded data. E.g. the following model is common for phase/direction tuning (used in eqn. 10 in Truccolo et al. 2005)

$$y = \beta_o + A \cdot cos(\varphi - \varphi_o)$$

Nonlinear features OK

- Features $(x_1, x_2, ...)$ can be any function of recorded data. E.g. the following model is common for phase/direction tuning (used in eqn. 10 in Truccolo et al. 2005)

$$y = \beta_o + A \cdot cos(\varphi - \varphi_o)$$

- A: amplitude parameter

Nonlinear features OK

- Features $(x_1, x_2, ...)$ can be any function of recorded data. E.g. the following model is common for phase/direction tuning (used in eqn. 10 in Truccolo et al. 2005)

$$y = \beta_o + A \cdot cos(\varphi - \varphi_o)$$

- A: amplitude parameter
- φ: the observed phase or direction

Nonlinear features OK

- Features $(x_1, x_2, ...)$ can be any function of recorded data. E.g. the following model is common for phase/direction tuning (used in eqn. 10 in Truccolo et al. 2005)

$$y = \beta_o + A \cdot cos(\varphi - \varphi_o)$$

- A: amplitude parameter
- φ: the observed phase or direction
- φ_o: preferred phase parameter

Nonlinear features OK

- Features $(x_1, x_2, ...)$ can be any function of recorded data. E.g. the following model is common for phase/direction tuning (used in eqn. 10 in Truccolo et al. 2005)

$$y = \beta_o + A \cdot cos(\varphi - \varphi_o)$$

- A: amplitude parameter
- φ: the observed phase or direction
- φ_o: preferred phase parameter
- Can be written in a form that is linear in parameters

$$y = \beta_o + Acos(\varphi_o)cos(\varphi) + Asin(\varphi_o)sin(\varphi)$$

Nonlinear features OK

- Features $(x_1, x_2, ...)$ can be any function of recorded data. E.g. the following model is common for phase/direction tuning (used in eqn. 10 in Truccolo et al. 2005)

$$y = \beta_o + A \cdot cos(\varphi - \varphi_o)$$

- A: amplitude parameter
- φ: the observed phase or direction
- φ_o: preferred phase parameter
- Can be written in a form that is linear in parameters

$$y = \beta_o + \beta_1 cos(\varphi) + \beta_2 sin(\varphi)$$

Point process model:

- Truccolo et al. 2005:
  - "neural spike trains form a sequence of discrete events or point process time series"

Point process model:

- Truccolo et al. 2005:
  - "neural spike trains form a sequence of discrete events or point process time series"
  - "standard linear or nonlinear regression methods are designed for analysis of continuous-valued data and not point process observations"

Point process model:

- Truccolo et al. 2005:
  - “neural spike trains form a sequence of discrete events or point process time series”
  - “standard linear or nonlinear regression methods are designed for analysis of continuous-valued data and not point process observations”
- Smoothing or binning can alter the structure

Point process model:

- Truccolo et al. 2005:
  - "neural spike trains form a sequence of discrete events or point process time series"
  - "standard linear or nonlinear regression methods are designed for analysis of continuous-valued data and not point process observations"
- Smoothing or binning can alter the structure
- GLM Point-process models directly model spike trains without these drawbacks

GLM point process models

- Consider a homogeneous Poisson process with rate λ.
  - expected # observations in time window Δ is $\lambda\Delta$

GLM point process models

- Consider a homogeneous Poisson process with rate λ.
  - expected # observations in time window Δ is $\lambda\Delta$
- For inhomogeneous Poisson process rate varies time $\lambda(t)$
  - # observations from t to $t + \Delta$ is $\int_t^{t+\Delta} \lambda(t)dt$

GLM point process models

- Consider a homogeneous Poisson process with rate λ.
  - expected # observations in time window Δ is $\lambda\Delta$
- For inhomogeneous Poisson process rate varies time $\lambda(t)$
  - # observations from t to $t+\Delta$ is $\int_t^{t+\Delta} \lambda(t)dt$
- $\lambda(t)$ is called the "intensity function"

GLM point process models

- Consider a homogeneous Poisson process with rate λ.
  - expected # observations in time window Δ is $\lambda\Delta$
- For inhomogeneous Poisson process rate varies time $\lambda(t)$
  - # observations from t to $t + \Delta$ is $\int_t^{t+\Delta} \lambda(t)dt$
- $\lambda(t)$ is called the "intensity function"
- In a point process model we predict **conditional intensity** based on measured covariates $\lambda(t|X(t))$

GLM point process models

- Consider a homogeneous Poisson process with rate λ.
  - expected # observations in time window Δ is $\lambda\Delta$
- For inhomogeneous Poisson process rate varies time $\lambda(t)$
  - # observations from t to $t+\Delta$ is $\int_t^{t+\Delta} \lambda(t)dt$
- $\lambda(t)$ is called the "intensity function"
- In a point process model we predict **conditional intensity** based on measured covariates $\lambda(t|X(t))$
- The Poisson GLM point process framework models conditional intensity functions of the form

$$\lambda(t|X(t)) = \exp\left\{\mu + XB\right\}$$

Fitting GLM point-process models

- In practice spiking time series are discretized at some resolution Δ

Fitting GLM point-process models

- In practice spiking time series are discretized at some resolution Δ
  - Predict sequence of spike counts $y_t = (y_1, y_2, ...)$

Fitting GLM point-process models

- In practice spiking time series are discretized at some resolution Δ
  - Predict sequence of spike counts $y_t = (y_1, y_2, ...)$
  - With a discrete approximation of the conditional intensity $\lambda_t = (\lambda_1, \lambda_2, ...)$

Fitting GLM point-process models

- In practice spiking time series are discretized at some resolution Δ
  - Predict sequence of spike counts $y_t = (y_1, y_2, ...)$
  - With a discrete approximation of the conditional intensity $\lambda_t = (\lambda_1, \lambda_2, ...)$
  - Based on discretely sampled covariates $X_t = (x_1, x_2, ...)$

Fitting GLM point-process models

- In practice spiking time series are discretized at some resolution Δ
  - Predict sequence of spike counts $y_t = (y_1, y_2, ...)$
  - With a discrete approximation of the conditional intensity $\lambda_t = (\lambda_1, \lambda_2, ...)$
  - Based on discretely sampled covariates $X_t = (x_1, x_2, ...)$
- Typically Δ chosen such that $y_t \in \{0, 1\}$, (larger bins can be used, making it a count process)

