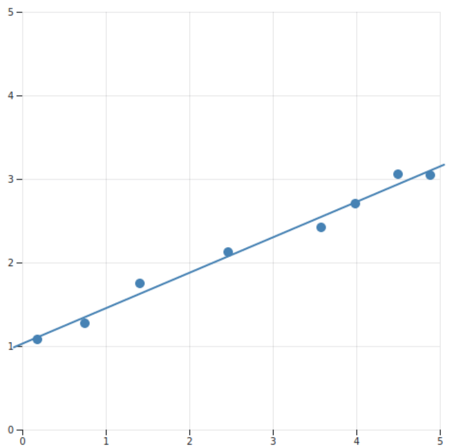
Photogrammetry & Robotics Lab

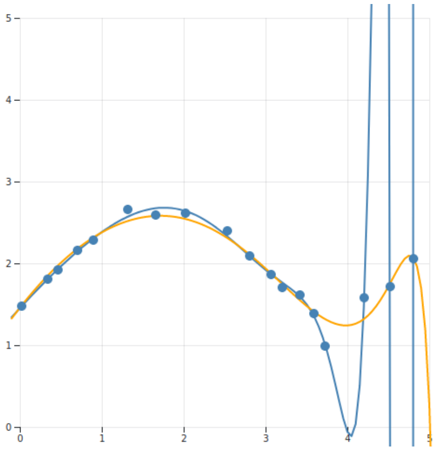
Machine Learning for Robotics and Computer Vision Tutorial

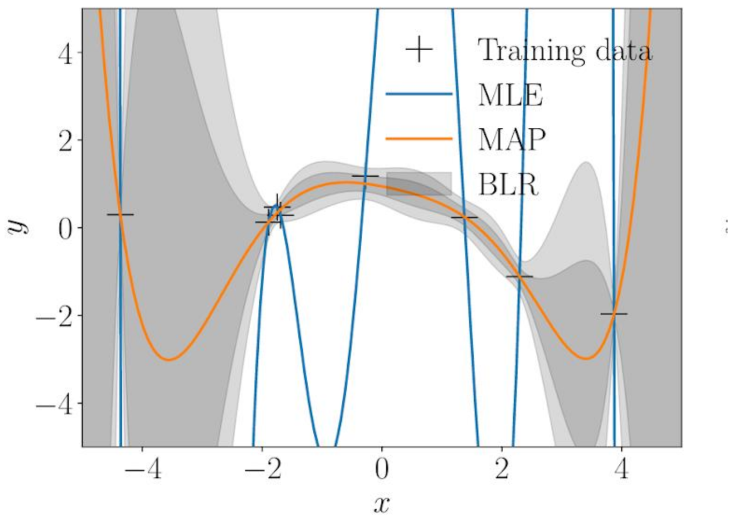
More on Regression

Jens Behley

Topics of this week lecture







- Linear Regression (ML, MAP, Bayesian)
- Closed-form solutions! (not always possible)

Recap: Linear Regression

- Under this assumptions, this leads to the following probabilistic formulation

$$P(y|\mathbf{x},\theta) = \mathcal{N}(y|f(\mathbf{x}),\sigma^2)$$

- In linear regression, we assume that parameters θ appear **linearly** in our model

$$f(\mathbf{x}) = \mathbf{x}^T\theta + \theta_0$$

- θ_0 is called **intercept** (or **bias**) that enables us to have also functions that do not pass through the origin

[MML Chap. 9.1]

Relation to Least Squares

- Maximum Likelihood:

$$\theta^\star = \arg\min_\theta - \log \prod_{n=1}^{N} P(y_n|\mathbf{x}_n, \theta)$$

- Showed that results in NLL:

$$\mathcal{L}(\theta) := \frac{1}{2\sigma^2} \sum_{n=1}^{N} \underbrace{\left(y_n - \mathbf{x}_n^T \theta\right)^2}_{\textbf{L2 Loss}}$$

- → NLL of Linear Regression is just Least Squares!
- (Losses are often denoted by $\ell(y_n, \hat{y}_n)$)

Recap: Non-linear Functions

- Linear regression is *linear in parameters*
- We can apply non-linear transformation:

$$f(\mathbf{x}) = \mathbf{x}^T\theta \longrightarrow f(\mathbf{x}) = \phi(\mathbf{x})^T\theta$$

