Gaussian Process (Closer look)

Presenter: Ryan Roussel

Day 4

Outline

- Detailed look at GP calculations
- Practical implementation of GP in python - GPyTorch

Gaussian distribution vs Gaussian process

Gaussian distributions $N(\mu, \Sigma)$

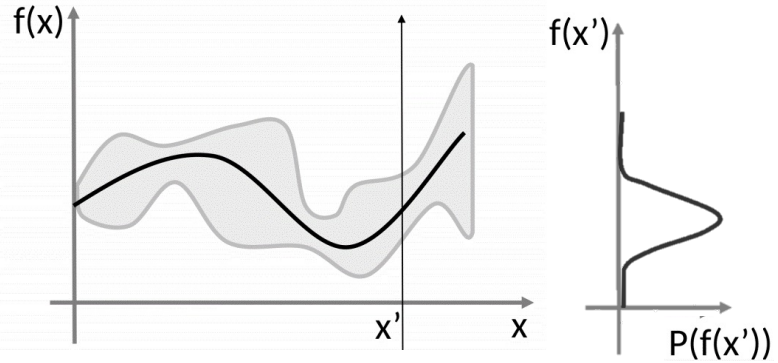
- Distribution over vectors.
- Fully specified by a mean and covariance.

Normal dist.
(1-D Gaussian)

What does this actually mean???

Gaussian processes $GP(m(x), k(x, x'))$

- Distribution over functions.
- Fully specified by a mean function and covariance function.

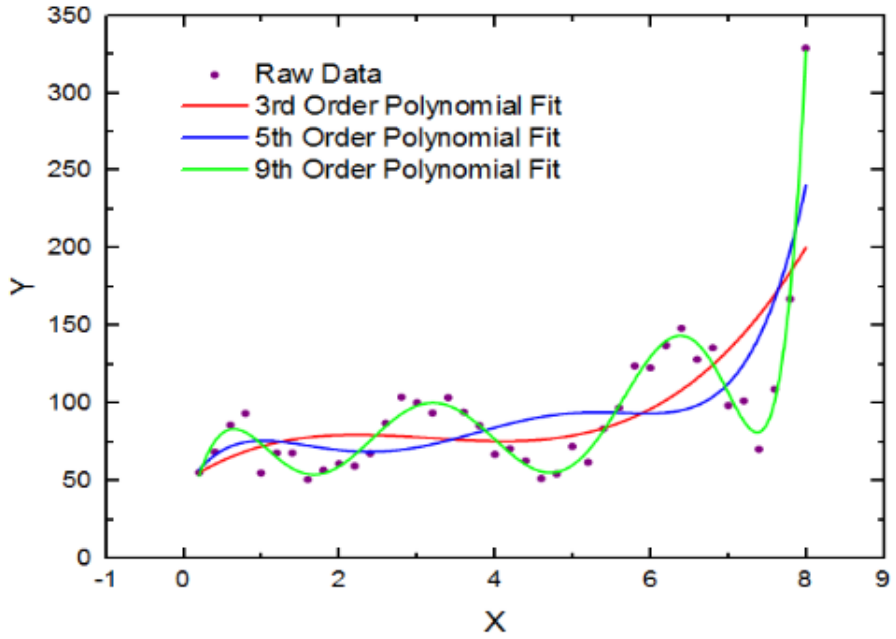


Gaussian process
(∞-D Gaussian)