Fitting GLM point-process models

- In practice spiking time series are discretized at some resolution Δ
  - Predict sequence of spike counts $y_t = (y_1, y_2, ...)$
  - With a discrete approximation of the conditional intensity $\lambda_t = (\lambda_1, \lambda_2, ...)$
  - Based on discretely sampled covariates $X_t = (x_1, x_2, ...)$
- Typically Δ chosen such that $y_t \in \{0, 1\}$, (larger bins can be used, making it a count process)
- Model estimated by finding parameters B that maximize the likelihood of the observed spikes y and covariates X

Fitting GLM point-process models

- In practice spiking time series are discretized at some resolution Δ
  - Predict sequence of spike counts $y_t = (y_1, y_2, ...)$
  - With a discrete approximation of the conditional intensity $\lambda_t = (\lambda_1, \lambda_2, ...)$
  - Based on discretely sampled covariates $X_t = (x_1, x_2, ...)$
- Typically Δ chosen such that $y_t \in \{0, 1\}$, (larger bins can be used, making it a count process)
- Model estimated by finding parameters B that maximize the likelihood of the observed spikes y and covariates X
  - Can be fit with iteratively reweighted least squares

Fitting GLM point-process models

- In practice spiking time series are discretized at some resolution Δ
  - Predict sequence of spike counts $y_t = (y_1, y_2, ...)$
  - With a discrete approximation of the conditional intensity $\lambda_t = (\lambda_1, \lambda_2, ...)$
  - Based on discretely sampled covariates $X_t = (x_1, x_2, ...)$
- Typically Δ chosen such that $y_t \in \{0, 1\}$, (larger bins can be used, making it a count process)
- Model estimated by finding parameters B that maximize the likelihood of the observed spikes y and covariates X
  - Can be fit with iteratively reweighted least squares
  - MATLAB glmfit

    mathworks.com/help/stats/glmfit.htmlt

Fitting GLM point-process models

- In practice spiking time series are discretized at some resolution Δ
  - Predict sequence of spike counts $y_t = (y_1, y_2, ...)$
  - With a discrete approximation of the conditional intensity $\lambda_t = (\lambda_1, \lambda_2, ...)$
  - Based on discretely sampled covariates $X_t = (x_1, x_2, ...)$
- Typically Δ chosen such that $y_t \in \{0, 1\}$, (larger bins can be used, making it a count process)
- Model estimated by finding parameters B that maximize the likelihood of the observed spikes y and covariates X
  - Can be fit with iteratively reweighted least squares
  - MATLAB glmfit
    mathworks.com/help/stats/glmfit.htmlt
  - Python scikits.statsmodels.GLM
    statsmodels.sourceforge.net/stable/glm.html

Fitting GLM point-process models

- In practice spiking time series are discretized at some resolution Δ
    - Predict sequence of spike counts $y_t = (y_1, y_2, ...)$
    - With a discrete approximation of the conditional intensity $\lambda_t = (\lambda_1, \lambda_2, ...)$
    - Based on discretely sampled covariates $X_t = (x_1, x_2, ...)$
- Typically Δ chosen such that $y_t \in \{0, 1\}$, (larger bins can be used, making it a count process)
- Model estimated by finding parameters B that maximize the likelihood of the observed spikes y and covariates X
    - Can be fit with iteratively reweighted least squares
    - MATLAB glmfit
      mathworks.com/help/stats/glmfit.htmlt
    - Python scikits.statsmodels.GLM
      statsmodels.sourceforge.net/stable/glm.html
    - I typically use gradient descent

GLM point process model for single unit spiking with ensemble history

$$\log[\lambda(t|X(t))\Delta] = \mu$$

GLM point process model for single unit spiking with ensemble history

$$\log[\lambda(t|X(t))\Delta] = \mu$$
$$+ \textbf{intrinsic history}$$

GLM point process model for single unit spiking with ensemble history

$$\log[\lambda(t|X(t))\Delta] = \mu + \textbf{intrinsic history} + \textbf{ensemble history}$$

GLM point process model for single unit spiking with ensemble history

$$\log[\lambda(t|X(t))\Delta] = \mu$$
$$+ \textbf{intrinsic history}$$
$$+ \textbf{ensemble history}$$
$$+ \textbf{extrinsic covariates}$$

Intrinsic history filter

- Theoretically, model the whole history

$$\lambda_t(t|\theta, y_{t-1}, y_{t-2}, ..., y_1) \propto \exp\left\{\sum_{\tau=1}^{t} \gamma_\tau y_{t-\tau}\right\}$$

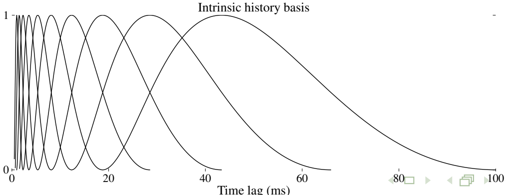
Intrinsic history filter

- Theoretically, model the whole history

$$\lambda_t(t|\theta, y_{t-1}, y_{t-2}, ..., y_1) \propto \exp\left\{\sum_{\tau=1}^{t} \gamma_\tau y_{t-\tau}\right\}$$

- In practice: use finite history duration with basis functions (a form of regularization: enforce smoothness and reduce parameters)

$$\lambda_t(t|\theta, y_{t-\tau:t-1}) \propto \exp\left\{B \cdot y_{t-\tau:t-1}\right\}$$



Ensemble history filter

- Ensemble history filters are treated similarly to intrinsic history

Ensemble history filter

- Ensemble history filters are treated similarly to intrinsic history
- Typically use fewer basis functions than for the intrinsic history, otherwise the number of parameters becomes prohibitive

Ensemble history filter

- Ensemble history filters are treated similarly to intrinsic history
- Typically use fewer basis functions than for the intrinsic history, otherwise the number of parameters becomes prohibitive
- If using regularization, typically each neuron's parameters are penalized as a group (sparse connectivity prior)

Ensemble history filter

- Ensemble history filters are treated similarly to intrinsic history
- Typically use fewer basis functions than for the intrinsic history, otherwise the number of parameters becomes prohibitive
- If using regularization, typically each neuron's parameters are penalized as a group (sparse connectivity prior)
- More on intrinsic history and network connectivity next week

Extrinsic covariates: kinematics

- Truccolo et al. 2005 explore a 2D velocity tuning model based on the x and y components of hand velocity.