- Let $\phi : \mathbb{R}^D \to \mathbb{R}^K$ and define $\mathbf{\Phi} \in \mathbb{R}^{N\times K}$ as

$$\mathbf{\Phi} = \begin{pmatrix} \phi(\mathbf{x}_1)^T \\ \vdots \\ \phi(\mathbf{x}_N)^T \end{pmatrix} \in \mathbb{R}^{N\times K}$$

- Everything else stays the same! Use normal equation with $\mathbf{\Phi}$ instead of $\mathbf{X}$

$$\theta = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{y}$$

[MML Chap. 9.2.1] [CVMLI Chap. 8.3]

Example: Polynomial transformation

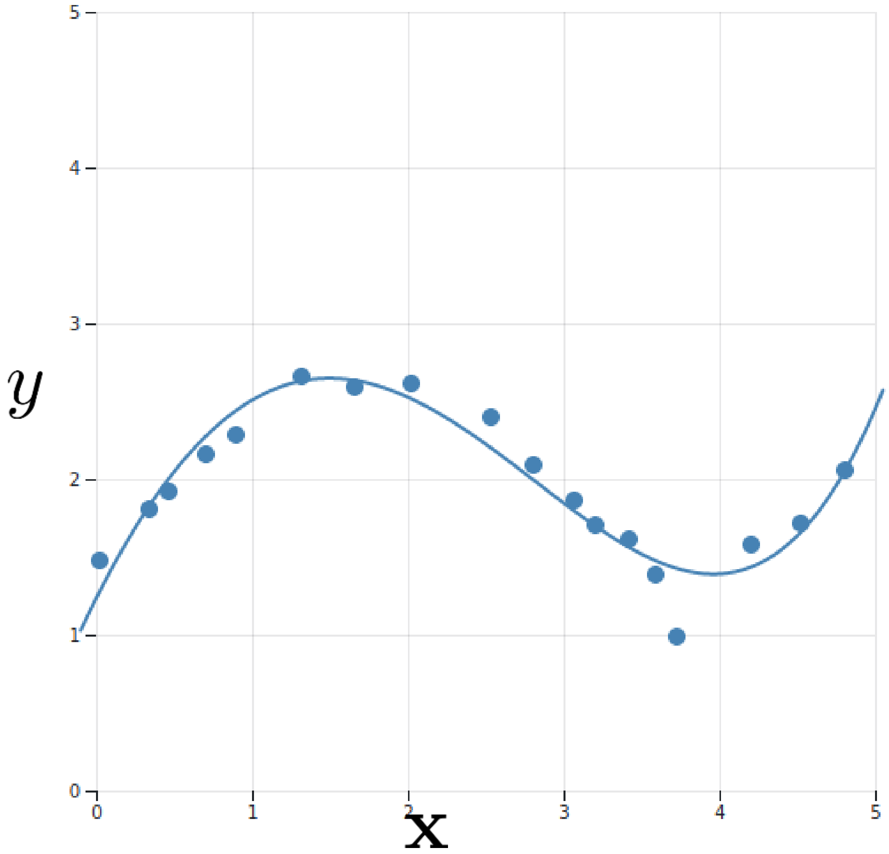
- With polynomial transformation, we can fit polynomials of degree K

$$\phi_{\text{poly}}(\mathbf{x}_n) = \begin{pmatrix} 1 \\ x_1 \\ x_1^2 \\ \vdots \\ x_1^K \\ \vdots \end{pmatrix} \in \mathbb{R}^{DK+1}$$

- With K=1 it's "vanilla" linear regression

[MML Chap. 9.2.1]

Recap: Example: Polynomial Fit



- With a polynomial of degree 3, we get a good fit. But can we do better with higher degrees?

Potential Problem

$$\phi_{\text{poly}}(\mathbf{x}_n) = \begin{pmatrix} 1 \\ x_1 \\ x_1^2 \\ \vdots \\ x_1^K \\ \vdots \end{pmatrix} \in \mathbb{R}^{DK+1}$$

- What happens when x =(10,103,1005) and we want K = 10?