Remember: Parametric Models vs. Non-parametric Models

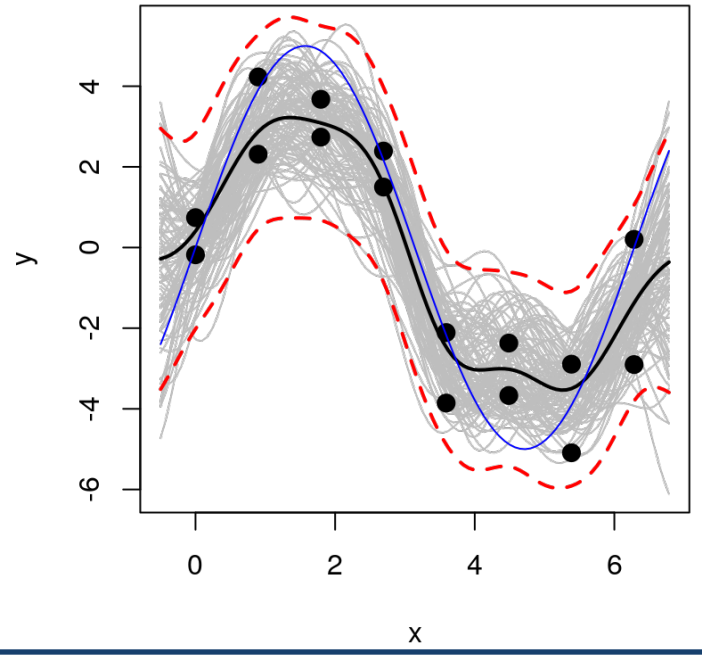
Parametric models

- Polynomial regression

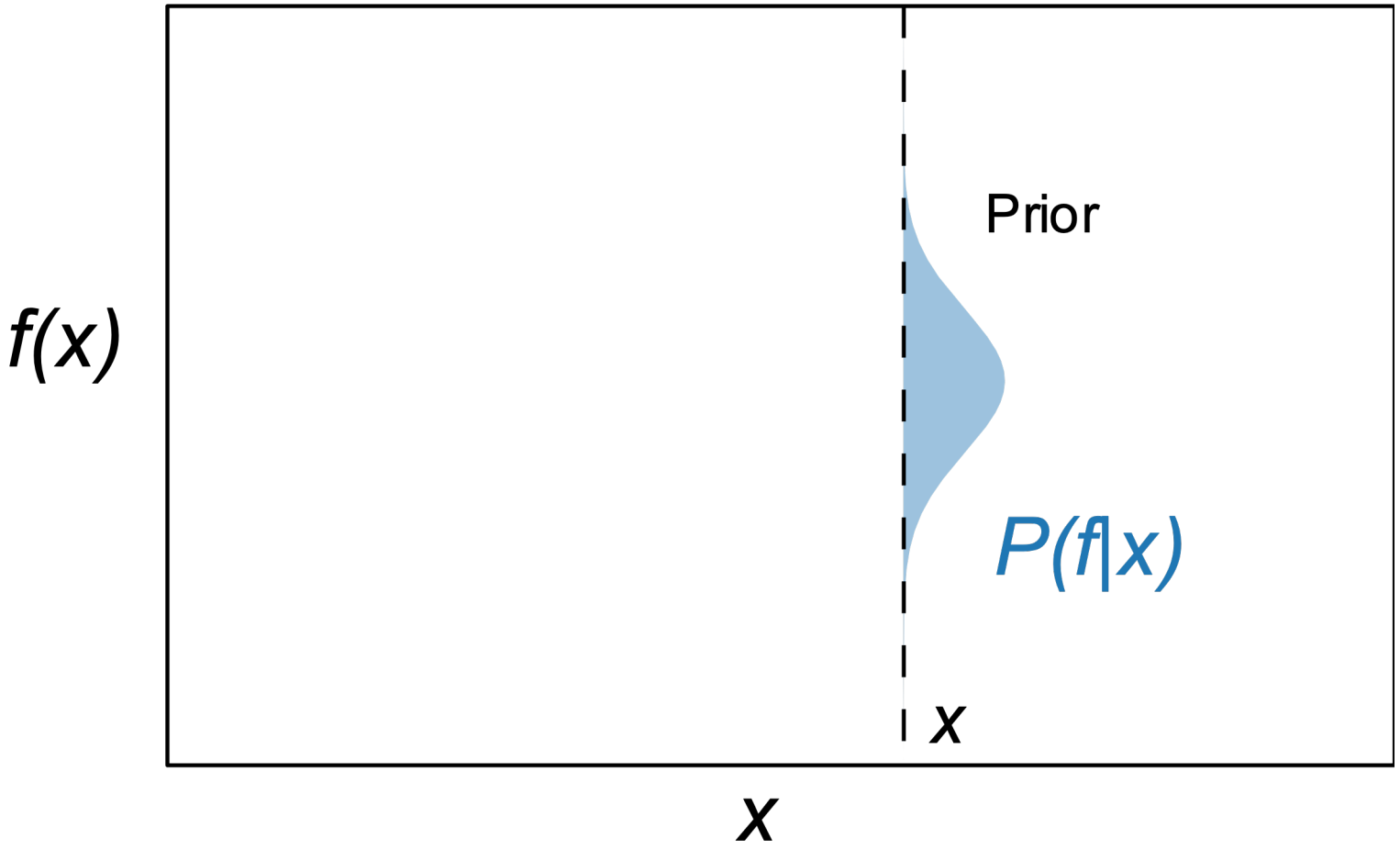


Non-parametric models

