Statistical Machine Learning: Part 3

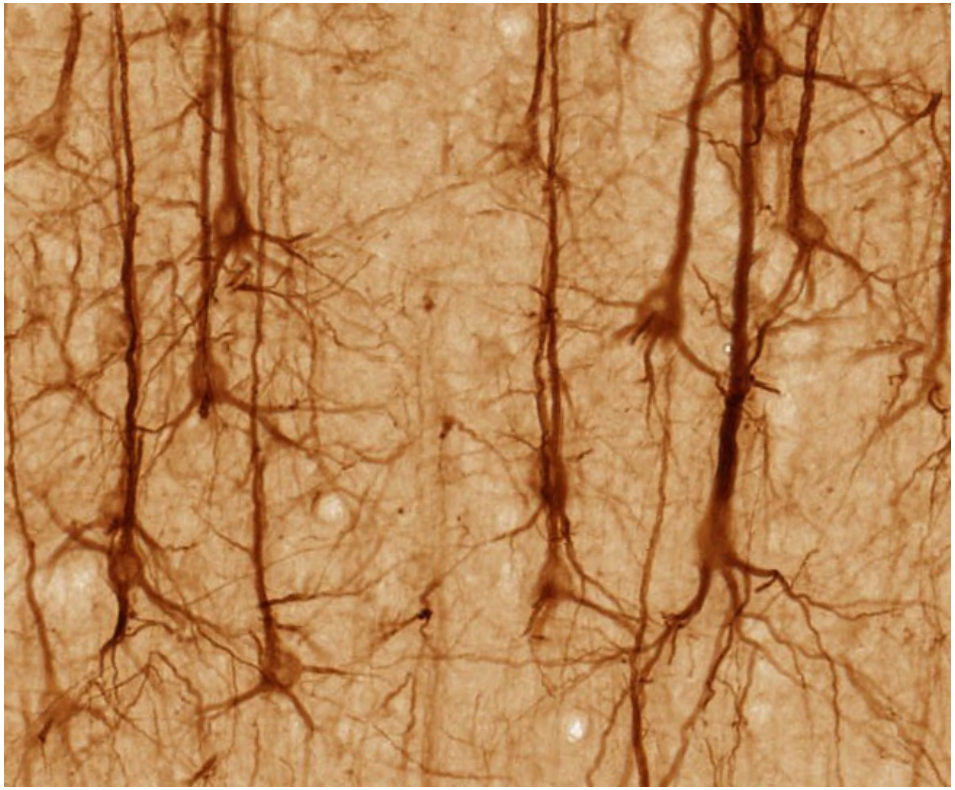
Dino Sejdinovic

February-April 2021

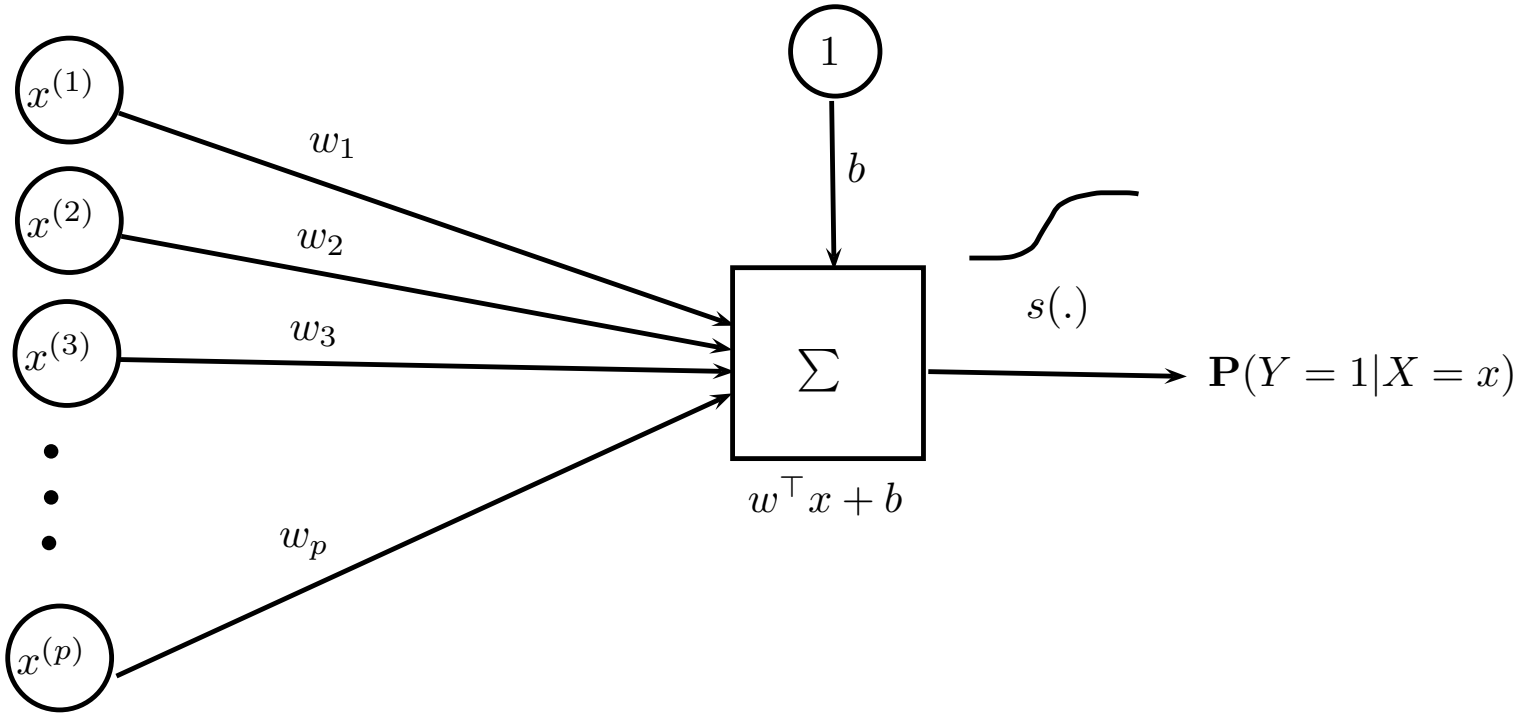
Neural Networks

Biological inspiration

- Basic computational elements: neurons.
- Receives signals from other neurons via dendrites.
- Sends processed signals via axons.
- Axon-dendrite interactions at synapses.
- $10^{10} - 10^{11}$ neurons.
- $10^{14} - 10^{15}$ synapses.

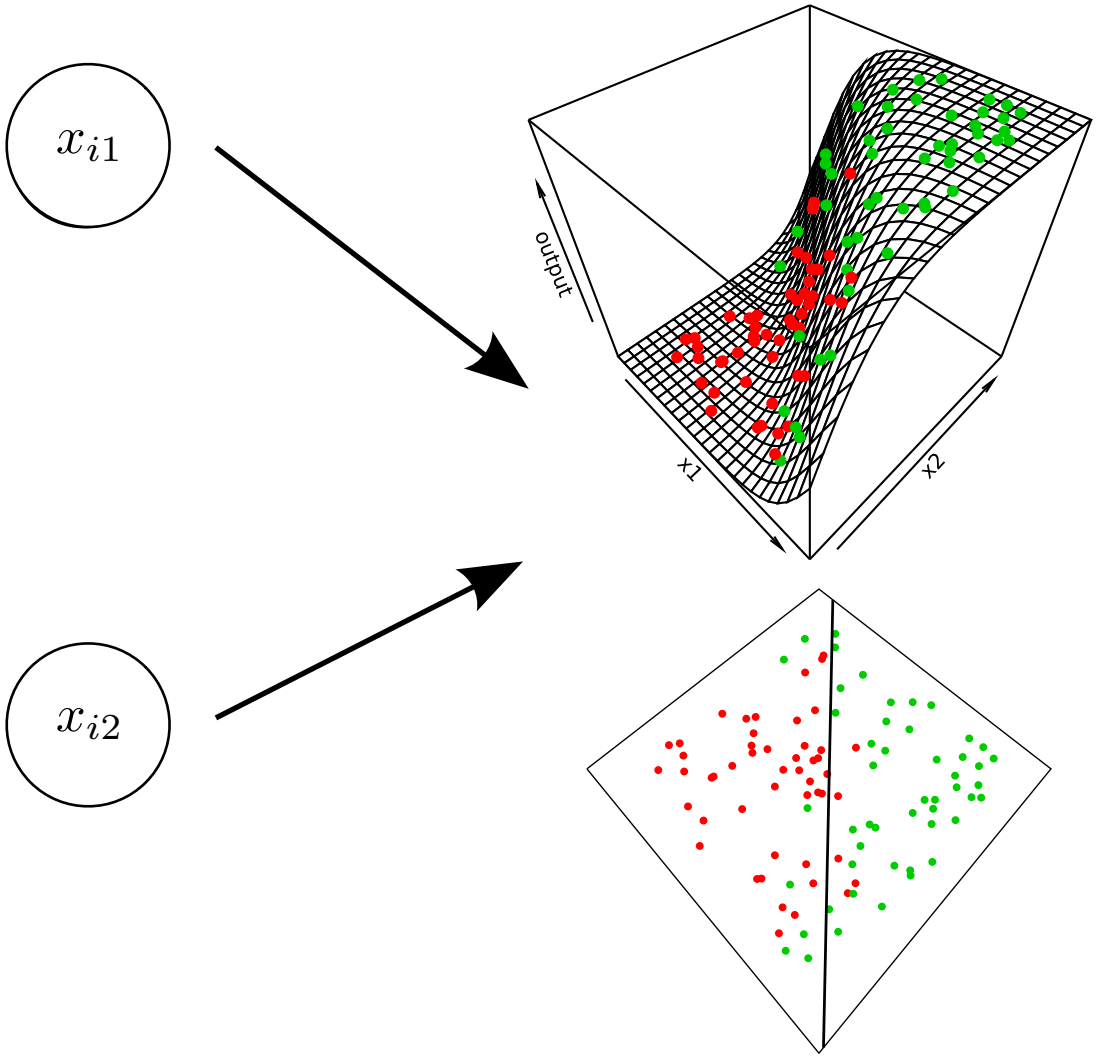


Single Neuron Classifier

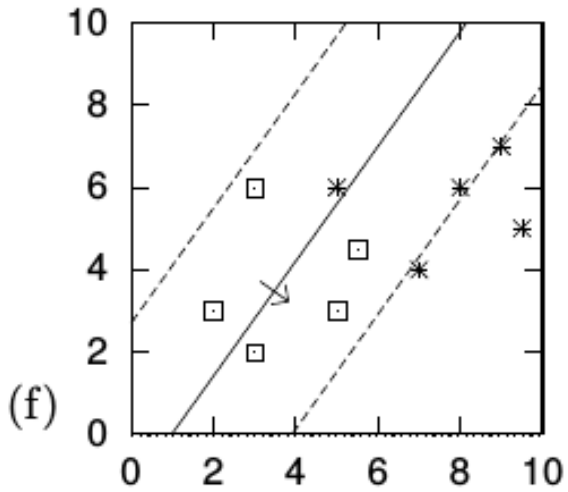


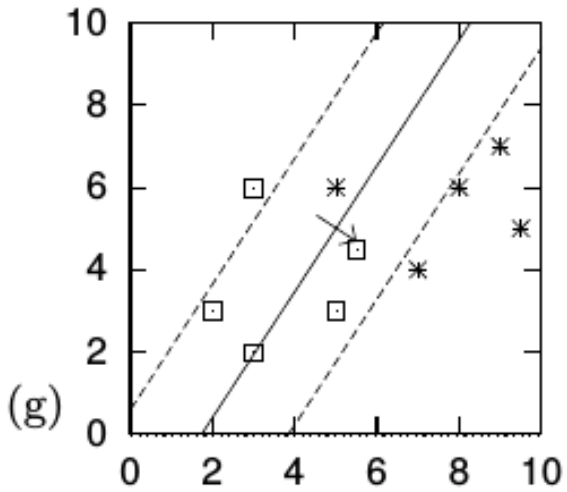
- **activation** $w^\top x + b$ (linear in **inputs** x)
- **activation/transfer function** s gives the **output/activity** (potentially nonlinear in x)
- common nonlinear activation function $s(a) = \frac{1}{1+e^{-a}}$: **logistic regression**
- learn w and b via gradient descent