Extrinsic covariates: kinematics

- Truccolo et al. 2005 explore a 2D velocity tuning model based on the x and y components of hand velocity.
- Hatsopoulos et al. 2007 use normalized, extended velocity trajectories "pathlets"

Maximum likelihood approach to model fitting

- Let y_t be an inhomogeneous Poisson with time varying rate λ_t

Maximum likelihood approach to model fitting

- Let y_t be an inhomogeneous Poisson with time varying rate λ_t
- The probability of observing k spikes in a time interval Δ is Poisson distributed

$$\Pr(y_t = k) \approx (\Delta\lambda_t)^{y_t} \frac{e^{-\Delta\lambda_t}}{y_t!}$$

Maximum likelihood approach to model fitting

- Let y_t be an inhomogeneous Poisson with time varying rate λ_t
- The probability of observing k spikes in a time interval Δ is Poisson distributed

$$\Pr(y_t = k) \approx (\Delta\lambda_t)^{y_t} \frac{e^{-\Delta\lambda_t}}{y_t!}$$

- Assuming conditional independence, the probability of observing an entire sequence y_t is

$$\Pr(y|\lambda) = \prod_{t=1}^{T} (\Delta\lambda_t)^{y_t} e^{-\Delta\lambda_t} / y_t!$$

Minimize the negative log-likelihood

- Fit the model by finding the parameters μ, B that **maximize the likelihood** $\mathcal{L}(\mu, B|y) = \Pr(y|\mu, B)$ of the observations[1]

$$\Pr(y|\mu, B) = \prod_{t=1}^{T} \lambda_t^{y_t} e^{-\lambda_t} / y_t!$$

$$\lambda_t = \exp(\mu + X_t B)$$

[1] **Note: writing λ_t here instead of $\Delta\lambda_t$, i.e. let $\Delta = 1$. In this case, parameters μ and B will take units of Δ**

Minimize the negative log-likelihood

- Fit the model by finding the parameters μ, B that **maximize the likelihood** $\mathcal{L}(\mu, B|y) = \Pr(y|\mu, B)$ of the observations[1]

$$\Pr(y|\mu, B) = \prod_{t=1}^{T} \lambda_t^{y_t} e^{-\lambda_t} / y_t!$$

$$\lambda_t = \exp(\mu + X_t B)$$

- In practice, **minimize the negative log-likelihood**, which, if Δ is small $s.t.$ y_t is always 0 or 1

$$-\ln \mathcal{L}(\mu, B|y) = \sum_{t=1}^{T} [\lambda_t - y_t \ln(\lambda_t)]$$

---

[1] **Note: writing λ_t here instead of $\Delta\lambda_t$, i.e. let $\Delta = 1$. In this case, parameters μ and B will take units of Δ**

Gradient of the negative log-likelihood

- Let $f(\mu, B)$ be the negative log likelihood

$$f(\mu, B) = -\ln \mathcal{L}(\mu, B|y) = \sum_{t=1}^{T} [\lambda_t - y_t \ln(\lambda_t)]$$

Gradient of the negative log-likelihood

- Let $f(\mu, B)$ be the negative log likelihood

$$f(\mu, B) = -\ln \mathcal{L}(\mu, B|y) = \sum_{t=1}^{T} [\lambda_t - y_t \ln(\lambda_t)]$$

- Substitute our model $\lambda_t = e^{\mu + X_t B}$

$$f(\mu, B) = \sum_{t=1}^{T} [e^{\mu + X_t B} - y_t(\mu + X_t B)]$$

Gradient of the negative log-likelihood

- Let $f(\mu, B)$ be the negative log likelihood

$$f(\mu, B) = -\ln \mathcal{L}(\mu, B|y) = \sum_{t=1}^{T} [\lambda_t - y_t \ln(\lambda_t)]$$

- Substitute our model $\lambda_t = e^{\mu + X_t B}$

$$f(\mu, B) = \sum_{t=1}^{T} [e^{\mu + X_t B} - y_t(\mu + X_t B)]$$

- The partial derivatives, w.r.t μ and $(\beta_1, \beta_2, ...) = B$ are:

$$\frac{\partial f}{\partial \mu} = \sum_{t=1}^{T} [e^{\mu + X_t B} - y_t]$$

$$\frac{\partial f}{\partial \beta_i} = \sum_{t=1}^{T} [X_{t,i} e^{\mu + X_t B} - y_t X_{t,i}]$$

Regularized GLM

- Cross-validation is necessary to assess over-fitting:
  - data should be separated into training and testing sets

Regularized GLM

- Cross-validation is necessary to assess over-fitting:
  - data should be separated into training and testing sets
- For larger number of parameters, the model **will** overfit:
  - regularization is necessary.

Regularized GLM

- Cross-validation is necessary to assess over-fitting:
  - data should be separated into training and testing sets
- For larger number of parameters, the model **will** overfit:
  - regularization is necessary.
- Regularization can be incorporated by adding a penalty term to the negative log-likelihood function

$$\underset{\mu,B}{\operatorname{argmin}} \{\mathbf{Penalty}(B) - \ln \mathcal{L}(\mu, B|y)\}$$

Regularized GLM

- Cross-validation is necessary to assess over-fitting:
  - data should be separated into training and testing sets
- For larger number of parameters, the model **will** overfit:
  - regularization is necessary.
- Regularization can be incorporated by adding a penalty term to the negative log-likelihood function

$$\underset{\mu,B}{\operatorname{argmin}} \{\textbf{Penalty}(B) - \ln \mathcal{L}(\mu, B|y)\}$$

- Conjugate gradient solvers are useful here

Regularized GLM: L1 and L2

- L2 penalty: Parameters penalized by their squared magnitudes

$$\alpha \sum_{i=1}^{N} \beta_i^2$$

Regularized GLM: L1 and L2

- L2 penalty: Parameters penalized by their squared magnitudes

$$\alpha \sum_{i=1}^{N} \beta_i^2$$

  - Equivalent to a Gaussian prior on parameters

Regularized GLM: L1 and L2

- L2 penalty: Parameters penalized by their squared magnitudes

$$\alpha \sum_{i=1}^{N} \beta_i^2$$

  - Equivalent to a Gaussian prior on parameters
  - Can be solved with gradient descent