- Gaussian Process

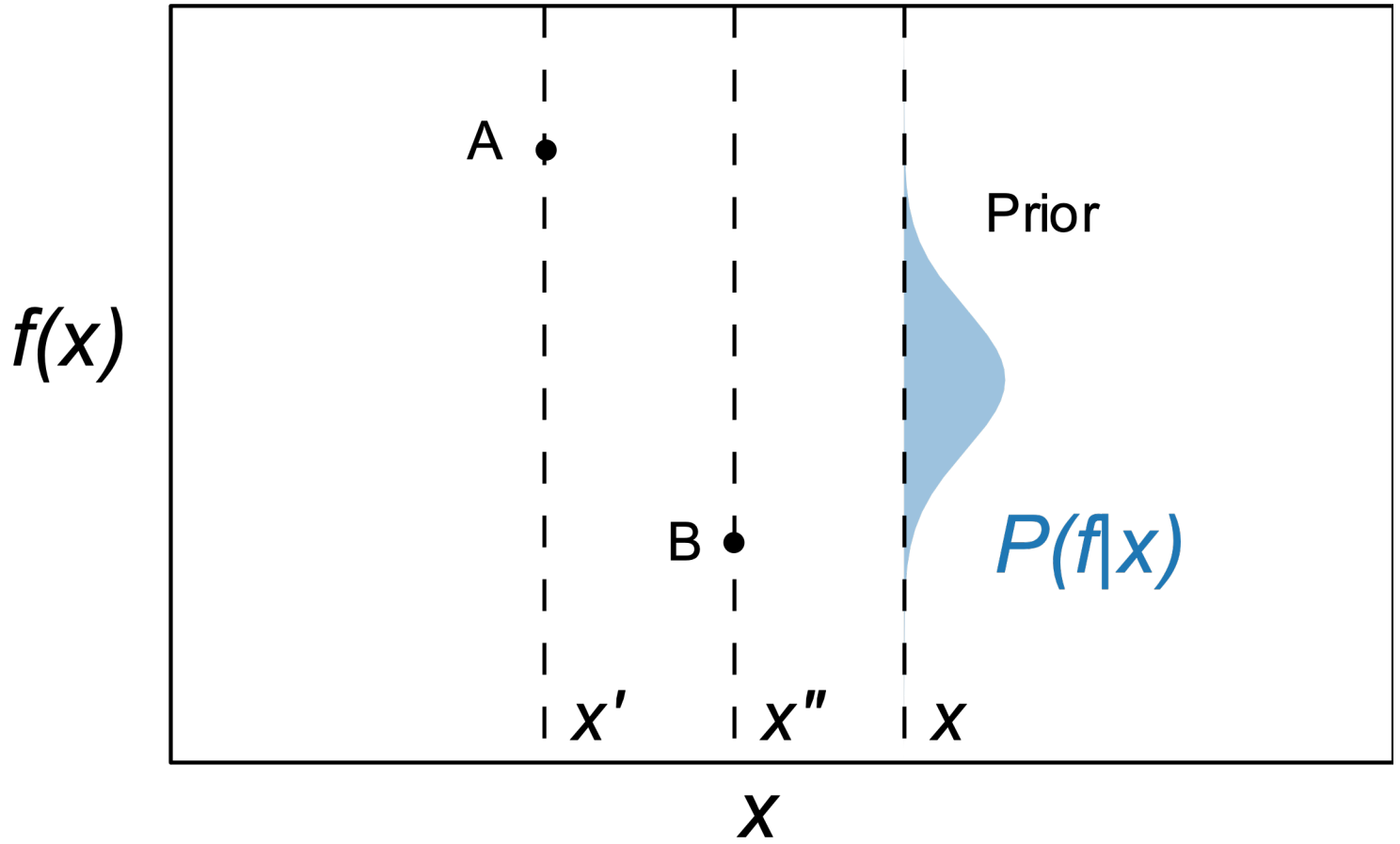


Some Intuition



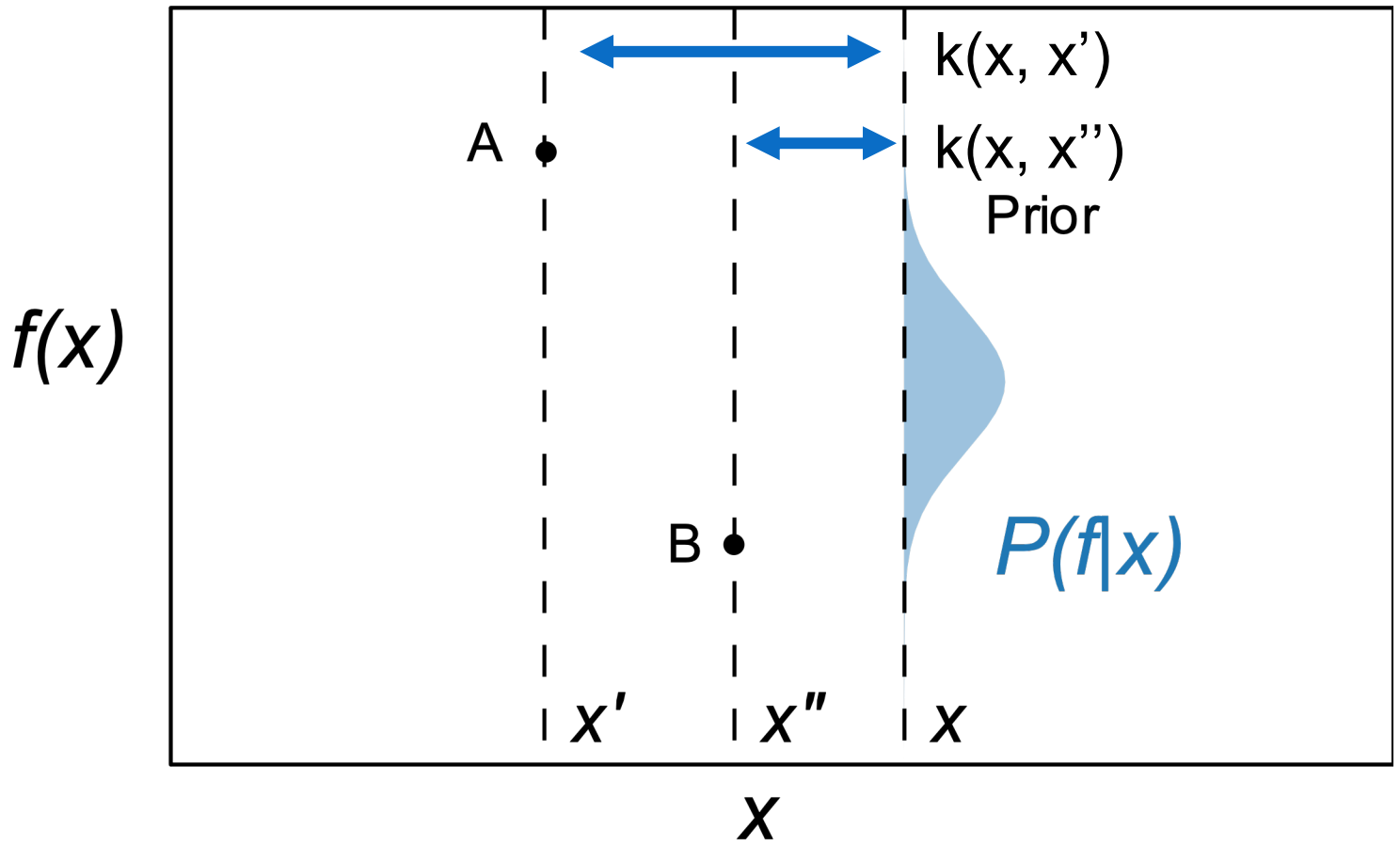
Some Intuition

Which observation will have a larger impact on changing P(*f* | *x*)?