Single Neuron Classifier

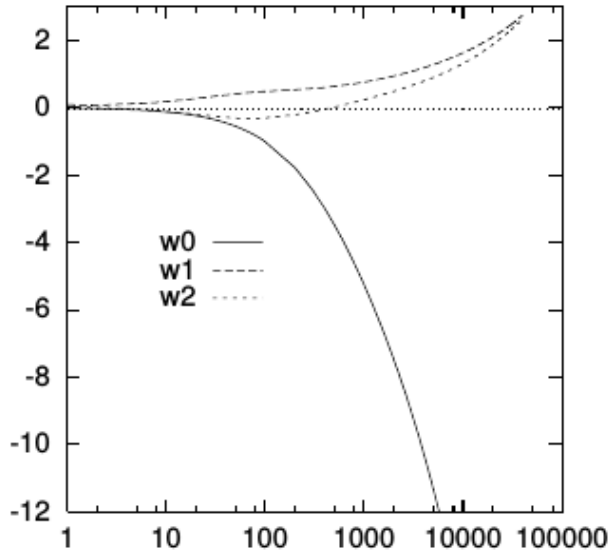


Overfitting



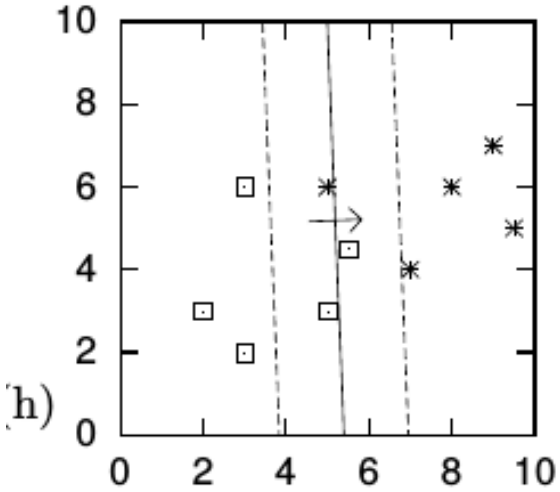


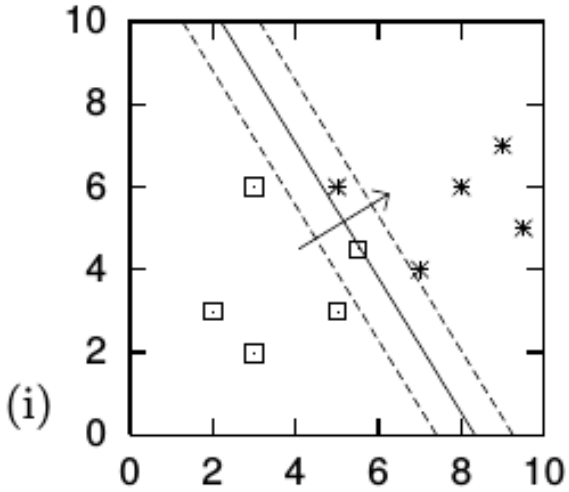
iterations 30,80



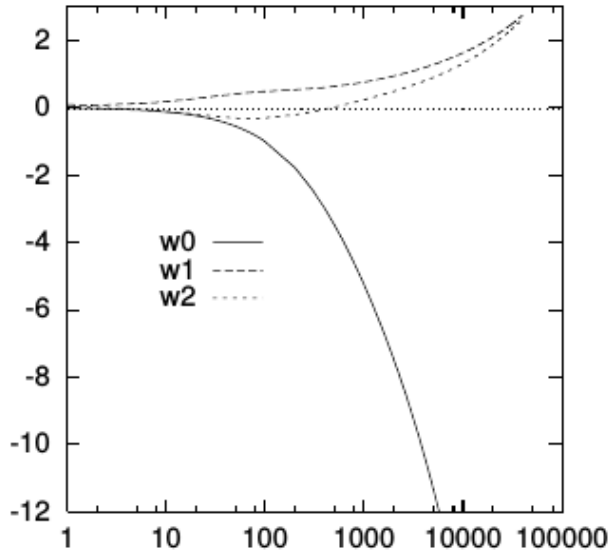
Figures from D. MacKay, **Information Theory, Inference and Learning Algorithms**

Overfitting



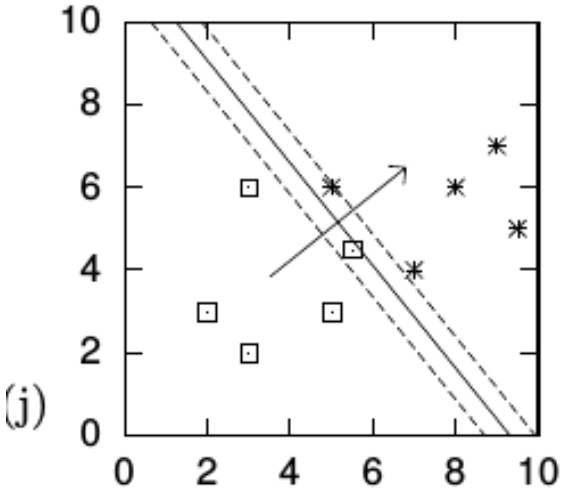


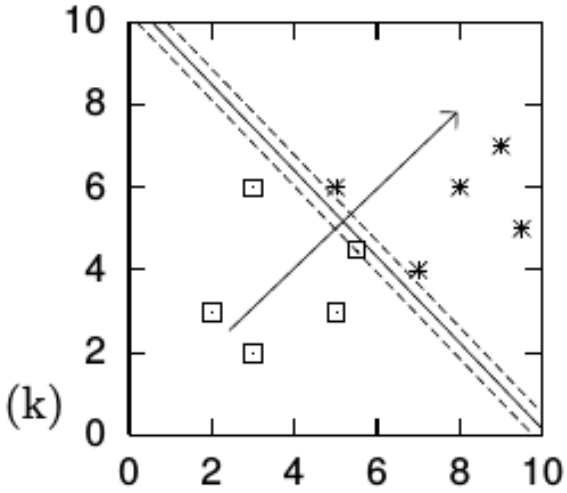
iterations 500,3000



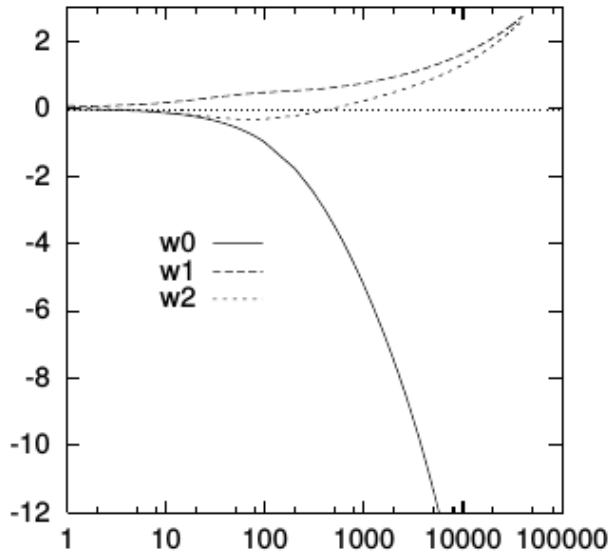
Figures from D. MacKay, **Information Theory, Inference and Learning Algorithms**

Overfitting





iterations 10000,40000



Figures from D. MacKay, **Information Theory, Inference and Learning Algorithms**

prevent overfitting by:

- **early stopping**: just halt the gradient descent
- regularization: L_2-regularization called **weight decay** in neural networks literature.

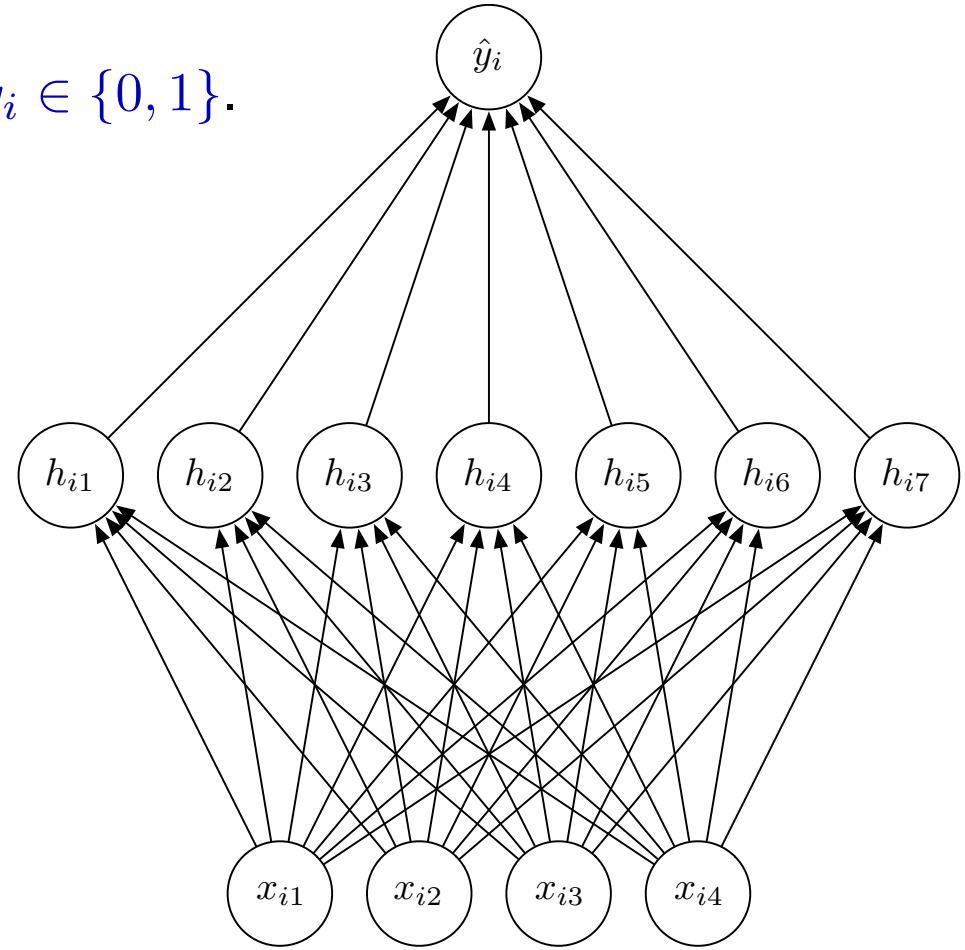
Multilayer Networks

- Data vectors $x_i \in \mathbb{R}^p$, binary labels $y_i \in \{0, 1\}$.
- **inputs** $x_{i1}, \ldots, x_{ip}$
- **output** $\hat{y}_i = \mathbb{P}(Y = 1 | X = x_i)$
- **hidden unit activities** $h_{i1}, \ldots, h_{im}$
  - Compute **hidden unit activities**:

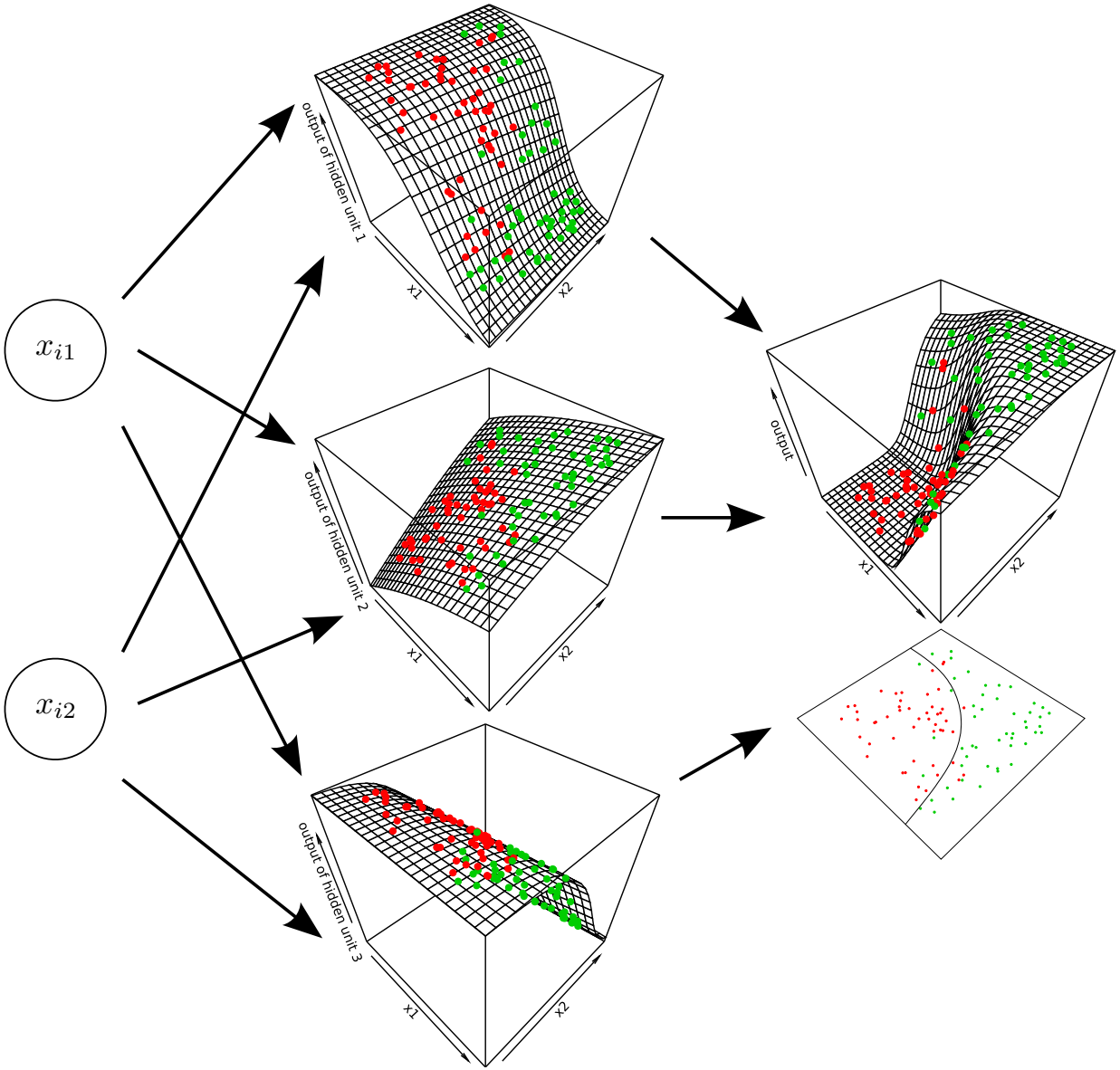
$$h_{il} = s\left(b_l^h + \sum_{j=1}^{p} w_{jl}^h x_{ij}\right)$$

  - Compute **output probability**:

$$\hat{y}_i = s\left(b^o + \sum_{l=1}^{m} w_k^o h_{il}\right)$$



Multilayer Networks



Training a Neural Network

- Objective function: L_2-regularized log-loss

$$J = -\sum_{i=1}^{n} y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i) + \frac{\lambda}{2} \left(\sum_{jl} (w_{jl}^h)^2 + \sum_{l} (w_l^o)^2 \right)$$

where

$$\hat{y}_i = s\left(b^o + \sum_{l=1}^{m} w_l^o h_{il} \right) \qquad h_{il} = s\left(b_l^h + \sum_{j=1}^{p} w_{jl}^h x_{ij} \right)$$

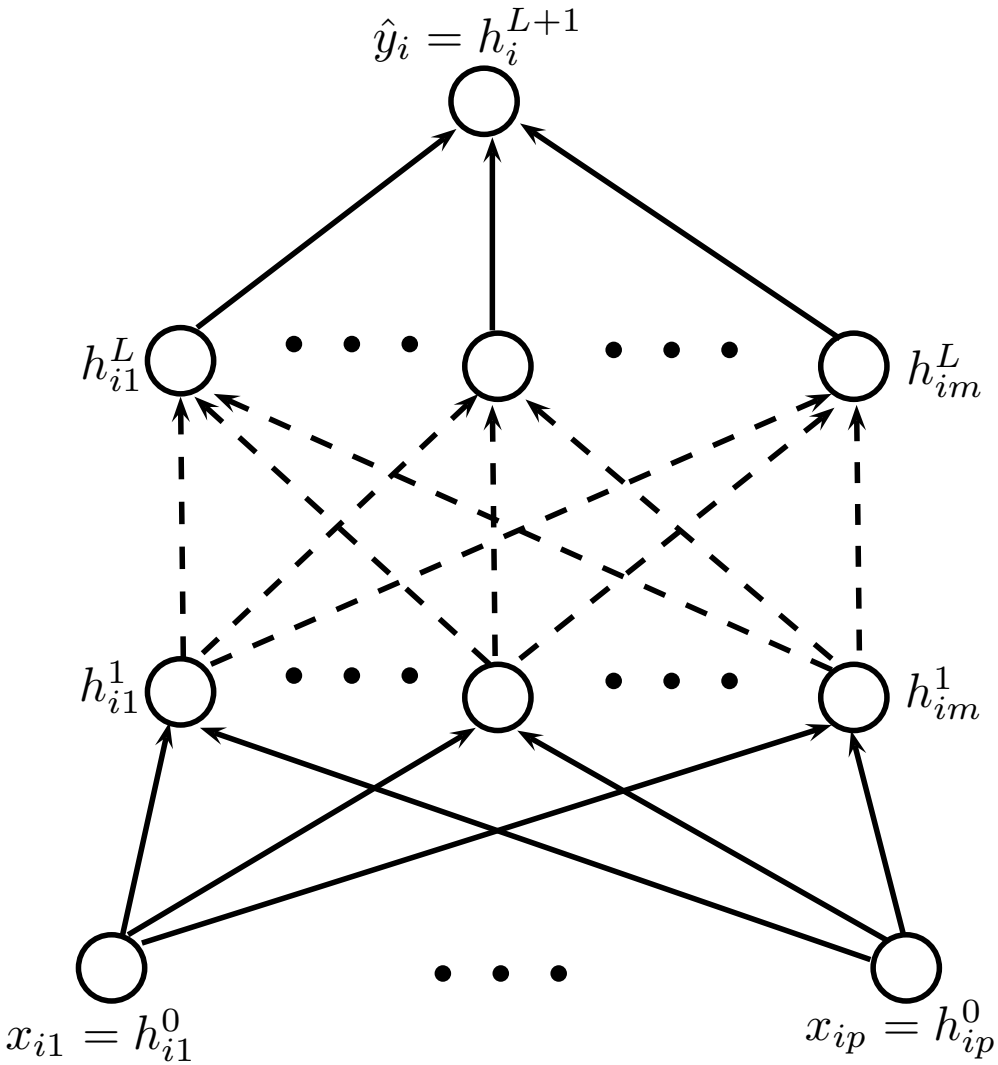
- Optimize parameters $\theta = \left\{ b^h, w^h, b^o, w^o \right\}$, where $b^h \in \mathbb{R}^m$, $w^h \in \mathbb{R}^{p \times m}$, $b^o \in \mathbb{R}$, $w^o \in \mathbb{R}^m$ with gradient descent.

$$\frac{\partial J}{\partial w_l^o} = \lambda w_l^o + \sum_{i=1}^{n} \frac{\partial J}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial w_l^o} = \lambda w_l^o + \sum_{i=1}^{n} (\hat{y}_i - y_i) h_{il},$$

$$\frac{\partial J}{\partial w_{jl}^h} = \lambda w_{jl}^h + \sum_{i=1}^{n} \frac{\partial J}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial h_{il}} \frac{\partial h_{il}}{\partial w_{jl}^h} = \lambda w_{jl}^h + \sum_{i=1}^{n} (\hat{y}_i - y_i) w_l^o h_{il} (1 - h_{il}) x_{ij}.$$

- L_2-regularization often called **weight decay**.
- Multiple hidden layers: **Backpropagation** algorithm