Regularized GLM: L1 and L2

- L2 penalty: Parameters penalized by their squared magnitudes

$$\alpha \sum_{i=1}^{N} \beta_i^2$$

  - Equivalent to a Gaussian prior on parameters
  - Can be solved with gradient descent
- L1 penalty: Parameters penalized by their absolute magnitude

$$\alpha \sum_{i=1}^{N} |\beta_i|$$

Regularized GLM: L1 and L2

- L2 penalty: Parameters penalized by their squared magnitudes

$$\alpha \sum_{i=1}^{N} \beta_i^2$$

  - Equivalent to a Gaussian prior on parameters
  - Can be solved with gradient descent

- L1 penalty: Parameters penalized by their absolute magnitude

$$\alpha \sum_{i=1}^{N} |\beta_i|$$

  - Promotes $\beta_i = 0$, useful for finding sparse solutions

Regularized GLM: L1 and L2

- L2 penalty: Parameters penalized by their squared magnitudes

$$\alpha \sum_{i=1}^{N} \beta_i^2$$

  - Equivalent to a Gaussian prior on parameters
  - Can be solved with gradient descent
- L1 penalty: Parameters penalized by their absolute magnitude

$$\alpha \sum_{i=1}^{N} |\beta_i|$$

  - Promotes $\beta_i = 0$, useful for finding sparse solutions
  - Discontinuous gradient at $\beta_i = 0$ percludes gradient descent.

Regularized GLM: L1 and L2

- L2 penalty: Parameters penalized by their squared magnitudes

$$\alpha \sum_{i=1}^{N} \beta_i^2$$

  - Equivalent to a Gaussian prior on parameters
  - Can be solved with gradient descent
- L1 penalty: Parameters penalized by their absolute magnitude

$$\alpha \sum_{i=1}^{N} |\beta_i|$$

  - Promotes $\beta_i = 0$, useful for finding sparse solutions
  - Discontinuous gradient at $\beta_i = 0$ percludes gradient descent.
  - Can use coordinate descent (although we ran into convergence issues?)

Regularized GLM: L1 approximation and L0

- $\sqrt{x^2 + \epsilon}$: Smooth approximation of L1 penalty that is suitable for gradient descent[2]

$$\alpha \sum_{i=1}^{N} \sqrt{\beta_i^2 + \varepsilon}$$

[2] Called "Charbonnier penalty" in the computer vision literature (Charbonnier et al. 1994)

Regularized GLM: L1 approximation and L0

- $\sqrt{x^2+\epsilon}$: Smooth approximation of L1 penalty that is suitable for gradient descent[2]

$$\alpha \sum_{i=1}^{N} \sqrt{\beta_i^2 + \varepsilon}$$

  - ϵ is chosen to be small, but strictly positive

[2] Called "Charbonnier penalty" in the computer vision literature (Charbonnier et al. 1994)

Regularized GLM: L1 approximation and L0

- $\sqrt{x^2+\epsilon}$: Smooth approximation of L1 penalty that is suitable for gradient descent[2]

$$\alpha \sum_{i=1}^{N} \sqrt{\beta_i^2 + \varepsilon}$$

  - ϵ is chosen to be small, but strictly positive
- L0 penalty: Constant penalty if a parameter is nonzero

$$\alpha \sum_{i=1}^{N} \delta(\beta_i \neq 0)$$

[2] Called "Charbonnier penalty" in the computer vision literature (Charbonnier et al. 1994)

Regularized GLM: L1 approximation and L0

- $\sqrt{x^2 + \epsilon}$: Smooth approximation of L1 penalty that is suitable for gradient descent[2]

$$\alpha \sum_{i=1}^{N} \sqrt{\beta_i^2 + \varepsilon}$$

  - ϵ is chosen to be small, but strictly positive
- L0 penalty: Constant penalty if a parameter is nonzero

$$\alpha \sum_{i=1}^{N} \delta(\beta_i \neq 0)$$

  - Computationally infeasible

[2] Called "Charbonnier penalty" in the computer vision literature (Charbonnier et al. 1994)

Regularized GLM: L1 approximation and L0

- $\sqrt{x^2 + \epsilon}$: Smooth approximation of L1 penalty that is suitable for gradient descent[2]

$$\alpha \sum_{i=1}^{N} \sqrt{\beta_i^2 + \varepsilon}$$

  - ϵ is chosen to be small, but strictly positive
- L0 penalty: Constant penalty if a parameter is nonzero

$$\alpha \sum_{i=1}^{N} \delta(\beta_i \neq 0)$$

  - Computationally infeasible
  - Greedy algorithms are a good (the best polynomial time?) approximation

[2] Called "Charbonnier penalty" in the computer vision literature (Charbonnier et al. 1994)

Regularized GLM: Group lasso

- Concept: penalize **groups** of parameters with the L1 norm

$$\alpha \sum_{i=1}^{N} \sqrt{\sum_{j=1}^{K} \beta_{i,j}^2}$$

[3] Roth, Volker, and Bernd Fischer. "The group-lasso for generalized linear models: uniqueness of solutions and efficient algorithms." Proceedings of the 25th international conference on Machine learning. ACM, 2008.

Regularized GLM: Group lasso

- Concept: penalize **groups** of parameters with the L1 norm

$$\alpha \sum_{i=1}^{N} \sqrt{\sum_{j=1}^{K} \beta_{i,j}^2}$$

  - Useful for penalizing ensemble filters as a group for each neuron

[3] Roth, Volker, and Bernd Fischer. "The group-lasso for generalized linear models: uniqueness of solutions and efficient algorithms." Proceedings of the 25th international conference on Machine learning. ACM, 2008.

Regularized GLM: Group lasso

- Concept: penalize **groups** of parameters with the L1 norm

$$\alpha \sum_{i=1}^{N} \sqrt{\sum_{j=1}^{K} \beta_{i,j}^2}$$

  - Useful for penalizing ensemble filters as a group for each neuron
  - Derivative is undefined at $\sqrt{\sum_{j=1}^{K} \beta_{i,j}^2} = 0$

[3] Roth, Volker, and Bernd Fischer. "The group-lasso for generalized linear models: uniqueness of solutions and efficient algorithms." Proceedings of the 25th international conference on Machine learning. ACM, 2008.