Some Intuition

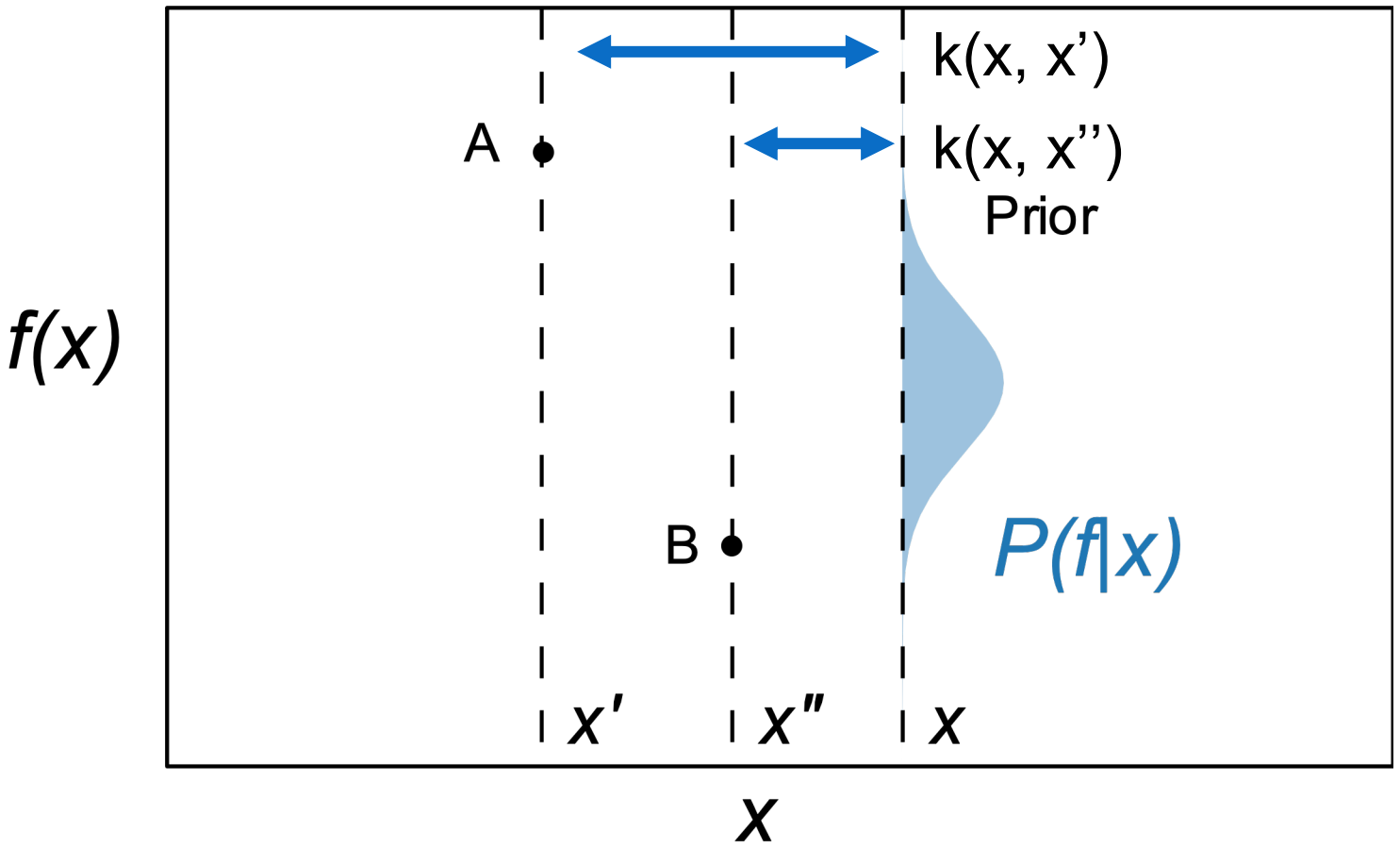
Which observation will have a larger impact on changing P(*f* | *x*)?



k(x, x') < k(x, x'')

Some Math

How do we quantify our observations impact on P(f | x)?



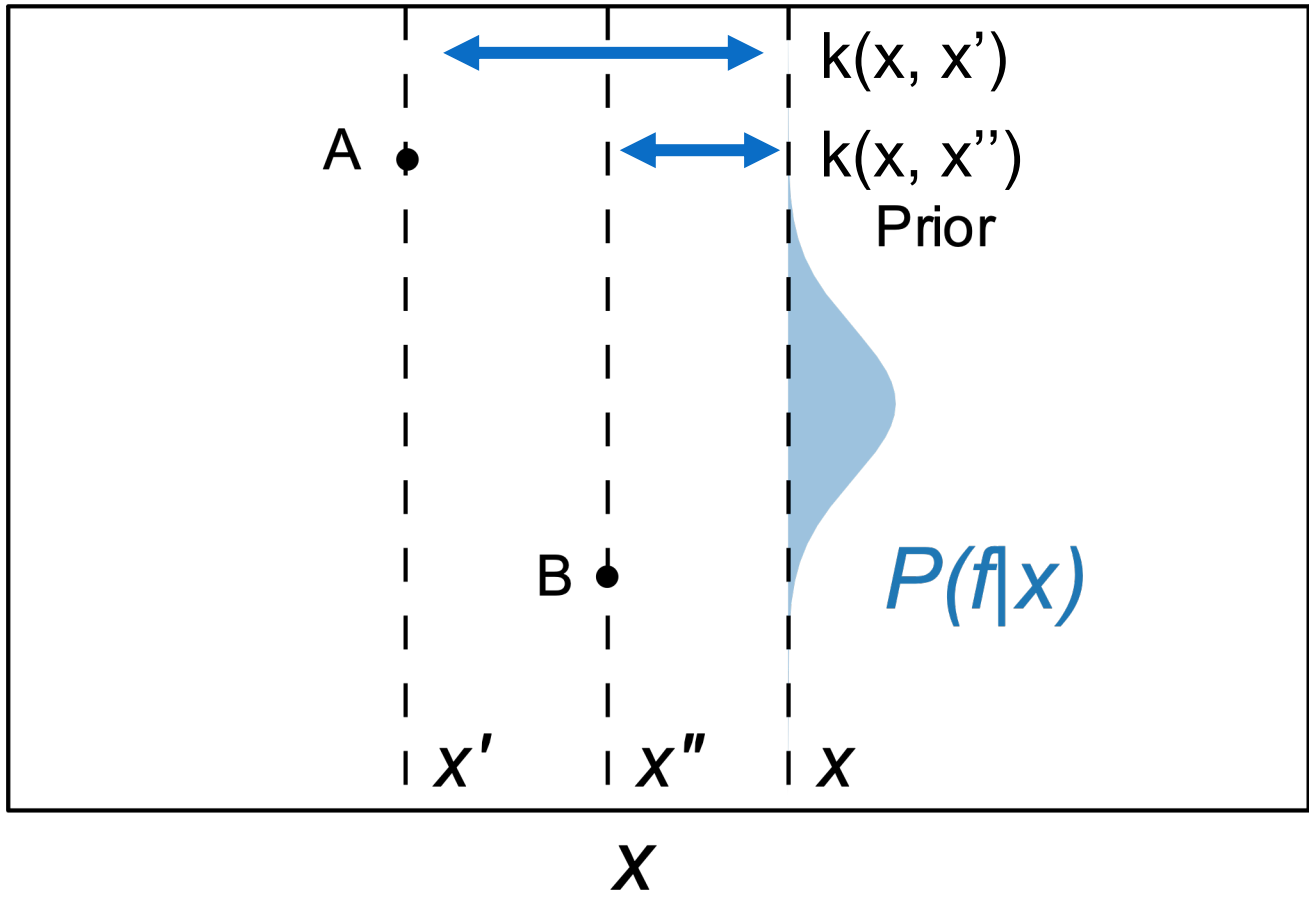
$$p(f_A, f_B, f) = N(\boldsymbol{\mu}, \Sigma)$$