Multiple hidden layers



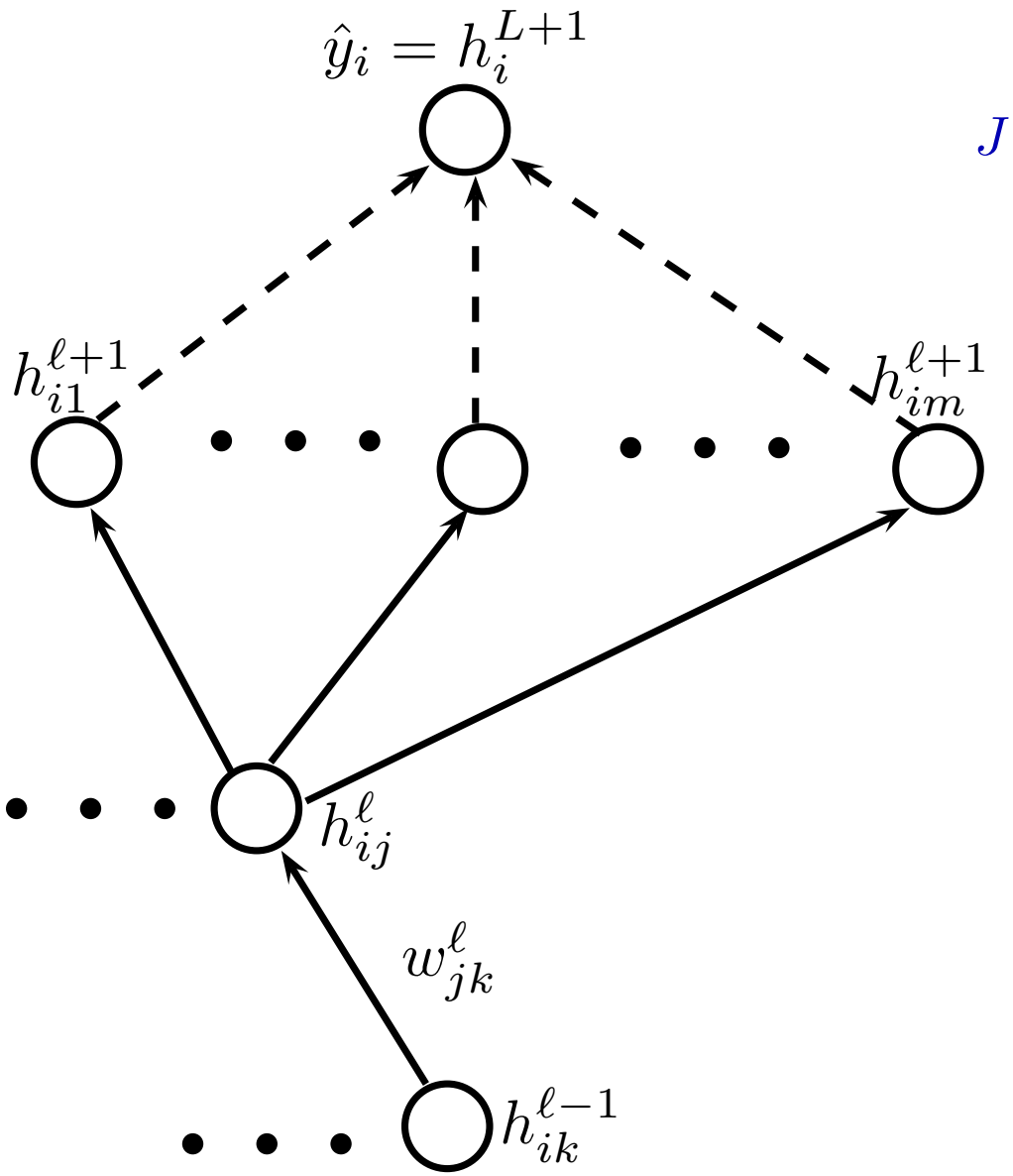
$$h_i^{\ell+1} = \underline{s}\left(W^{\ell+1} h_i^{\ell}\right)$$

- $W^{\ell+1} = \left(w_{jk}^{\ell}\right)_{jk}$: weight matrix at the $(\ell+1)$-th layer, weight w_{jk}^{ℓ} on the edge between $h_{ik}^{\ell-1}$ and h_{ij}^{ℓ}
- $\underline{s}$: entrywise (logistic) transfer function

$$\hat{y}_i = \underline{s}\left(W^{L+1}\underline{s}\left(W^{L}\left(\cdots \underline{s}\left(W^{1} x_i\right)\right)\right)\right)$$

- **Many** hidden layers can be used: they are usually thought of as forming a hierarchy from low-level to high-level features.

Backpropagation



$$J = -\sum_{i=1}^{n} y_i \log h_i^{L+1} + (1-y_i) \log(1-h_i^{L+1}$$

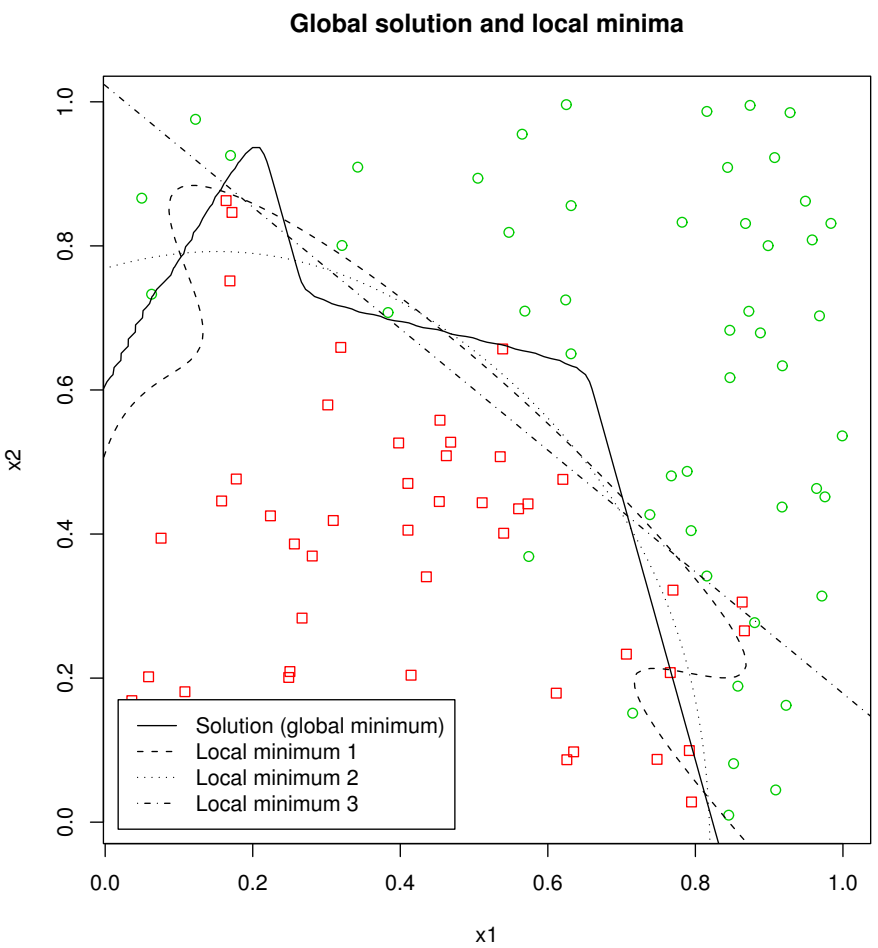
- Gradients wrt h_{ij}^{ℓ} computed by recursive applications of chain rule, and propagated through the network backwards.

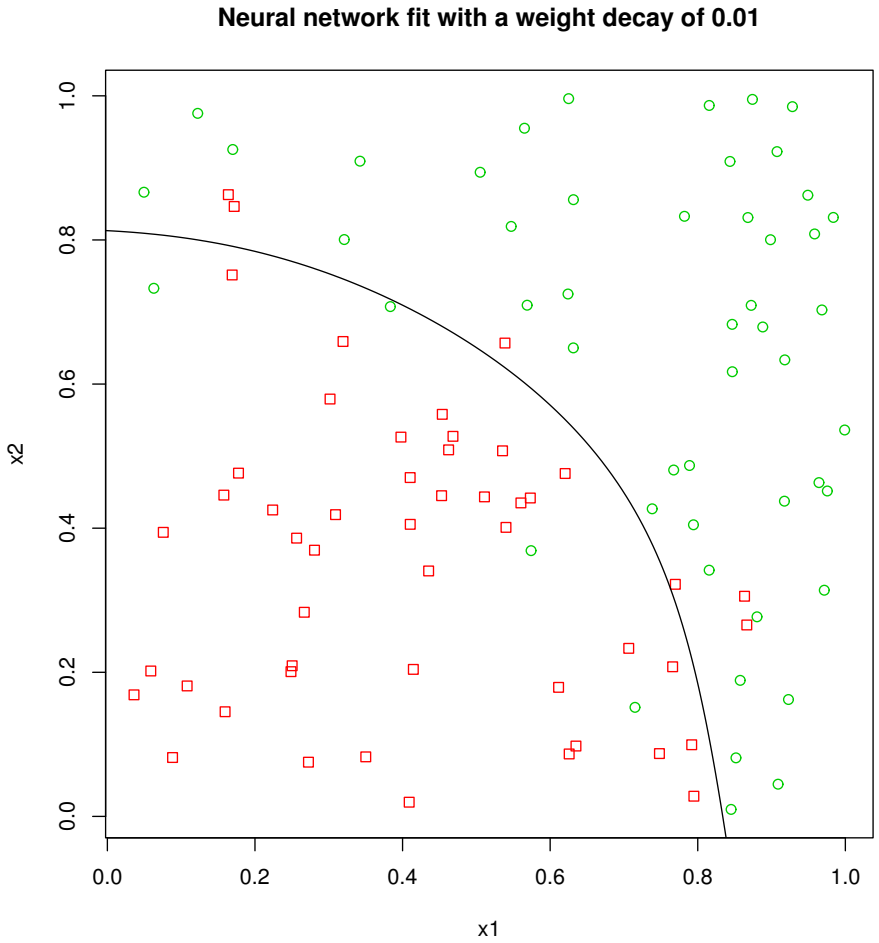
$$\frac{\partial J}{\partial h_i^{L+1}} = -\frac{y_i}{h_i^{L+1}} + \frac{1-y_i}{1-h_i^{L+1}}$$

$$\frac{\partial J}{\partial h_{ij}^{\ell}} = \sum_{r=1}^{m} \frac{\partial J}{\partial h_{ir}^{\ell+1}} \frac{\partial h_{ir}^{\ell+1}}{\partial h_{ij}^{\ell}}$$

$$\frac{\partial J}{\partial w_{jk}^{\ell}} = \sum_{i=1}^{n} \frac{\partial J}{\partial h_{ij}^{\ell}} \frac{\partial h_{ij}^{\ell}}{\partial w_{jk}^{\ell}}$$

Neural Networks





Neural Networks – Variations

- Other loss functions can be used, e.g. for regression:

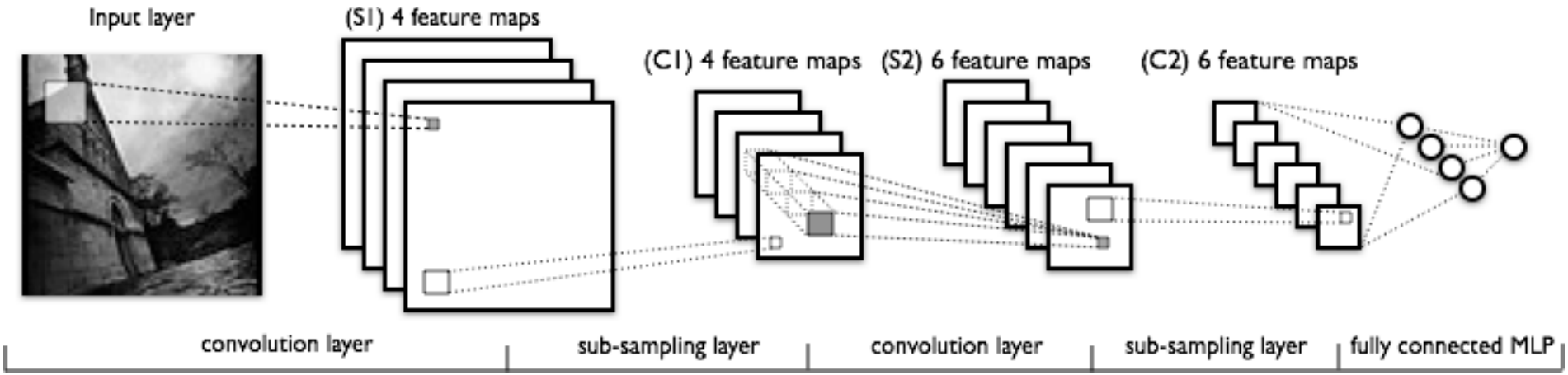
$$\sum_{i=1}^{n} |y_i - \hat{y}_i|^2$$

  For multiclass classification, use **softmax** outputs:

$$\hat{y}_{ik} = \frac{\exp(b_k^o + \sum_\ell w_{lk}^o h_{i\ell})}{\sum_{k'} \exp(b_{k'}^o + \sum_\ell w_{lk'}^o h_{i\ell})} \qquad L(y_i, \hat{y}_i) = \sum_{k=1}^{K} \mathbb{1}(y_i = k) \log \hat{y}_{ik}$$

- Other activation functions can be used:
  - **rectified linear unit (ReLU)**: $s(z) = \max(0, z)$
  - **softplus**: $s(z) = \log(1 + \exp(z))$
  - **tanh**: $s(z) = \tanh(z)$

Deep Convolutional Neural Networks



- Input is a 2D image, $X \in \mathbb{R}^{p \times q}$.
- **Convolution**: detects simple object parts or features

$$A^m = s(X * W^m) \qquad A^m_{jk} = s\left(b^m + \sum_{fg} X_{j-f,k-g} W^m_{fg}\right)$$

  Weights W^m now correspond to a **filter** to be learned - typically much smaller than the input thus encouraging sparse connectivity.
- **Pooling and Sub-sampling**: replace the output with a summary statistic of the nearby outputs, e.g. max-pooling (allows invariance to small translations in the input).