Regularized GLM: Group lasso

- Concept: penalize groups of parameters with the L1 norm

$$\alpha \sum_{i=1}^{N} \sqrt{\sum_{j=1}^{K} \beta_{i,j}^2}$$

  - Useful for penalizing ensemble filters as a group for each neuron
  - Derivative is undefined at $\sqrt{\sum_{j=1}^{K} \beta_{i,j}^2} = 0$
  - Roth & Fischer 2008[3] discuss an efficient fitting procedure

[3] Roth, Volker, and Bernd Fischer. "The group-lasso for generalized linear models: uniqueness of solutions and efficient algorithms." Proceedings of the 25th international conference on Machine learning. ACM, 2008.

Two-layer crossvalidation

- The strength of regularization α is another free parameter
  - Can be fixed in advance or...

Two-layer crossvalidation

- The strength of regularization α is another free parameter
  - Can be fixed in advance or...
  - Two-level crossvalidation can be used to estimate α from the data

Two-layer crossvalidation

- The strength of regularization α is another free parameter
  - Can be fixed in advance or...
  - Two-level crossvalidation can be used to estimate α from the data
- K-fold two-layer crossvalidation results in K^2 parameter fitting steps
  - This can get slow on large datasets

Regularization paths

- Regularization paths:
  - When evaluating a range of regularization parameters e.g. $\alpha = (0, 0.1, 1, 20)$,
  - Carry over the model weights μ and β.
  - That is, for example, start the parameter tuning for $\alpha = 1$ with the μ, B returned from $\alpha = 0.1$

Initialization with closed form solution

- Fitting many models using a large amount of data can take a very long time

Initialization with closed form solution

- Fitting many models using a large amount of data can take a very long time
- Using an approximation to estimate a good starting location can give some speedup
- Ramirez, A. D., & Paninski, L. (2014). Fast inference in generalized linear models via expected log-likelihoods. Journal of computational neuroscience, 36(2), 215-234.

Truccolo et al. 2005

$$\lambda(t \mid H(t)) = \lim_{\Delta \to 0} \frac{P[N(t + \Delta) - N(t) = 1 \mid H(t)]}{\Delta} \qquad (1)$$

$$P(N_{1:K} \mid \theta) = \prod_{k=1}^{K} [\lambda(t_k \mid \theta, H_k)\Delta]^{\Delta N_k} [1 - \lambda(t_k \mid \theta, H_k)\Delta]^{1-\Delta N_k} + o(\Delta^J) \quad (2)$$

$$P(N_{1:K} \mid \theta) = \exp\left\{ \sum_{k=1}^{K} \log\left[\lambda(t_{\mathrm{k}} \mid \theta, H_k)\Delta\right]\Delta N_k - \sum_{k=1}^{K} \lambda(t_k \mid \theta, H_k)\Delta \right\} + o(\Delta^J) \quad (3)$$

$$\log \lambda(t_k \mid \theta, H_k) = \sum_{i=1}^{q} \theta_i g_i[v_i(t_k + \tau)] \tag{4}$$

$$\log\{[1-\lambda(t_k \mid \theta, H_k)\Delta]^{-1}[\lambda(t_k \mid \theta, H_k)\Delta]\} = \sum_{i=1}^{q} \theta_i g_i[v_i(t_k+\tau)] \qquad (5)$$

$$\lambda(t_k \mid N_{1:k}^{1:C}, \mathbf{x}_{k+\tau}, \theta) = \lambda_I(t_k \mid N_{1:k}, \theta_I)\lambda_E(t_k \mid N_{1:k}^{1:C}, \theta_E)\lambda_X(t_k \mid \mathbf{x}_{k+\tau}, \theta_X) \qquad (6)$$

$$\lambda_I(t_k \mid N_{1:k}, \theta_I) = \exp\left\{ \gamma_0 + \sum_{n=1}^{Q} \gamma_n \Delta N_{k-n} \right\} \qquad (7)$$

$$\lambda_E(t_k \mid N^{1:C}_{1:k}, \theta_E) = \exp\left\{\beta_0 + \sum_c \sum_{r=1}^{R} \beta^c_r \Delta N^c_{k-r}\right\} \tag{8}$$

$$\lambda_E(t_k \mid N_{1:k}^{1:C}, \theta_E) = \exp\left\{ \beta_0 + \sum_c \sum_{r=1}^{R} \beta_r^c \left(N_{k-(r-1)W}^{c} - N_{k-rW}^{c}\right) \right\} \qquad (9)$$

$$\lambda_X(t_k \mid \mathbf{x}_{k+\tau}, \theta_X) = \exp\{\alpha_0 + |V_{k+\tau}| \left[\alpha_1 \cos(\phi_{k+\tau}) + \alpha_2 \sin(\phi_{k+\tau})\right]\} \quad (10)$$

$$\lambda(t_k \mid N_{1:k}^{1:C}, \mathbf{x}_{k+\tau}, \theta) = \exp\{\mu + \sum_{n=1}^{Q} \gamma_n \Delta N_{k-n}$$

$$+ \sum_{c} \sum_{r=1}^{R} \beta_r^c \Delta N_{k-r}^c + |V_{k+\tau}| \, [\alpha_1 \cos(\phi_{k+\tau}) + \alpha_2 \sin(\phi_{k+\tau})]\} \quad (11)$$

$$\lambda(t_k \mid \theta, H_k) = \frac{p(t_e \mid \theta, H_k)}{1 - \int_{u_{N_{k-1}}}^{t_k} p[t \mid \theta, H(t)]\mathrm{d}t} \tag{12}$$

$$s(t_k \mid \mathbf{x}_{t+\tau}, N_{1:k}^{1:C}, \theta_X, \theta_E) = \exp\Bigg\{\mu + \sum_{c} \sum_{r=1}^{R} \beta_r^c (N_{k-(r-1)W}^c - N_{k-rW}^c) + |V_{k+\tau}| [\alpha_1 \cos(\phi_{k+\tau}) + \alpha_2 \sin(\phi_{k+\tau})]\Bigg\} \quad (13)$$

$$z_j = 1 - \exp\left\{ -\int_{u_j}^{u_{j+1}} \lambda(t \mid H(t), \hat{\theta}) \mathrm{d}t \right\} \qquad (14)$$

$$M(t_k) = \sum_{i=k-B}^{k} \Delta N_i - \int_{t_{k-B}}^{t_k} \lambda(t \mid H(t), \hat{\theta}) \mathrm{d}t \qquad (15)$$

$$AIC(q) = -2 \log L(\hat{\theta} \mid H_k) + 2q \tag{16}$$

$$\mathbf{x}_{k+\tau} = \mu_{\mathbf{x}} + \mathbf{F}\mathbf{x}_{k+\tau-1} + \varepsilon_{k+\tau} \qquad (17)$$