$$\Sigma = \begin{pmatrix} k(x',x') & k(x',x'') & k(x',x) \\ k(x',x'') & k(x'',x'') & k(x'',x) \\ k(x',x) & k(x'',x) & k(x,x) \end{pmatrix}$$

$$p(f, | f_A, f_B) = \frac{p(f, f_A, f_B)}{p(f_A, f_B)}$$

Some Math

How do we quantify our observations impact on P(f | x)?



$$\Sigma = \begin{pmatrix} k(x',x') & k(x',x'') & k(x',x) \\ k(x',x'') & k(x'',x'') & k(x'',x) \\ k(x',x) & k(x'',x) & k(x,x) \end{pmatrix}$$

$$p(f,|\, f_A, f_B) = \frac{p(f, f_A, f_B)}{p(f_A, f_B)}$$

$$p(f,|\, f_A, f_B) = N(\mu^*, \sigma^*)$$

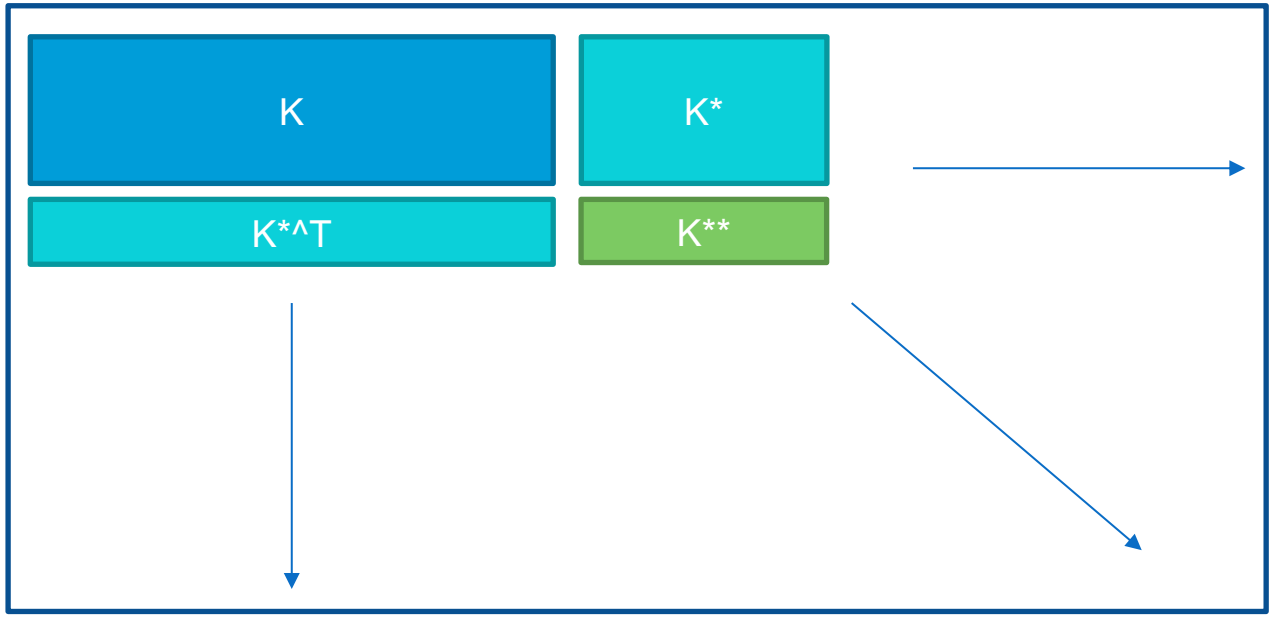
$$\mu^* = K^* K^{-1} \boldsymbol{f}$$

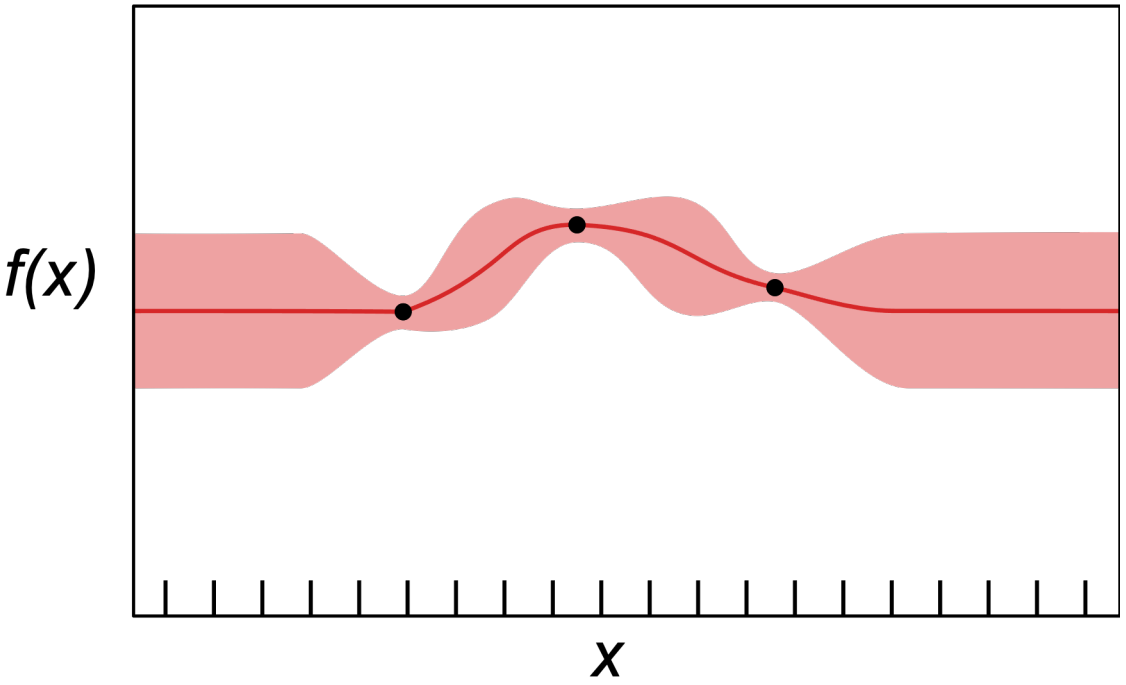
$$\sigma^* = K^{**} - K^{*T} K^{-1} K^*$$

Note: Zero prior mean

Making Predictions in Practice

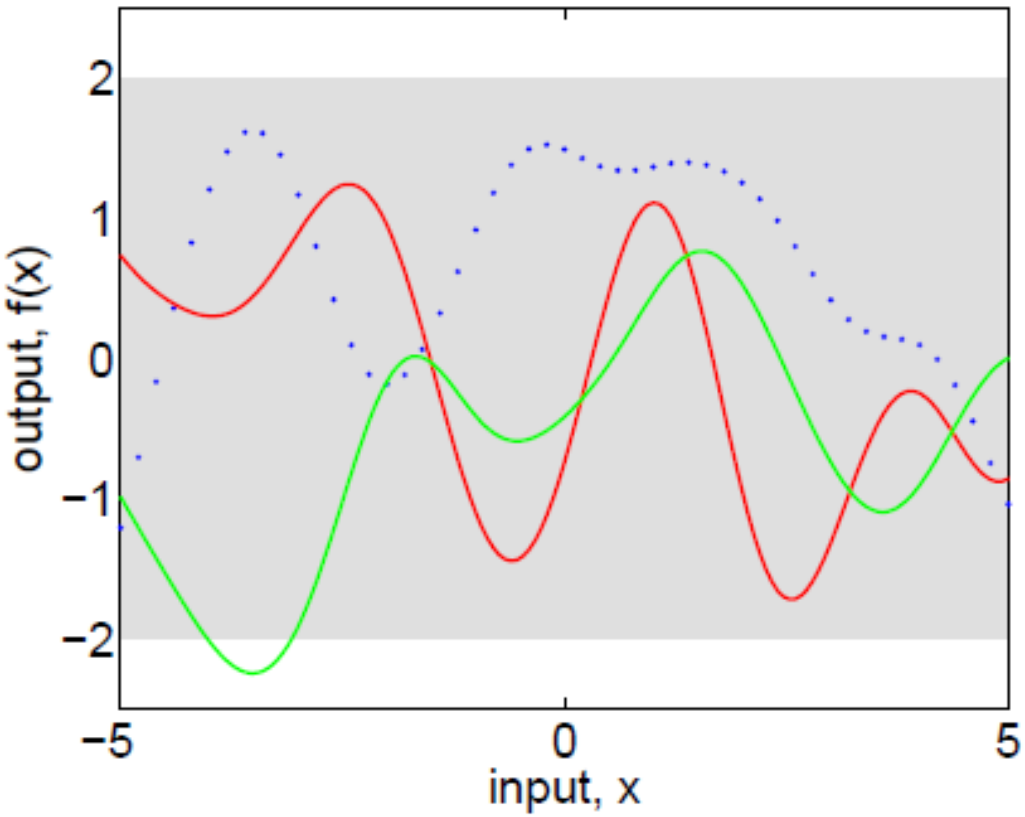
What about multiple predictions? Note: we only have to invert K