Potential Problem

$$\phi_{\text{poly}}(\mathbf{x}_n) = \begin{pmatrix} 1 \\ x_1 \\ x_1^2 \\ \vdots \\ x_1^K \\ \vdots \end{pmatrix} \in \mathbb{R}^{DK+1}$$

- What happens when x =(10,103,1005) and we want K = 10?
  - 1005^{10} = 10.511.401.320.40.790.642.597.666.015.625

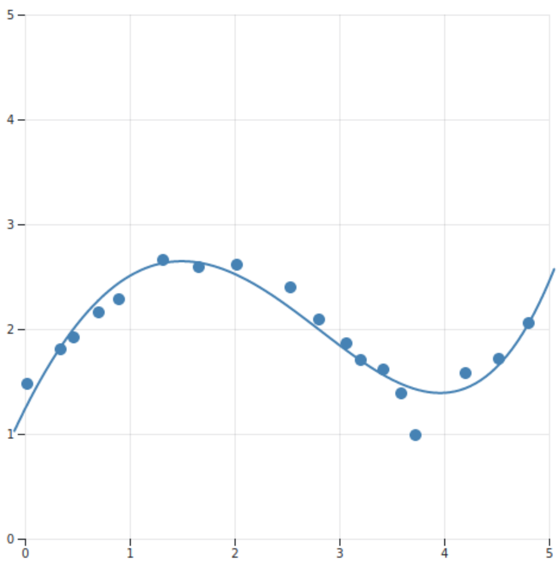
How can we avoid numerical overflow?

Solution: Feature Normalization

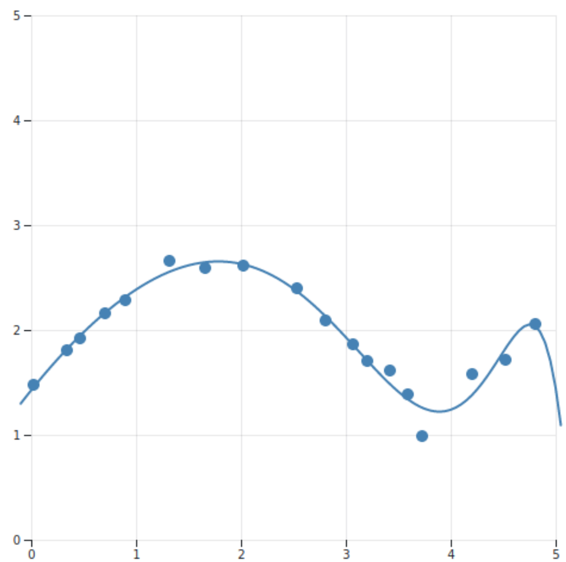
- Feature normalization ensures that polynomial transform get’s not too large
  - **Important:** Same normalization to test samples
- Different options:
  - Divide by maximum value
    x = (10,103,1005)/(40,150,2000)
    ➔ (0.25, 0.69, 0.50) 0.69^10 = 0.0244
  - Standardization (especially with image data)
    - Divide by mean and variance of training data

- If you provide pre-trained models: provide normalization constants

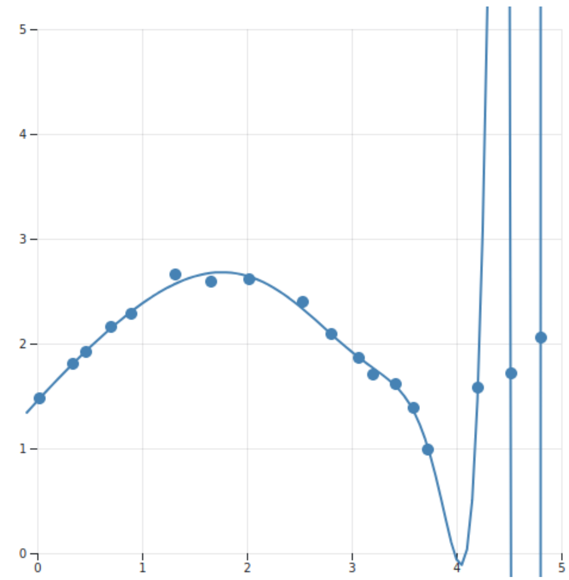
Recap: Example: Overfitting



degree 3



degree 5



degree 8

- Increasing the degree, we will get better training error
- But function will have implausible shape

Common Parameter Estimation

- Learning is finding parameters of

$$P(\theta|\mathbf{x}_{1:N}, y_{1:N}) \propto P(y_{1:N}|\mathbf{x}_{1:N}, \theta) \boxed{P(\theta)}$$

- Paradigms for parameter estimation:

1. Point estimate with uniform prior
   → Maximum Likelihood Estimation
2. Point estimate with given prior
   → Maximum A posteriori Estimation (MAP)
3. Determine posterior over the parameters
   → Bayesian Estimation