LeCun et al, Krizhevsky et al.

Dropout Training of Neural Networks

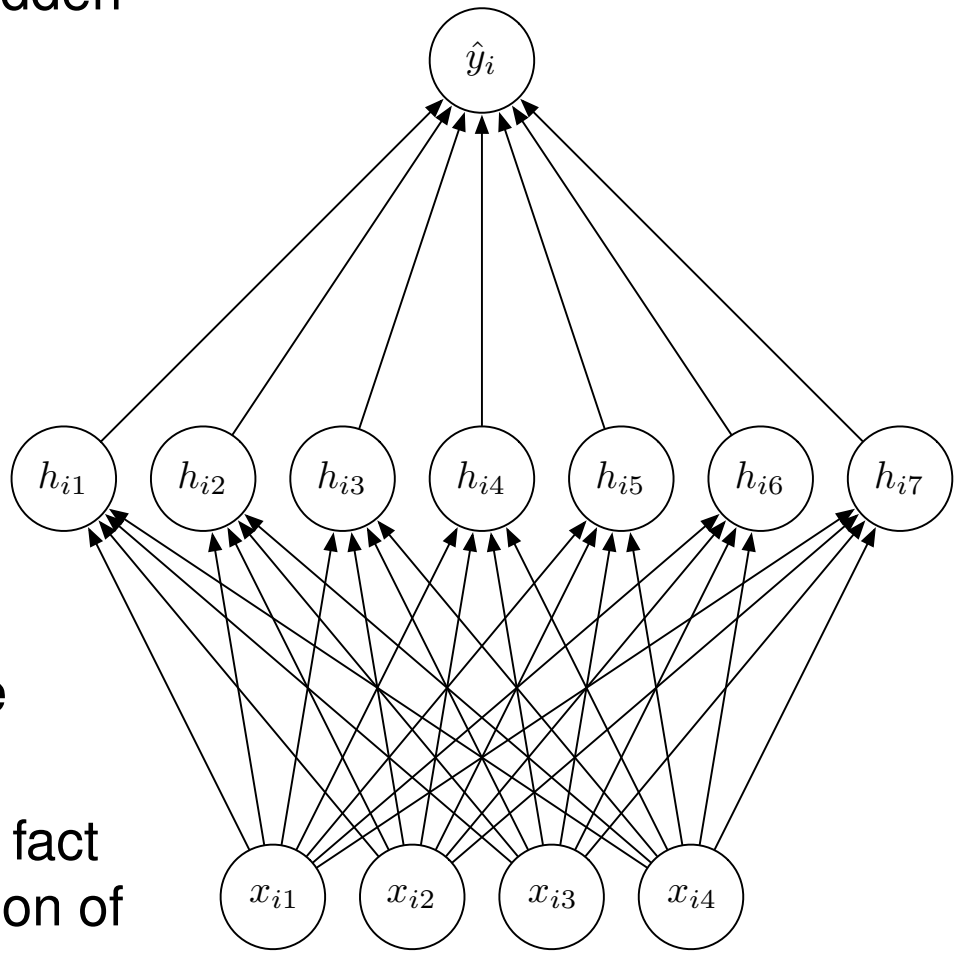
- Neural network with single layer of hidden units:
  - **Hidden unit activations**:

$$h_{ik} = s\left(b_k^h + \sum_{j=1}^{p} W_{jk}^h x_{ij}\right)$$

  - **Output probability**:

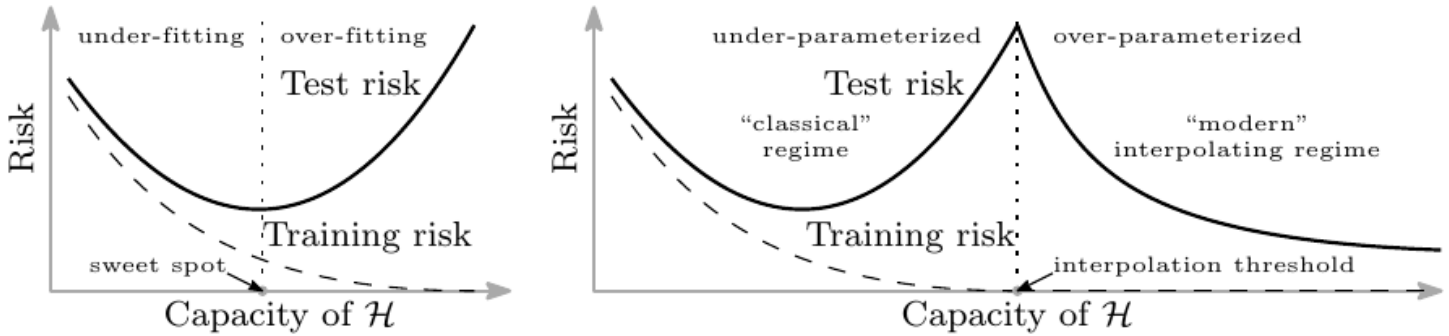
$$\hat{y}_i = s\left(b^o + \sum_{k=1}^{m} W_k^o h_{ik}\right)$$

- Large, overfitted networks often have co-adapted hidden units.
- What each hidden unit learns may in fact be useless, e.g. predicting the negation of predictions from other units.
- Can prevent co-adaptation by randomly **dropping out** units from network.



Hinton et al (2012).

Model complexity and modern "interpolating" regime



- **Left:** The classical U-shaped risk curve arising from the bias-variance trade-off.
- **Right:** The double descent risk curve, which incorporates the U-shaped risk curve (i.e., the classical regime) together with the observed behavior from using high capacity function classes (i.e., the modern interpolating regime). The predictors to the right of the interpolation threshold have zero training risk.

Belkin et al, Reconciling modern machine learning practice and the bias-variance trade-off

Neural Networks – Discussion

- Nonlinear hidden units introduce modelling flexibility.
- In contrast to user-introduced nonlinearities, features are global, and can be learned to maximize predictive performance.
- Neural networks with a single hidden layer and sufficiently many hidden units can model arbitrarily complex functions.
- Highly flexible framework, with many variations to solve different learning problems and introduce domain knowledge.
- Optimization problem is **not convex**, and objective function can have many local optima, plateaus and ridges.
- On large scale problems, often use **stochastic gradient descent**, along with a whole host of techniques for optimization, regularization, and initialization. Most of this easy to use and combine in deep learning frameworks such as PyTorch, Tensorflow, Keras.
- Explosion of interest in the field since 2012+ and **many new developments** not covered here, especially by teams of Geoffrey Hinton, Yann LeCun, Yoshua Bengio, Andrew Ng and others. See also `http://deeplearning.net/`.

Gaussian Processes

Parametric vs Nonparametric models

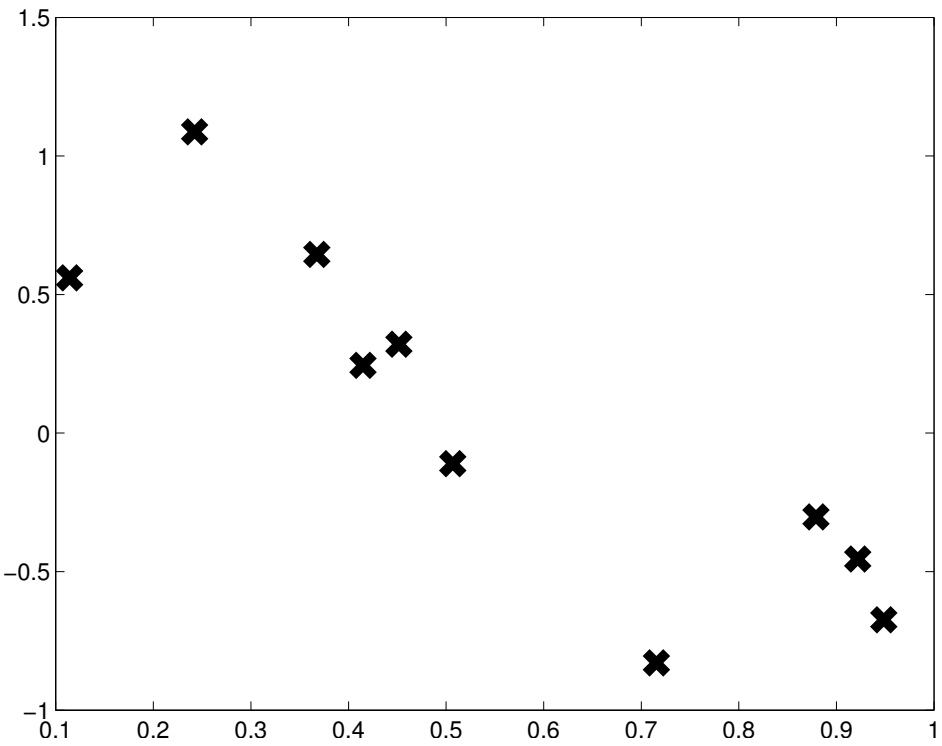
- **Parametric models** have a fixed finite number of parameters, regardless of the dataset size. In the Bayesian setting, given the parameter vector θ, the predictions are independent of the data $\mathcal{D}$.