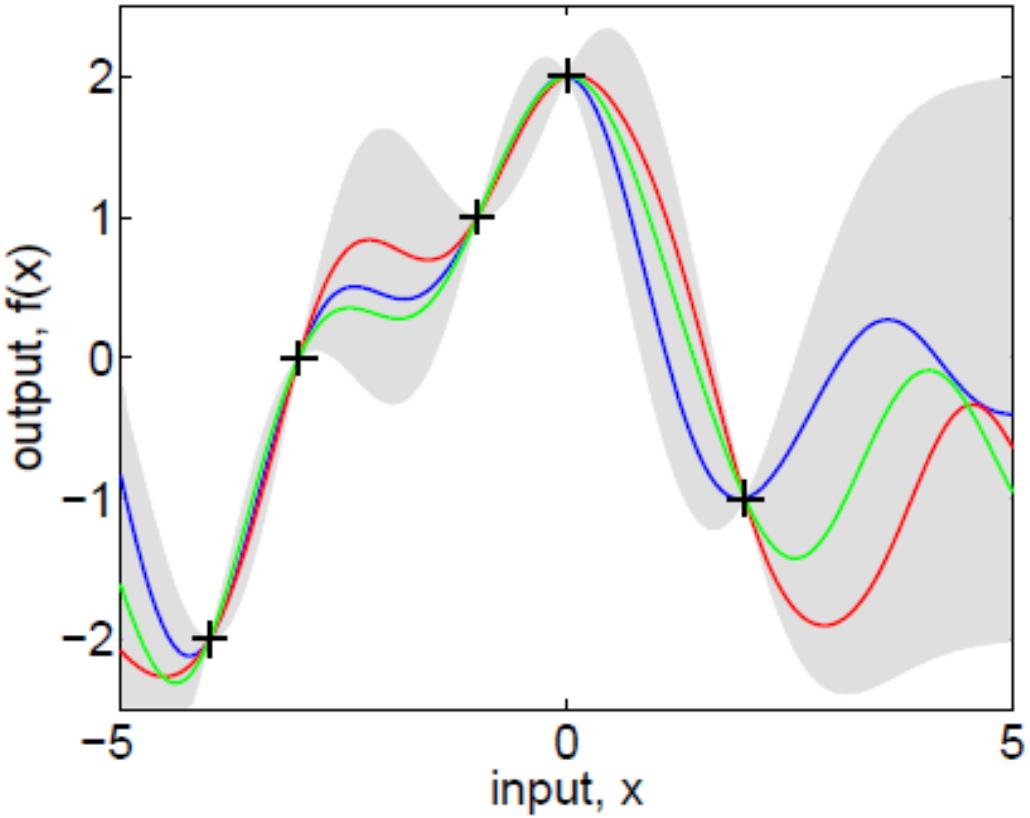


Draw function samples? Sample from the joint posterior distribution at requested points

Another viewpoint



Prior



Posterior

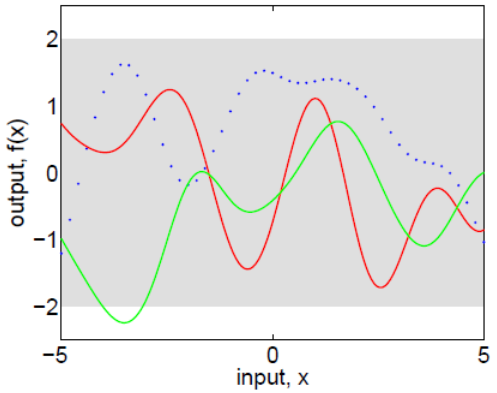
Implementation in GPyTorch – Predictions

```python
# We will use the simplest form of GP model, exact inference
class ExactGPModel(gpytorch.models.ExactGP):
    def __init__(self, train_x, train_y, likelihood):
        super(ExactGPModel, self).__init__(train_x, train_y, likelihood)
        self.mean_module = gpytorch.means.ConstantMean()
        self.covar_module = gpytorch.kernels.ScaleKernel(gpytorch.kernels.RBFKernel())

    def forward(self, x):
        mean_x = self.mean_module(x)
        covar_x = self.covar_module(x)
        return gpytorch.distributions.MultivariateNormal(mean_x, covar_x)

# initialize likelihood and model
likelihood = gpytorch.likelihoods.GaussianLikelihood()
model = ExactGPModel(train_x, train_y, likelihood)
```

$p(f|x, X, \phi) = N(m'(x), \sigma'^2(x))$
(prior)

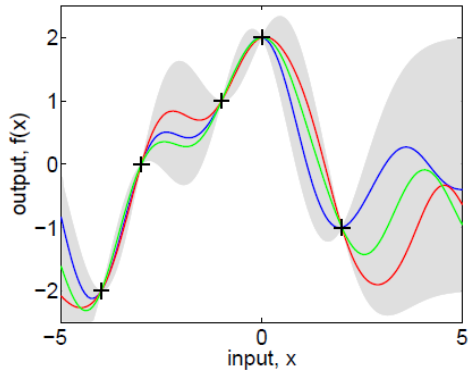


$p(f|x, \boldsymbol{y}, X, \phi) = N(m(x), \sigma^2(x))$
(posterior)

```python
f_preds = model(test_x)
y_preds = likelihood(model(test_x))

f_mean = f_preds.mean
f_var = f_preds.variance
f_covar = f_preds.covariance_matrix
f_samples = f_preds.sample(sample_shape=torch.Size(1000,))
```

$p(y|f, \theta) = N(f, \theta^2)$
(posterior predictive)