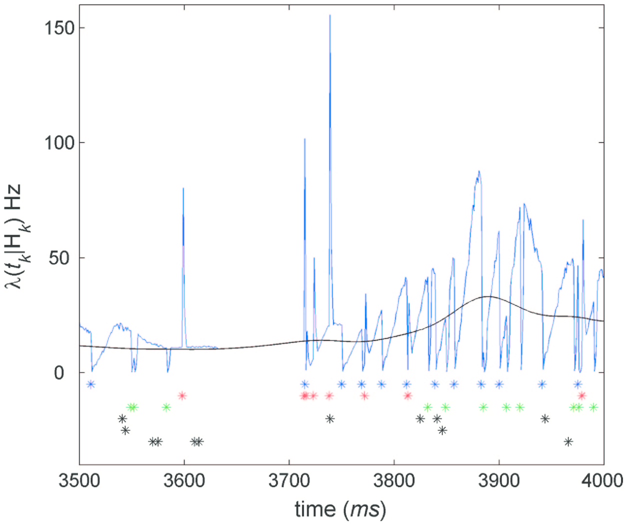
$$\mathbf{x}_{k+\tau\,|\,k+\tau-1} = \mu_{\mathbf{x}} + \mathbf{F}\mathbf{x}_{k+\tau-1\,|\,k+\tau-1} \tag{18}$$

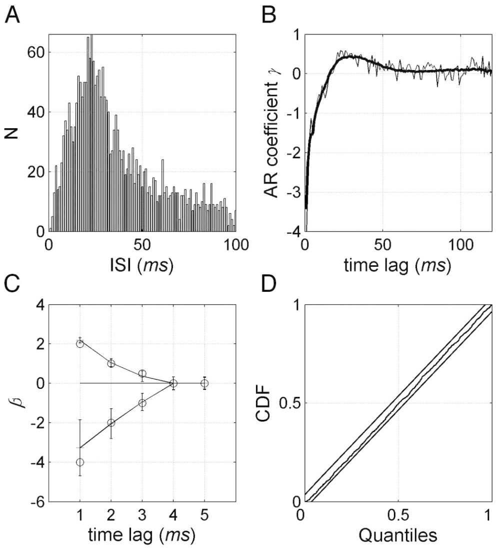
$$\mathbf{W}_{k+\tau\,|\,k+\tau-1} = \mathbf{F}\mathbf{W}_{k+\tau-1\,|\,k+\tau-1}\mathbf{F} + \mathbf{W}_{\varepsilon} \qquad (19)$$

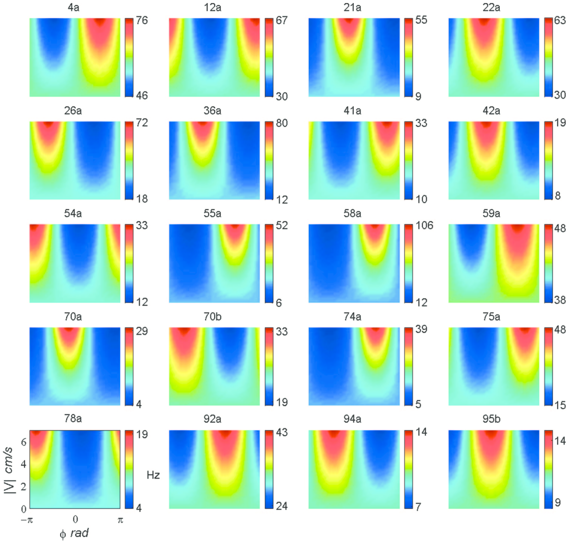
$$
\begin{aligned}
\mathbf{W}_{k+\tau\,|\,k+\tau} = \Bigg[& \mathbf{W}^{-1}_{k+\tau\,|\,k+\tau-1} + \sum_{c=1}^{C} [\nabla \log \lambda^{c} (t_k \,|\, N^{c}_{1:k}, \mathbf{x}_{k+\tau\,|\,k+\tau-1}, \hat{\theta}^{c})] \\
& \lambda^{c}(t_k \,|\, N^{c}_{1:k}, \mathbf{x}_{k+\tau\,|\,k+\tau-1}, \hat{\theta}^{c})\Delta[\nabla \log \lambda^{c} (t_k \,|\, N^{c}_{1:k}, \mathbf{x}_{k+\tau\,|\,k+\tau-1}, \hat{\theta}^{c})]' \\
& - \sum_{c=1}^{C} \nabla^{2} \log \lambda^{c} (t_k \,|\, N^{c}_{1:k}, \mathbf{x}_{k+\tau\,|\,k+\tau-1}, \hat{\theta}^{c}) \\
& [\Delta N^{c}_{1:k} - \lambda^{c}(t_k \,|\, N^{c}_{1:k}, \mathbf{x}_{k+\tau\,|\,k+\tau-1}, \hat{\theta}^{c})\Delta] \Bigg]^{-1} \qquad (20)
\end{aligned}
$$

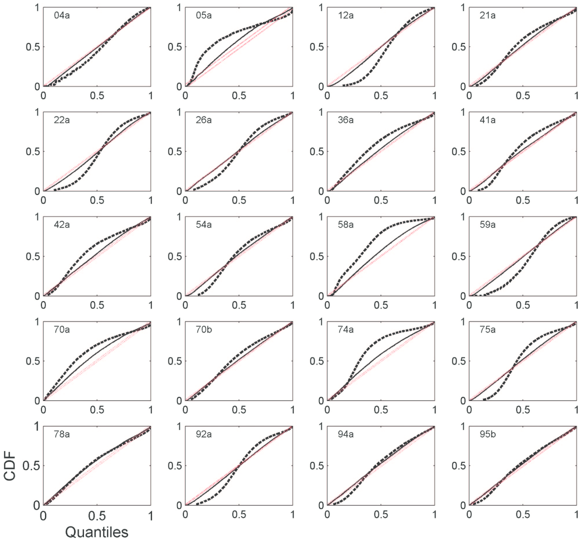
$$\mathbf{x}_{k+\tau\,|\,k+\tau} = \mathbf{x}_{k+\tau\,|\,k+\tau-1} + \mathbf{W}_{k+\tau\,|\,k+\tau}$$
$$\times \sum_{c=1}^{C} \nabla \log \lambda^c(t_k \,|\, N^c_{1:k}, \mathbf{x}_{k+\tau\,|\,k+\tau-1}, \hat{\theta}^c)\,[\Delta N^c_{1:k} - \lambda^c(t_k \,|\, N^c_{1:k}, \mathbf{x}_{k+\tau\,|\,k+\tau-1}, \hat{\theta}^c)\Delta] \tag{21}$$