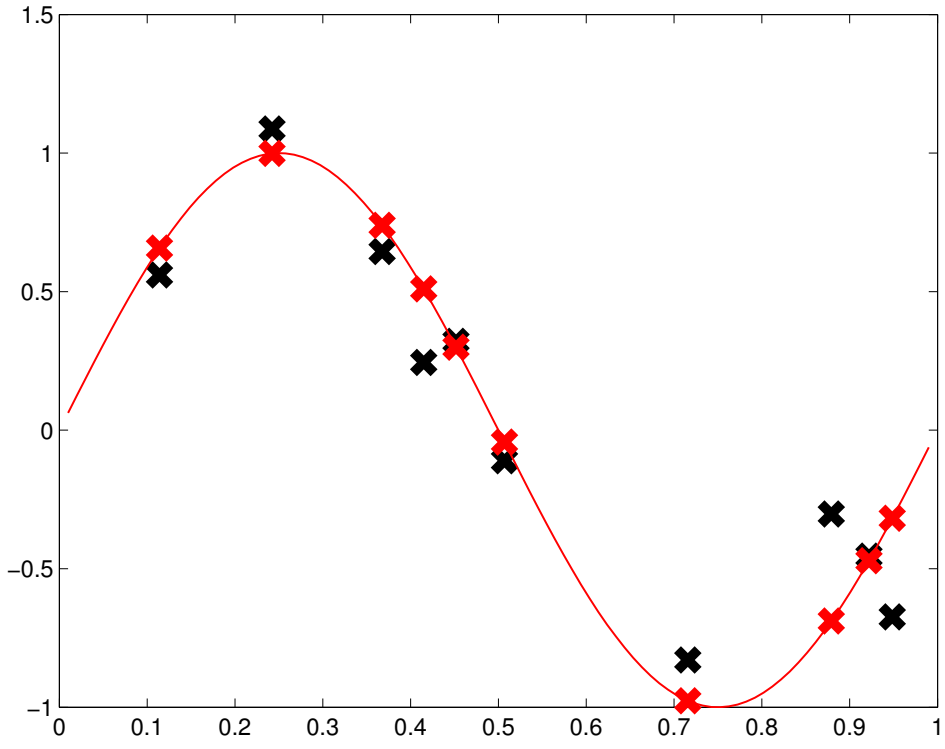
$$p(\tilde{x}, \theta|\mathcal{D}) = p(\theta|\mathcal{D})p(\tilde{x}|\theta)$$

  Parameters can be thought of as a data summary: communication channel flows from data to the predictions through the parameters.

- **Nonparametric models** allow the number of "parameters" to grow with the dataset size. Alternatively, predictions depend on the data (and the hyperparameters).

Regression





- We are given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, $x_i \in \mathbb{R}^p$, $y_i \in \mathbb{R}$.
- Regression: learn the underlying real-valued function $f(x)$.

Different Flavours of Regression

- We can model response y_i as a noisy version of the underlying function f evaluated at input x_i:

$$y_i|f(x_i) \sim \mathcal{N}(f(x_i), \sigma^2)$$

  Appropriate loss: $L(y, f(x)) = (y - f(x))^2$
- **Frequentist Parametric** approach: model f as f_θ for some parameter vector θ. Fit θ by ML / ERM with squared loss (linear regression).
- **Frequentist Nonparametric** approach: model f as the unknown parameter taking values in an infinite-dimensional space of functions. Fit f by regularized ML / ERM with squared loss (kernel ridge regression)
- **Bayesian Parametric** approach: model f as f_θ for some parameter vector θ. Put a prior on θ and compute a posterior $p(\theta|\mathcal{D})$ (Bayesian linear regression).
- **Bayesian Nonparametric** approach: treat f as the random variable taking values in an infinite-dimensional space of functions. Put a prior over functions $f \in \mathcal{F}$, and compute a posterior $p(f|\mathcal{D})$ (Gaussian Process regression).

- Just work with the function values at the inputs $\mathbf{f} = (f(x_1), \ldots, f(x_n))^\top$
- What properties of the function can we incorporate?
  - Multivariate normal prior on $\mathbf{f}$:

$$\mathbf{f} \sim \mathcal{N}(0, \mathbf{K})$$

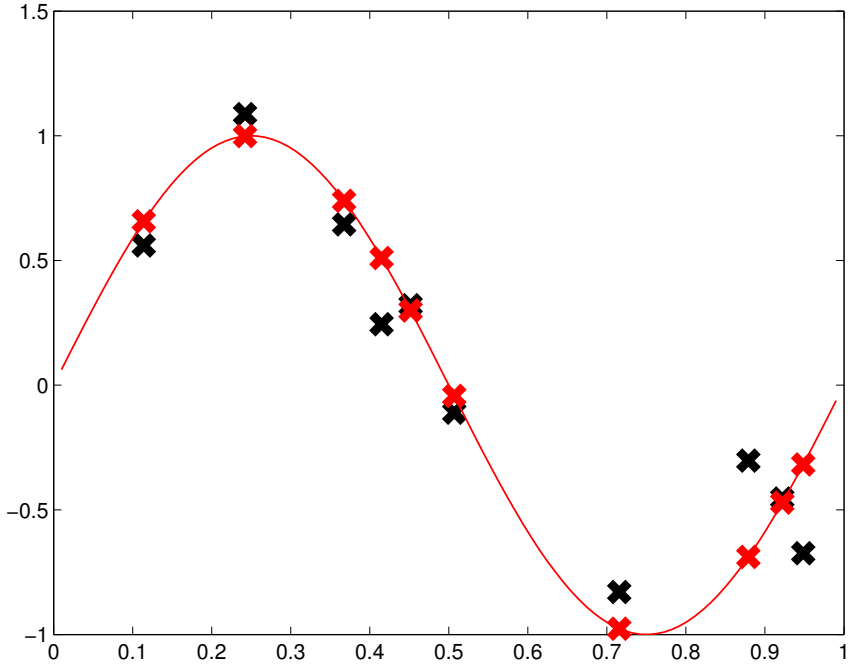
  - Use a kernel function k to define $\mathbf{K}$:

$$\mathbf{K}_{ij} = k(x_i, x_j)$$

  - Expect regression functions to be smooth: If x and x' are close by, then $f(x)$ and $f(x')$ have similar values, i.e. strongly correlated.

$$\begin{pmatrix} f(x) \\ f(x') \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} k(x,x) & k(x,x') \\ k(x',x) & k(x',x') \end{pmatrix} \right)$$

The prior $p(\mathbf{f})$ encodes our prior knowledge about the function.



- Model:

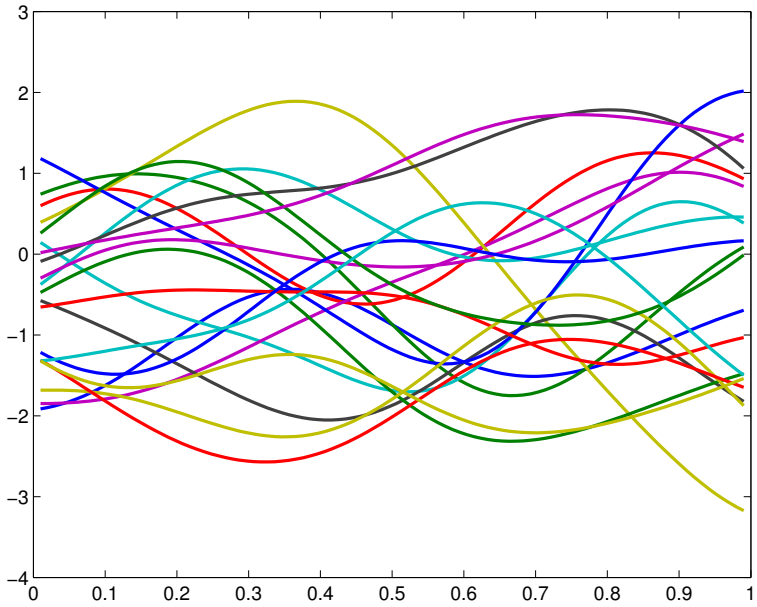
$$\mathbf{f} \sim \mathcal{N}(0, \mathbf{K})$$
$$y_i | f_i \sim \mathcal{N}(f_i, \sigma^2)$$

Gaussian Processes

- What does a multivariate normal prior mean?
- Imagine $\mathbf{x}$ forms an infinitesimally dense grid of data space. Simulate prior draws

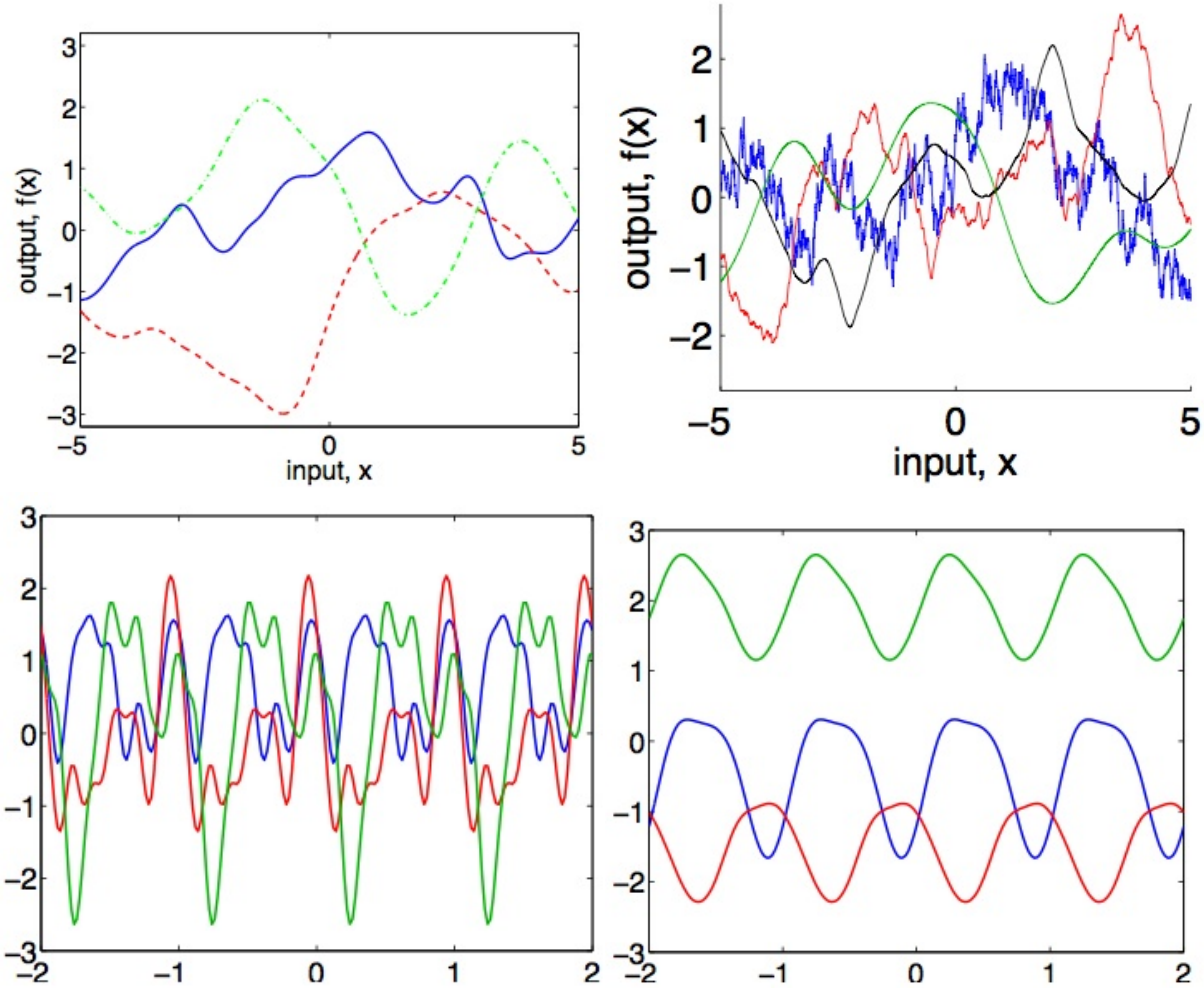
$$\mathbf{f} \sim \mathcal{N}(0, \mathbf{K})$$

  Plot f_i vs x_i for $i = 1, \ldots, n$.
- The corresponding prior over functions is called a **Gaussian Process** (GP): any finite number of evaluations of which follow a Gaussian distribution.



