Poisson GLM for neural data analysis

- Poisson generalized linear model (GLM) with regularization.
- Unlike Glmnet toolbox (https://glmnet.stanford.edu/articles/glmnet.html) or built-in LASSOGLM function, this will have different lambda values for each variable type, and will search in the n_variable-D grid space.
- Implemented higher-order Tikhonov regularization. This could be useful for smooth parameters.

Example

```
n_sample = 3600; n_var = 30;
c = -1;
beta1 = 0.2 * sin(linspace(0, pi, n_var))';
beta2 = 0.2 * cos(linspace(0, 4*pi, n_var))';
X_1 = randn(n_sample, n_var);
X_2 = randn(n_sample, n_var);
mu = [X_1, X_2] * [beta1; beta2] + c;
y = poissrnd(exp(mu), n_sample, 1);

% Zeroth-order Tikhonov regularization
out0 = glm.cvglm({X_1, X_2}, y);

% First-order Tikhonov regularization
out1 = glm.cvglm({X_1, X_2}, y, 'order', 1);

% Second-order Tikhonov regularization
out2 = glm.cvglm({X_1, X_2}, y, 'order', 2);

fig = figure(1); clf;
subplot(121); hold on;
plot(beta1, 'k');
plot(out0.w(out0.prm.index{2}), ':r');
plot(out1.w(out1.prm.index{2}), ':g');
plot(out2.w(out2.prm.index{2}), ':b');
subplot(122); hold on;
plot(beta2, 'k');
plot(out0.w(out0.prm.index{3}), ':r');
plot(out1.w(out1.prm.index{3}), ':g');
plot(out2.w(out2.prm.index{3}), ':b');
```

Notes

Log linear model

Firing rate at time t can be modeled as,

$$r(t) = e^{(\mathbf{k}*\mathbf{x})(t)+c}$$

Or we can directly model spike number per time bin

$$n(t) = e^{(\mathbf{k}*\mathbf{x})(t)+c}/dt$$

Time bin size dt is only affecting constant, so it could be potentially ignored. $\mathbf{x}$ is input stimuli, $\mathbf{k}$ is a kernel function, and c is a constant. The kernel $\mathbf{k}$ can be decomposed into multiple subkernels like boxcar functions or cosine bumps.

$$\mathbf{k} = \mathbf{k_1}w_1 + \mathbf{k_2}w_2 + \mathbf{k_3}w_3$$

Distributivity and associativity can be applied to convolution.

$$f * (g + h) = f * g + f * h$$

$$a(f * g) = (af) * g$$

Therefore,

$$r(t) = e^{(\mathbf{k_1}*\mathbf{x})(t)w_1+(\mathbf{k_2}*\mathbf{x})(t)w_2+(\mathbf{k_3}*\mathbf{x})(t)w_3+c}$$

$$r(t) = e^{\begin{bmatrix}1 & (\mathbf{k_1}*\mathbf{x})(t) & (\mathbf{k_2}*\mathbf{x})(t) & (\mathbf{k_3}*\mathbf{x})(t)\end{bmatrix}\begin{bmatrix}c\\w_1\\w_2\\w_3\end{bmatrix}}$$

$$\mathbf{r} = e^{\mathbf{X}\mathbf{w}}$$

where $\mathbf{r} \in \mathbb{R}^{m\times 1}$ is a column vector for firing rate, $\mathbf{X} \in \mathbb{R}^{m\times n}$ is a combined matrix convolved by kernels, and $\mathbf{w} \in \mathbb{R}^{n\times 1}$ is a weight vector for these kernels. m is the number of time bins, n is the number of variables including constant.

Poisson negative log-likelihood

The probability that we get certain number of spikes ($y(t)$) for each time bin with expected spike count ($n(t)$) is calculated like this,

$$p(y(t)|n(t)) = \frac{e^{-n(t)}(n(t))^{y(t)}}{y(t)!}$$

$$-log(p(t)) = n(t) - y(t)log(n(t)) + log(y(t)!)$$

Then we sum every time bin,

$$-log(p) = \sum_t (n(t) - y(t)log(n(t))) + log(y(t)!)$$

$$-log(p) = (\sum_t n(t)) - \mathbf{y}'\mathbf{X}\mathbf{w} + c$$

where c are terms that are not dependent on $\mathbf{w}$. To maximize p, we should minimize negative log-likelihood.

Gradient for $\mathbf{w}$ is,

$$\mathbf{X}'(\mathbf{n} - \mathbf{y})$$

Hessian for $\mathbf{w}$ is,

$$\mathbf{X}'diag(\mathbf{n})\mathbf{X}$$

Tikhonov regularization

We want to minimize penalized negative log-likelihood.

$$\min_w -log(p) + \sum_i \frac{1}{2}\lambda_i \left\|\mathbf{L}_i\mathbf{w}_i\right\|_2^2$$

where the matrix $L \in \mathbb{R}^{k\times n}, k \leq n$ is the regularization operator.

Common choices for regularization operator is the identity matrix (=zeroth-order Tikhonov regularization=ridge regression). For smooth weights, scaled finite derivative matrices could be used, such as

$$L_1 = \frac{1}{2}\begin{bmatrix} -1 & 1 & & & \mathbf{0} \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ \mathbf{0} & & & -1 & 1 \end{bmatrix},$$

$$L_2 = \frac{1}{4}\begin{bmatrix} 1 & -2 & 1 & & & \mathbf{0} \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ \mathbf{0} & & & 1 & -2 & 1 \end{bmatrix}$$