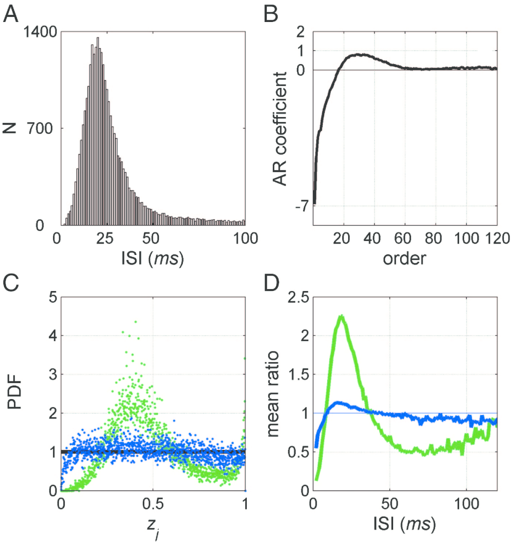
$$(\mathbf{x}_{k+\tau} - \mathbf{x}_{k+\tau\,|\,k+\tau})' \mathbf{W}^{-1}_{k+\tau\,|\,k+\tau} (\mathbf{x}_{k+\tau} - \mathbf{x}_{k+\tau\,|\,k+\tau}) \leq \chi^2_{0.95}(m) \qquad (22)$$

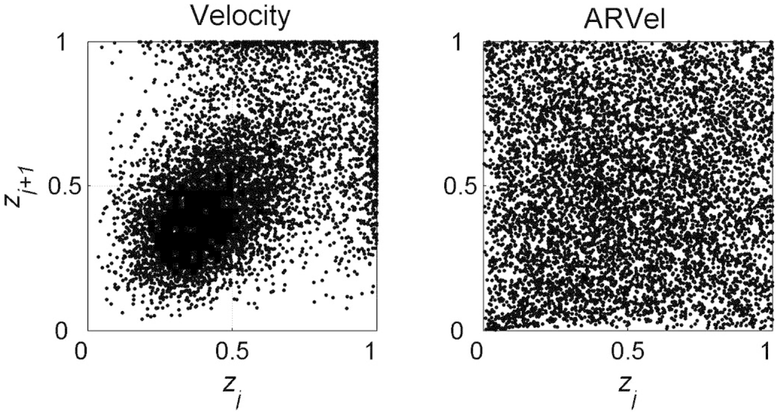

150
100
50
0
$\lambda(t_k|H_k)$ Hz
3500
3600
3700
3800
3900
4000
time (*ms*)

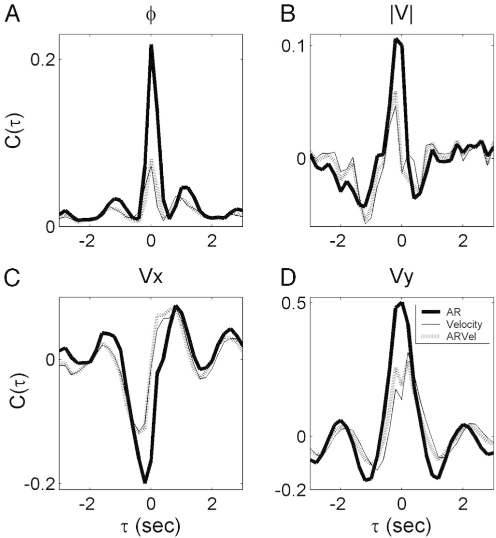

A
N
ISI (ms)
B
AR coefficient γ
time lag (ms)
C
β
time lag (ms)
D
CDF
Quantiles

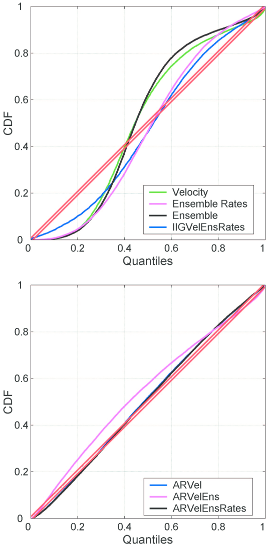

4a
76
46
12a
67
30
21a
55
9
22a
63
30
26a
72
18
36a
80
12
41a
33
10
42a
19
8
54a
33
12
55a
52
6
58a
106
12
59a
48
38
70a
29
4
70b
33
19
74a
39
5
75a
48
15
78a
19
4
92a
43
24
94a
14
7
95b
14
9
|V| cm/s
6
4
2
0
−π
0
π
ϕ rad
Hz

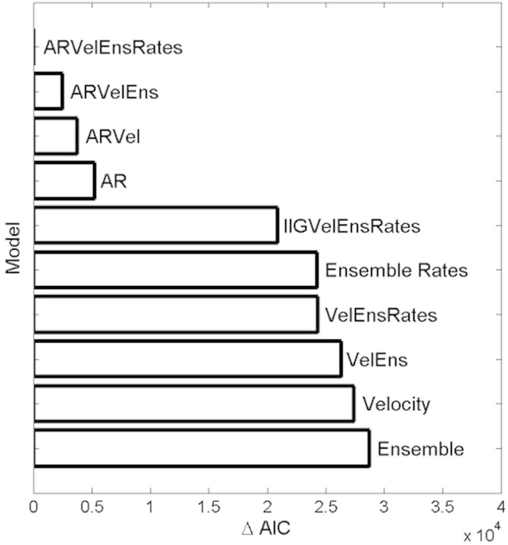

04a
05a
12a
21a
22a
26a
36a
41a
42a
54a
58a
59a
70a
70b
74a
75a
78a
92a
94a
95b
CDF
Quantiles
0
0.5
1