Gaussian Processes

- Different kernels lead to different function characteristics.



Carl Rasmussen. Tutorial on Gaussian Processes at NIPS 2006.

Gaussian Processes

$$\mathbf{f}|\mathbf{x} \sim \mathcal{N}(0, \mathbf{K})$$
$$\mathbf{y}|\mathbf{f} \sim \mathcal{N}(\mathbf{f}, \sigma^2 I)$$

- Posterior distribution:

$$\mathbf{f}|\mathbf{y} \sim \mathcal{N}(\mathbf{K}(\mathbf{K} + \sigma^2 I)^{-1}\mathbf{y}, \mathbf{K} - \mathbf{K}(\mathbf{K} + \sigma^2 I)^{-1}\mathbf{K})$$

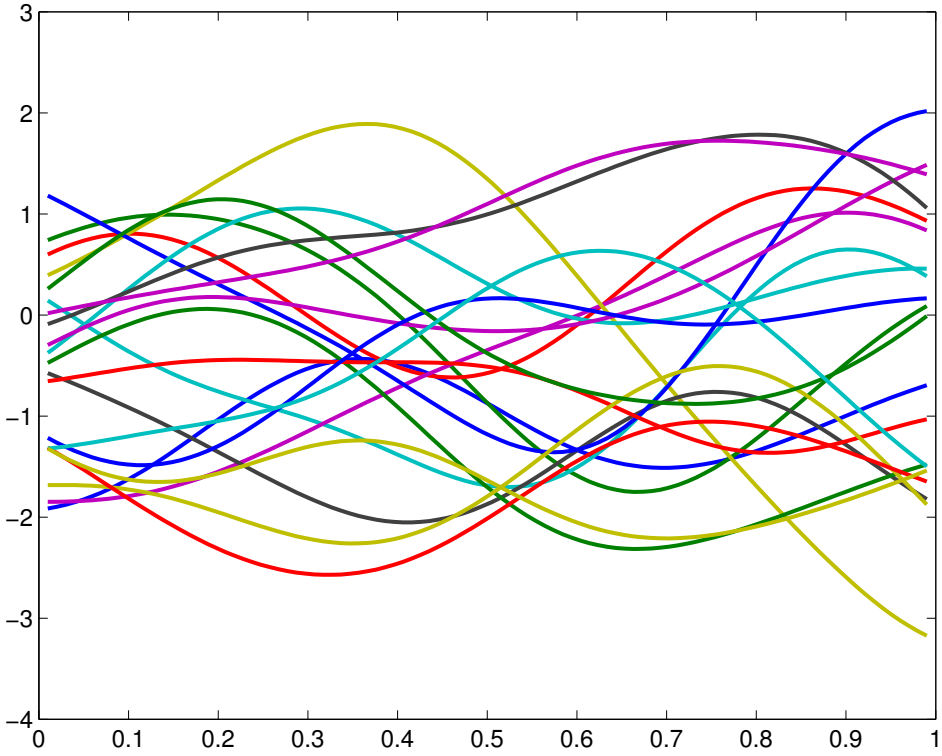
- Posterior predictive distribution: Suppose $\mathbf{x}'$ is a test set. We can extend our model to include the function values $\mathbf{f}'$ at the test set:

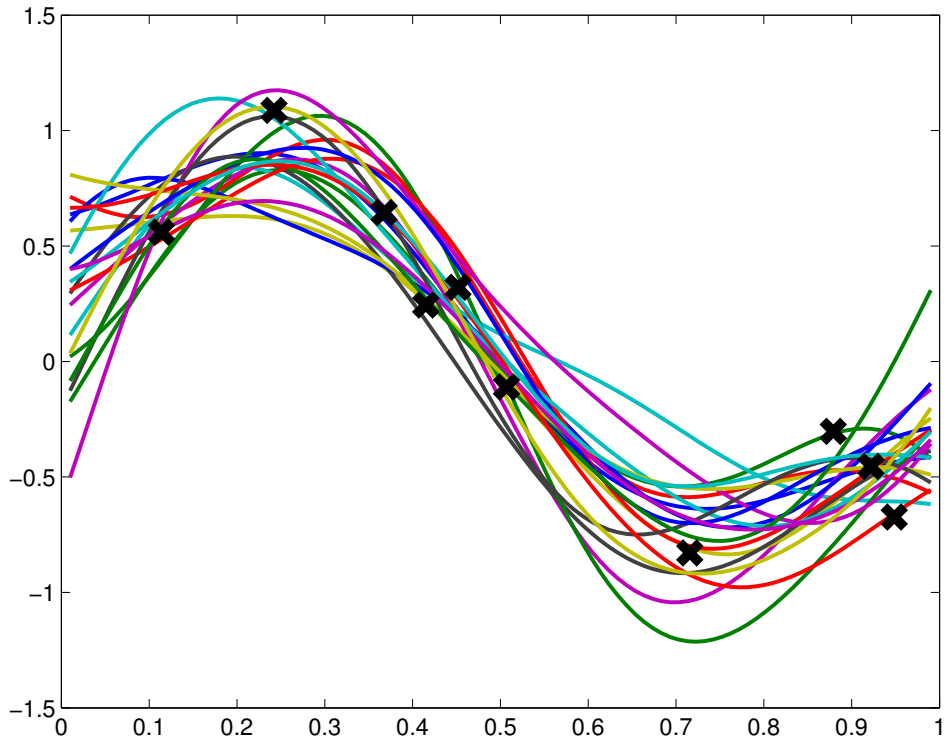
$$\begin{pmatrix} \mathbf{f} \\ \mathbf{f}' \end{pmatrix} |\mathbf{x}, \mathbf{x}' \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \mathbf{K}_{\mathbf{xx}} & \mathbf{K}_{\mathbf{xx}'} \\ \mathbf{K}_{\mathbf{x}'\mathbf{x}} & \mathbf{K}_{\mathbf{x}'\mathbf{x}'} \end{pmatrix} \right)$$
$$\mathbf{y}|\mathbf{f} \sim \mathcal{N}(\mathbf{f}, \sigma^2 I)$$

  where $\mathbf{K}_{\mathbf{xx}'}$ is matrix with (i, j)-th entry $k(x_i, x'_j)$.
- Some manipulation of multivariate normals gives:

$$\mathbf{f}'|\mathbf{y} \sim \mathcal{N}\left(\mathbf{K}_{\mathbf{x}'\mathbf{x}}(\mathbf{K}_{\mathbf{xx}} + \sigma^2 I)^{-1}\mathbf{y}, \mathbf{K}_{\mathbf{x}'\mathbf{x}'} - \mathbf{K}_{\mathbf{x}'\mathbf{x}}(\mathbf{K}_{\mathbf{xx}} + \sigma^2 I)^{-1}\mathbf{K}_{\mathbf{xx}'}\right)$$

Gaussian Processes





GP regression demo: http://www.tmpl.fi/gp/

GP regression and Kernel Ridge Regression

If KRR and GPR use the same kernel and if the regularization parameter λ equals the noise variance σ^2, KRR estimate of the function coincides with the GPR posterior mean/mode. Indeed, recall that in KRR we are solving empirical risk minimisation

$$\min_{f \in \mathcal{H}_k} \sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2 + \sigma^2 \left\|f\right\|_{\mathcal{H}_k}^2 ,$$

and are fitting a function of the form $f(x) = \sum_{i=1}^{n} \alpha_i k\left(\cdot, x_i\right)$. Closed form solution is given by $\alpha = \left(\mathbf{K_{xx}} + \sigma^2 I\right)^{-1} \mathbf{y}$. But then if we wish to predict function values at a new set $\mathbf{x}' = \{x'_j\}_{j=1}^{m}$ of input vectors, we have

$$f(x'_j) = \sum_{i=1}^{n} \alpha_i k\left(x'_j, x_i\right) = \left[k(x'_j, x_1), \ldots, k(x'_j, x_n)\right] \left(\mathbf{K_{xx}} + \sigma^2 I\right)^{-1} \mathbf{y},$$

and $\left[k(x'_j, x_1), \ldots, k(x'_j, x_n)\right]$ is the j-th row of $\mathbf{K_{x'x}}$.
More generally, GP posterior mode for any likelihood model lies in the RKHS (essentially the same proof as the representer theorem).

Hyperparameters: Maximum marginal likelihood

Marginal likelihood of the hyperparameter vector $\theta = (\nu, \sigma^2)$ (ν: kernel parameters, σ^2: noise in the observation model)

$$p(\mathbf{y}|\theta) = \int p(\mathbf{y}|\mathbf{f}, \theta) p(\mathbf{f}|\theta) d\mathbf{f} = \mathcal{N}\left(\mathbf{y}; 0, \mathbf{K}_\nu + \sigma^2 I\right).$$

Writing $\mathbf{K}_{\theta+} = \mathbf{K}_\nu + \sigma^2 I$, marginal log-likelihood is

$$\log p(\mathbf{y}|\theta) = -\frac{1}{2} \log |\mathbf{K}_{\theta+}| - \frac{1}{2} \mathbf{y}^\top \mathbf{K}_{\theta+}^{-1} \mathbf{y} - \frac{n}{2} \log(2\pi). \tag{1}$$

Typically a nonconvex function of θ.

Hyperparameters: Bayesian treatment

Place a prior $p(\theta)$ on θ and draw samples $\{\theta_j\}$ from the posterior

$$p(\theta|\mathbf{y}) \quad \propto \quad p(\theta)p(\mathbf{y}|\theta) = p(\theta)\int p(\mathbf{y}|\mathbf{f},\theta)p(\mathbf{f}|\theta)d\mathbf{f}.$$

Integrate uncertainty over hyperparameters into predictions:

$$\begin{aligned} p(\mathbf{f}'|\mathbf{y}) &= \int p(\mathbf{f}'|\mathbf{y},\theta)p(\theta|\mathbf{y})d\theta \\ &\approx \sum_j p(\mathbf{f}'|\mathbf{y},\theta_j). \end{aligned}$$

GP with a logistic link

Consider the binary classification model with classes -1 and $+1$. Need to map Gaussian process into $(0, 1)$ with a nonlinear activation/link function, e.g.

$$p(y_i = +1|f(x_i)) = \sigma(f(x_i)) = \frac{1}{1 + e^{-f(x_i)}}. \tag{2}$$

Non-conjugate so exact posterior inference intractable.