NLL with Prior

- Assume Gaussian prior for parameters:

$$P(\theta) = \mathcal{N}(\theta | 0, b^2 \mathbf{Id})$$

- NLL is then

$$\mathcal{L}_{\text{MAP}}(\theta) = -\log \prod_{n=1}^{N} P(y_n | \mathbf{x}_n, \theta) \underline{- \log P(\theta)}$$

- Inserting Gaussians results in

Squared Length

$$\mathcal{L}_{MAP}(\theta) = \frac{1}{2\sigma^2} \sum_{n=1}^{N} \left(y_n - \phi(\mathbf{x}_n)^T \theta \right)^2 + \frac{1}{2b^2} \theta^T \theta + \text{const}$$

$$= \frac{1}{2\sigma^2} (\mathbf{y} - \mathbf{\Phi}\theta)^T (\mathbf{y} - \mathbf{\Phi}\theta) + \frac{1}{2b^2} \theta^T \theta + \text{const}$$

NLL with Prior

- Assume Gaussian prior for parameters:

$$P(\theta) = \mathcal{N}(\theta | 0, b^2 \mathbf{Id})$$

- NLL is then

$$\mathcal{L}_{\text{MAP}}(\theta) = -\log \prod_{n=1}^{N} P(y_n | \mathbf{x}_n, \theta) \underline{- \log P(\theta)}$$

- Inserting Gaussians results in

Squared Length

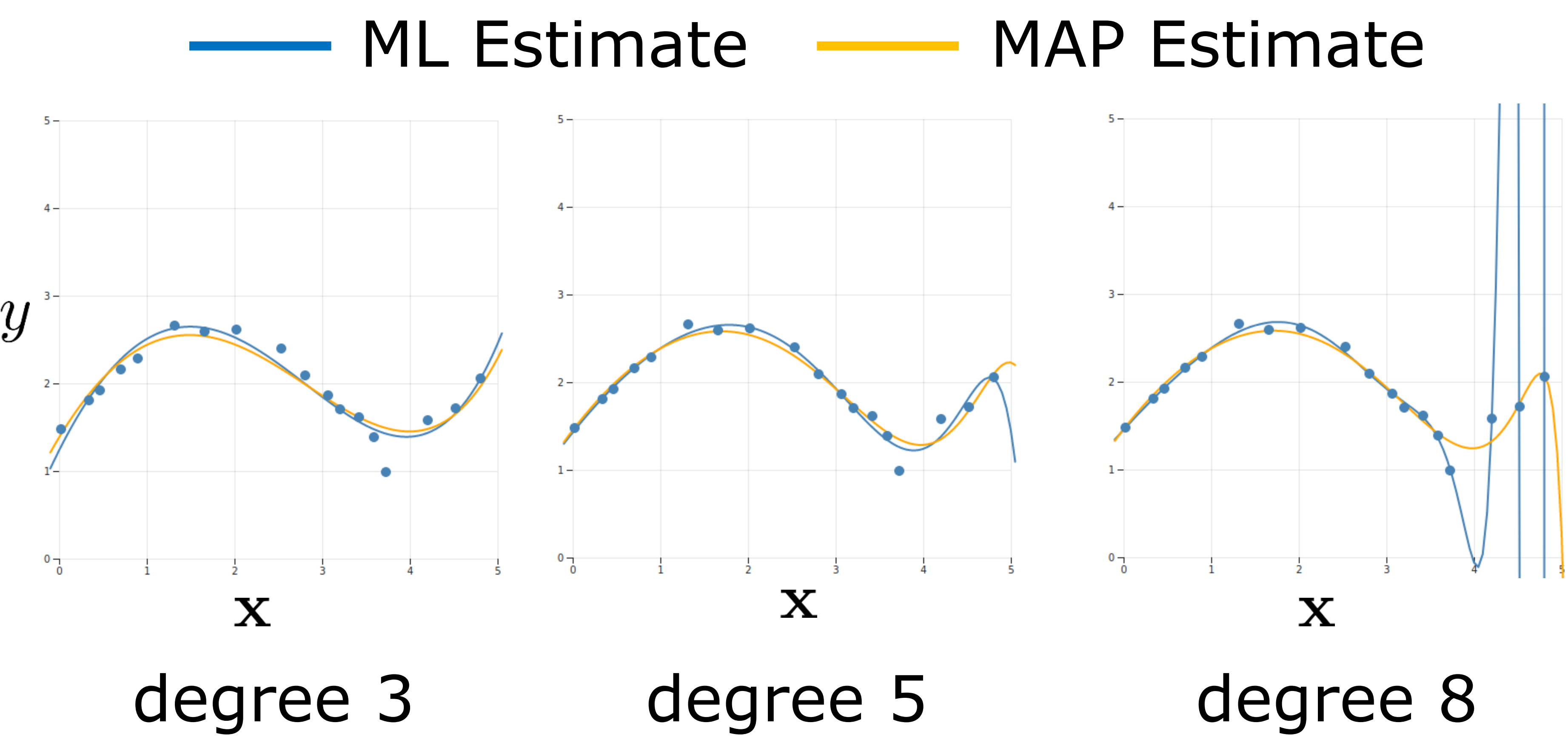
$$\mathcal{L}_{MAP}(\theta) = \frac{1}{2\sigma^2} \sum_{n=1}^{N} \left(y_n - \phi(\mathbf{x}_n)^T \theta \right)^2 + \frac{1}{2b^2} \theta^T \theta + \text{const}$$