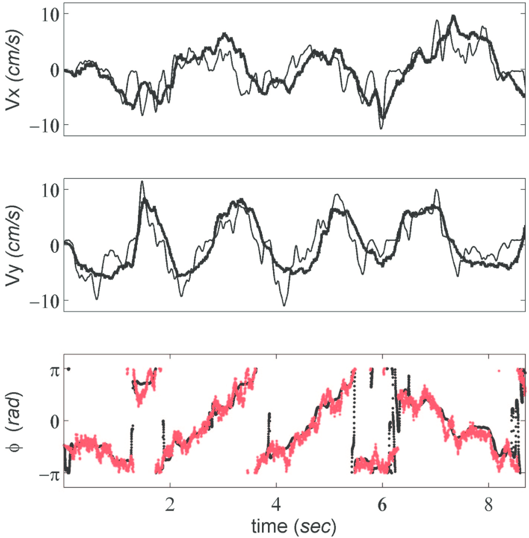

A
1400
N 700
0
0 25 50 100
ISI (ms)
B
2
1
0
AR coefficient
-7
20 40 60 80 100 120
order
C
5
4
3
PDF
2
1
0
0 0.5 1
z_i
D
2.5
2
1.5
mean ratio
1
0.5
0
0 50 100
ISI (ms)


Velocity
ARVel
1
0.5
0
z_{j+1}
0
0.5
1
z_j
z_j


A
ϕ
C(τ)
0.2
0
-2
0
2
B
|V|
0.1
0
-2
0
2
C
Vx
C(τ)
0
-0.2
-2
0
2
τ (sec)
D
Vy
0.5
0
-0.2
AR
Velocity
ARVel
-2
0
2
τ (sec)


CDF
Quantiles
Velocity
Ensemble Rates
Ensemble
IIGVelEnsRates
ARVel
ARVelEns
ARVelEnsRates


ARVelEnsRates
ARVelEns
ARVel
AR
IIGVelEnsRates
Ensemble Rates
VelEnsRates
VelEns
Velocity
Ensemble
Model
0 0.5 1 1.5 2 2.5 3 3.5 4
Δ AIC
x 10^4


10
0
−10
Vx (cm/s)
10
0
−10
Vy (cm/s)
π
0
−π
ϕ (rad)
2
4
6
8
time (sec)

$$A_k = \{\text{spike in } (t_{k-1}, t_k] \,|\, H_k\}$$

$$E_k = \{A_k\}^{\Delta N_k} \{A_k^c\}^{1-\Delta N_k}$$

$$H_k = \left\{ \bigcap_{j=1}^{k-1} E_j \right\} \qquad (A1)$$

$$P(N_{1:K}) = P(u_j \in (t_{k_j-1}, t_{k_j}], j = 1, \ldots, J, \cap N(T) = J)$$

$$= \prod_{k=1}^{K} P(A_k)^{\Delta N_k} P(A_k^c)^{1-\Delta N_k} \qquad (A2)$$

$$P(A_k) = \lambda(t_k \mid H_k)\Delta + o(\Delta)$$

$$P(A_k^c) = 1 - \lambda(t_k \mid H_k)\Delta + o(\Delta) \qquad (A3)$$

$$P(N_{1:K}) = \prod_{k=1}^{K} [\lambda(t_k \mid H_k)\Delta]^{\Delta N_k}[1 - \lambda(t_k \mid H_k)\Delta]^{1-\Delta N_k} + o(\Delta^J) \qquad (A4)$$

$$
\begin{aligned}
P(N_{1:K}) &= \prod_{k=1}^{K} [\lambda(t_k \mid H_k)\Delta]^{\Delta N_k}[1 - \lambda(t_k \mid H_k)\Delta]^{-\Delta N_k} \prod_{k=1}^{K} [1 - \lambda(t_k \mid H_k)\Delta] + o(\Delta^J) \\
&= \prod_{k=1}^{K} \left[\frac{\lambda(t_k \mid H_k)\Delta}{1 - \lambda(t_k \mid H_k)\Delta} \right]^{\Delta N_k} \prod_{k=1}^{K} \exp\{-\lambda(t_k \mid H_k)\Delta\} + o(\Delta^J) \\
&= \exp\left\{ \sum_{k=1}^{K} \log [\lambda(t_k \mid H_k)\Delta]\Delta N_k - \sum_{k=1}^{K} \lambda(t_k \mid H_k)\Delta \right\} + o(\Delta^J) \qquad (A5)
\end{aligned}
$$

$$P(N_{0:T}) = \lim_{\Delta \to 0} \frac{\exp\left\{ \sum_{k=1}^{K} \log\left[\lambda(t_k \mid H_k)\Delta\right]\Delta N_k - \sum_{k=1}^{K} \lambda(t_k \mid H_k)\Delta \right\} + o(\Delta^J)}{\Delta^J}$$

$$= \lim_{\Delta \to 0} \frac{\exp\left\{ \sum_{k=1}^{K} \log \lambda(t_k \mid H_k)\Delta N_k - \sum_{k=1}^{K} \lambda(t_k \mid H_k)\Delta \right\}\Delta^J + o(\Delta^J)}{\Delta^J}$$

$$= \exp\left\{ \int_0^T \log \lambda(t \mid H(t))\mathrm{d}N(t) - \int_0^T \lambda(t \mid H(t))\mathrm{d}t \right\} \qquad (A6)$$

$$f(y \mid \boldsymbol{\Theta}, \phi) = \exp\{[y\boldsymbol{\Theta} - b(\boldsymbol{\Theta})]/a(\phi) + c(y, \phi)\} \qquad (A7)$$

$$f(\mathbf{y} \mid \boldsymbol{\Theta}, \phi) \propto \exp\left\{ \sum_{k=1}^{K} y_k \log \lambda_k - \sum_{k=1}^{K} \lambda_k \right\} \qquad (A8)$$

$$f(\mathbf{y} \mid \mathbf{\Theta}, \phi) = \prod_{k=1}^{K} [P_k]^{y_k} [1 - P_k]^{1-y_k} \qquad (A9)$$

$$\log\left[\frac{P(A_k \mid \theta, H_k)}{1 - P(A_k \mid \theta, H_k)}\right] \approx \log\left[\lambda(t_k \mid \theta, H_k)\Delta\right] \qquad (A10)$$

END

Generalized linear model: nonlinearity on the response variable

$$F(y) = XB$$

Regularized GLM: Elastic net

- Concept: dynamic trade-off between L1 and L2

Regularized GLM: Elastic net

- Concept: dynamic trade-off between L1 and L2
    - Code crashed pretty hard.