Implementation in GPyTorch – Hyperparameter Fitting

```
# this is for running the notebook in our testing framework
import os
smoke_test = ('CI' in os.environ)
training_iter = 2 if smoke_test else 50


# Find optimal model hyperparameters
model.train()
likelihood.train()

# Use the adam optimizer
optimizer = torch.optim.Adam(model.parameters(), lr=0.1)  # Includes GaussianLikelihood parameters

# "Loss" for GPs - the marginal log likelihood
mll = gpytorch.mlls.ExactMarginalLogLikelihood(likelihood, model)

for i in range(training_iter):
    # Zero gradients from previous iteration
    optimizer.zero_grad()
    # Output from model
    output = model(train_x)
    # Calc loss and backprop gradients
    loss = -mll(output, train_y)
    loss.backward()
    print('Iter %d/%d - Loss: %.3f   lengthscale: %.3f   noise: %.3f' % (
        i + 1, training_iter, loss.item(),
        model.covar_module.base_kernel.lengthscale.item(),
        model.likelihood.noise.item()
    ))
    optimizer.step()
```

$$p(y|\vec{x}) = \int p(y|f,\vec{x})p(f|\vec{x})df$$

$$\log p(y|\vec{x}) = \underbrace{-\frac{1}{2}y^{\mathrm{T}}K^{-1}y}_{\text{Data fit}} \underbrace{-\frac{1}{2}\log|K|}_{\text{Complexity penalty}} - \frac{N}{2}\log 2\pi$$

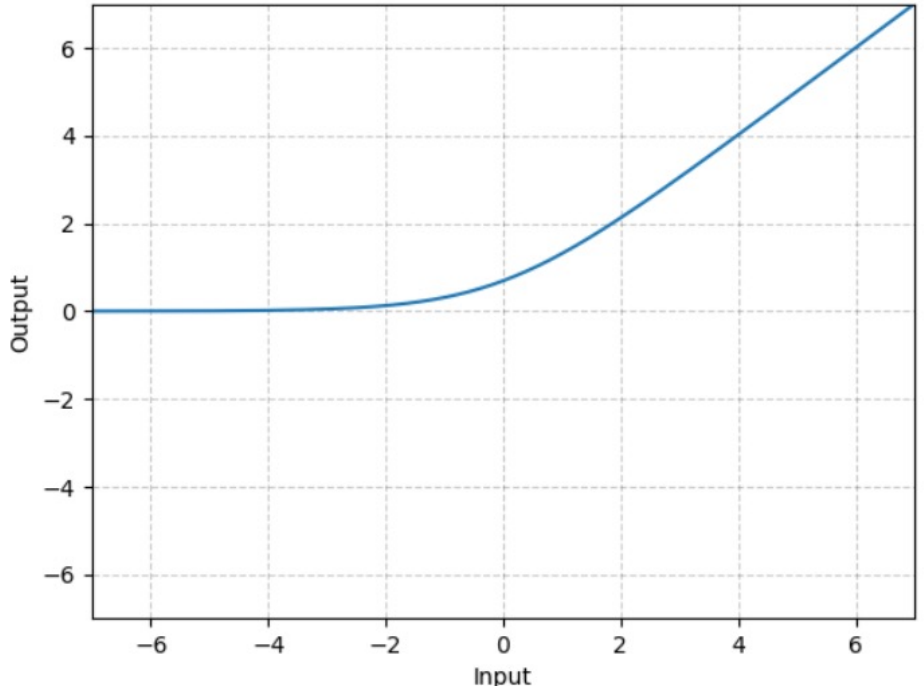
Implementation in GPyTorch – Hyperparameter Fitting (constraints)

```
for param_name, param in model.named_parameters():
    print(f'Parameter name: {param_name:42} value = {param.item()}')
```

```
Parameter name: likelihood.noise_covar.raw_noise           value = 0.0
Parameter name: mean_module.constant                       value = 0.0
Parameter name: covar_module.raw_outputscale               value = 0.0
Parameter name: covar_module.base_kernel.raw_lengthscale   value = 0.0
```

```
print(f'Actual noise value: {likelihood.noise}')
print(f'Noise constraint: {likelihood.noise_covar.raw_noise_constraint}')
```

```
Actual noise value: tensor([0.6932], grad_fn=<AddBackward0>)
Noise constraint: GreaterThan(1.000E-04)
```



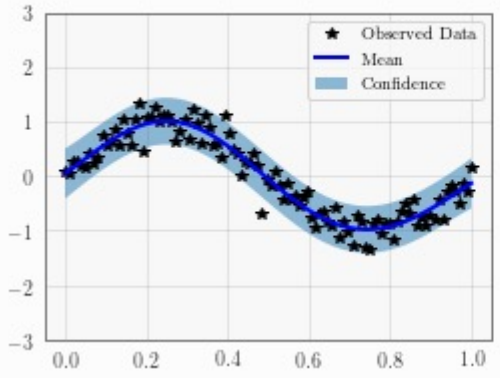
Softplus transform

Implementation in GPyTorch – Data Preprocessing and Visualization

- Normalize input data → [0,1]
- Standardize output data → <y> = 0, <y^2>=1

```
with torch.no_grad():
    # Initialize plot
    f, ax = plt.subplots(1, 1, figsize=(4, 3))

    # Get upper and lower confidence bounds
    lower, upper = observed_pred.confidence_region()
    # Plot training data as black stars
    ax.plot(train_x.numpy(), train_y.numpy(), 'k*')
    # Plot predictive means as blue line
    ax.plot(test_x.numpy(), observed_pred.mean.numpy(), 'b')
    # Shade between the lower and upper confidence bounds
    ax.fill_between(test_x.numpy(), lower.numpy(), upper.numpy(), alpha=0.5)
    ax.set_ylim([-3, 3])
    ax.legend(['Observed Data', 'Mean', 'Confidence'])
```



Books and Resources

- The GP bible: Gaussian Processes for Machine Learning - C. Rasmussen and C. Williams. 2006
  - Free download: http://www.gaussianprocess.org/gpml/
  - Especially chapters 1,2,4,5,8

- https://distill.pub/2019/visual-exploration-gaussian-processes/

Thank you for your attention!

Questions?