$$= \frac{1}{2\sigma^2} (\mathbf{y} - \mathbf{\Phi}\theta)^T (\mathbf{y} - \mathbf{\Phi}\theta) + \boxed{\lambda} \theta^T \theta + \text{const}$$

Regularizer

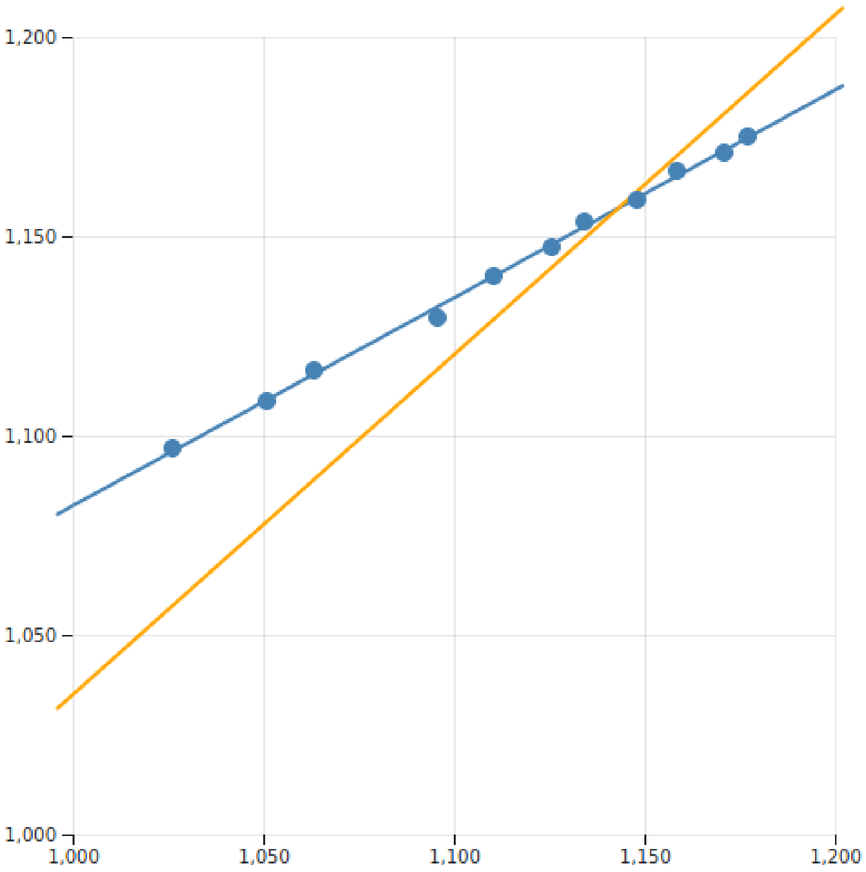
[MML Chap. 9.2.3]

Example: ML vs. MAP Estimate



- Smoother functions even at higher degrees!

But there is a problem ...



$$\lambda = 0.1$$

- Let's say y is in [1000, 1200]

Any idea?

Bias term

$$= \frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{\Phi}\theta)^T(\mathbf{y} - \mathbf{\Phi}\theta) + \boxed{\lambda}\,\theta^T\theta + \text{const}$$

- As also the bias term is regularized, but needs to be large…
- **Solution:** Exclude bias term in regularization (set $(\theta^T\theta)_0 = 0$)

See you next week!