Laplace approximation

Find MAP $\hat{\mathbf{f}}^{\text{MAP}}$ by maximizing

$$\begin{aligned}\log p(\mathbf{f}|\mathbf{y}) &= \text{const} + \log p(\mathbf{f}) + \log p(\mathbf{y}|\mathbf{f}) \\ &= \text{const} - \frac{1}{2}\mathbf{f}^\top \mathbf{K}^{-1}\mathbf{f} + \sum_{i=1}^{n} \log \sigma(y_i f(x_i)).\end{aligned}$$

Gradient:

$$\frac{\partial \log p(\mathbf{f}|\mathbf{y})}{\partial \mathbf{f}} = -\mathbf{K}^{-1}\mathbf{f} + \mathbf{g}_\mathbf{f}$$

where the gradient of the likelihood is $\mathbf{g}_\mathbf{f} = \frac{\partial \log p(\mathbf{y}|\mathbf{f})}{\partial \mathbf{f}}$ with $[\mathbf{g}_\mathbf{f}]_i = \frac{\partial \log p(\mathbf{y}|\mathbf{f})}{\partial f_i} = \sigma(-y_i f(x_i))y_i$.

Laplace approximation

Hessian:

$$\frac{\partial^2 \log p(\mathbf{f}|\mathbf{y})}{\partial \mathbf{f} \partial \mathbf{f}^\top} = -\mathbf{K}^{-1} - \mathbf{D}_\mathbf{f},$$

where $\mathbf{D}_\mathbf{f} = -\frac{\partial^2 \log p(\mathbf{y}|\mathbf{f})}{\partial \mathbf{f} \partial \mathbf{f}^\top}$ is the negative Hessian of the log-likelihood, which is an $n \times n$ diagonal matrix, $(\mathbf{D}_\mathbf{f})_{ii} = \sigma(f(x_i))\sigma(-f(x_i)) \geq 0$.
Approximation to the posterior of $\mathbf{f}$:

$$\tilde{p}(\mathbf{f}|\mathbf{y}) \quad = \quad \mathcal{N}\left(\mathbf{f} \,\middle|\, \hat{\mathbf{f}}^{\text{MAP}}, \left(\mathbf{K}^{-1} + \mathbf{D}_{\hat{\mathbf{f}}^{\text{MAP}}}\right)^{-1}\right).$$

Approximation to the predictive posterior:

$$\tilde{p}(\mathbf{f}'|\mathbf{y}) = \mathcal{N}\left(\mathbf{f}' \,|\, \mathbf{K}_{\mathbf{x}'\mathbf{x}}\mathbf{K}_{\mathbf{x}\mathbf{x}}^{-1}\hat{\mathbf{f}}^{\text{MAP}}, \mathbf{K}_{\mathbf{x}'\mathbf{x}'} - \mathbf{K}_{\mathbf{x}'\mathbf{x}}\left(\mathbf{K}_{\mathbf{x}\mathbf{x}} + \mathbf{D}_{\hat{\mathbf{f}}^{\text{MAP}}}^{-1}\right)^{-1}\mathbf{K}_{\mathbf{x}\mathbf{x}'}\right). \quad (3)$$

Same mean as the plug-in predictive $p\left(\mathbf{f}'|\hat{\mathbf{f}}^{\text{MAP}}\right)$ but the plug-in underestimates the variance.

General non-Gaussian observation models

Consider function values $\mathbf{f} = (f(x_1), \ldots, f(x_n))^\top$ at a set of inputs, and observations $\mathbf{y} = (y_1, \ldots, y_n)$, with a general observation model

$$\mathbf{f} \sim \mathcal{N}(0, \mathbf{K})$$

$$\mathbf{y}|\mathbf{f} \sim p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^{n} p\left(y_i|f(x_i)\right).$$

- Posterior distribution $p(\mathbf{f}|\mathbf{y})$ is no longer tractable.
- **Variational approximation**: write $q(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mu, \Sigma)$ and learn μ, Σ by optimizing the lower bound on the marginal likelihood (ELBO):

$$\mathcal{L}(\mu, \Sigma) = \mathbb{E}_q \log p(\mathbf{y}|\mathbf{f}) - \underbrace{KL\left(q(\mathbf{f})||p(\mathbf{f})\right)}_{\text{tractable}}$$

- **Inducing points** / **landmarks**: often coupled with a scalable GP approximation, taking $m \ll n$ inducing inputs $z_1, \ldots, z_m$ and respective values $\mathbf{u} = (f(z_1), \ldots, f(z_m))$, with a joint variational posterior $q(\mathbf{f}, \mathbf{u}) = p(\mathbf{f}|\mathbf{u})q(\mathbf{u})$, so that only variational parameters of $q(\mathbf{u})$ need to be inferred.

Large Scale Approximations

Kernel methods at scale

- Expressivity of kernel methods (rich, often infinite-dimensional hypothesis spaces) comes with a cost that scales at least quadratically in the number of observations n (due to needing to compute, store and often invert the Gram matrix)! We arrived at this by trying to avoid paying the cost in the dimension of the hypothesis space (e.g., for order d polynomial kernels, scales as $\binom{p+d}{d}$, and infinite for many kernels).
- But now we have to pay in terms of n which is problematic when we have a lot of observations (and this is exactly when we want to use a rich expressive model with a high-dimensional hypothesis class!)
- Scaling up kernel methods is a very active research area
  [Sonnenburg et al, 2006; Rahimi & Recht 2007; Le, Sarlos & Smola, 2013; Wilson et al, 2014; Dai et al, 2014; Sriperumbudur & Szabo, 2015].
- Main idea: study the desired hypothesis space and scale its dimension down - then undo the kernel trick!
- So we went the full circle (!?)
  explicit basis functions $\rightarrow$ implicit basis functions $\rightarrow$ explicit basis functions

Random Fourier features: Inverse Kernel Trick

Bochner's representation: any positive definite **translation-invariant** kernel on $\mathbb{R}^p$ can be written as

$$\begin{aligned} k(x,y) &= \int_{\mathbb{R}^p} \exp\left(i\omega^\top(x-y)\right) d\Lambda(\omega) \\ &= \int_{\mathbb{R}^p} \left\{\cos\left(\omega^\top x\right)\cos\left(\omega^\top y\right) + \sin\left(\omega^\top x\right)\sin\left(\omega^\top y\right)\right\} d\Lambda(\omega) \end{aligned}$$

for some positive measure (w.l.o.g. a probability distribution) Λ.

- Sample m frequencies $\{\omega_j\} \sim \Lambda$ and use a Monte Carlo estimator of the kernel function instead [Rahimi & Recht, 2007]:

$$\begin{aligned} \hat{k}(x,y) &= \frac{1}{m}\sum_{j=1}^{m}\left\{\cos\left(\omega_j^\top x\right)\cos\left(\omega_j^\top y\right) + \sin\left(\omega_j^\top x\right)\sin\left(\omega_j^\top y\right)\right\} \\ &= \langle\varphi_\omega(x), \varphi_\omega(y)\rangle_{\mathbb{R}^{2m}}, \end{aligned}$$

  with an explicit set of features $x \mapsto \frac{1}{\sqrt{m}}\left[\cos\left(\omega_1^\top x\right), \sin\left(\omega_1^\top x\right), \ldots\right]$.

- How fast does m need to grow with n? Sublinear for regression [Bach, 2015]

Inducing variables / Nyström

- Directly approximate the $n \times n$ Gram matrix K_{XX} of a set of inputs $\{x_i\}_{i=1}^n$ with

$$\hat{K}_{XX} = K_{XZ} K_{ZZ}^{-1} K_{ZX}$$

  where K_{ZZ} is $m \times m$ on "inducing" inputs $\{z_i\}_{i=1}^m$.
- Corresponds to explicit feature representation $x \mapsto K_{xZ} K_{ZZ}^{-1/2}$.
- Surrogate kernel $\hat{k}(x, x') = \langle k_|(\cdot, x), k_|(\cdot, x') \rangle$, where $k_|(\cdot, x)$ is a projection of $k(\cdot, x)$ to span $\{k(\cdot, z_1), \ldots, k(\cdot, z_m)\}$
- Often used in regression with Gaussian processes: with the use of Sherman-Morrison-Woodbury identity, reduces $O(n^3)$ cost to $O(nm^2)$.

  [Quiñonero-Candela and Rasmussen, 2005, Snelson and Ghahramani, 2006]
- m can grow much slower than n in regression without sacrificing performance [Rudi, Camoriano & Rosasco, 2015].

Bayesian Optimization

Towards End-to-End Learning

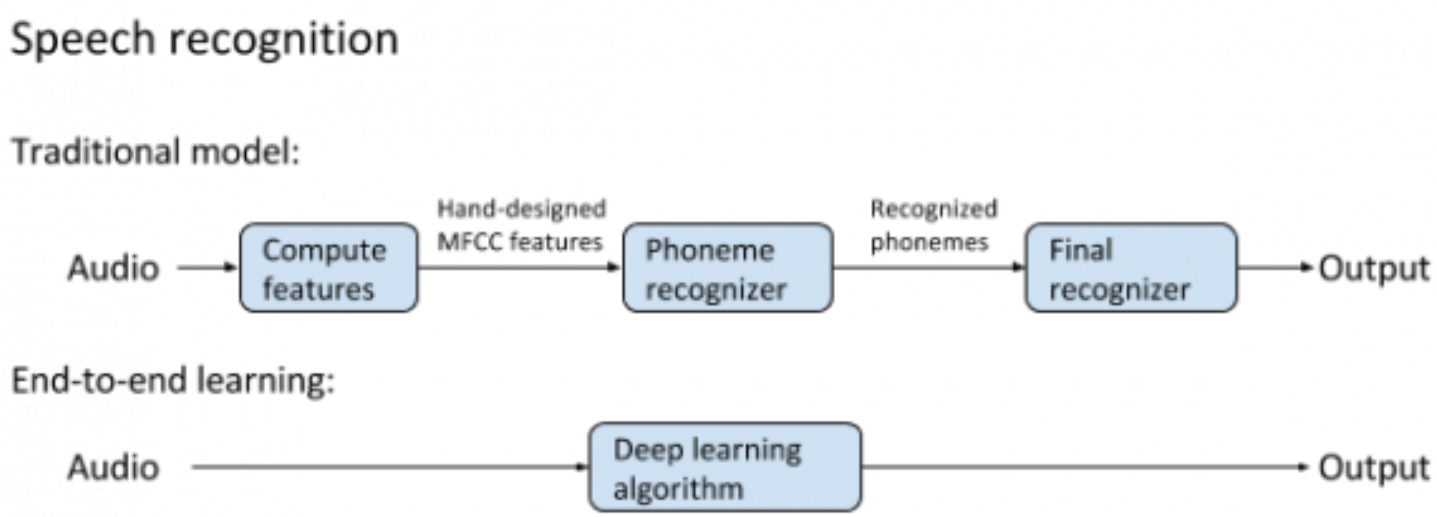
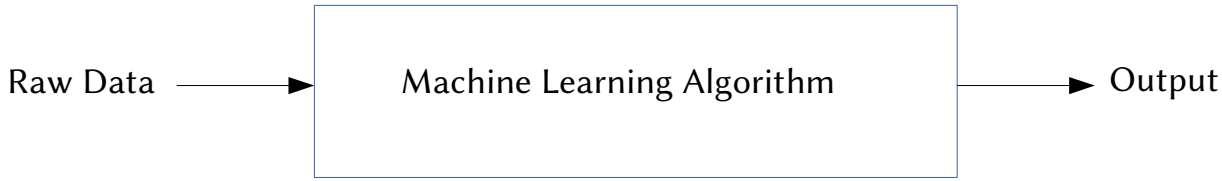
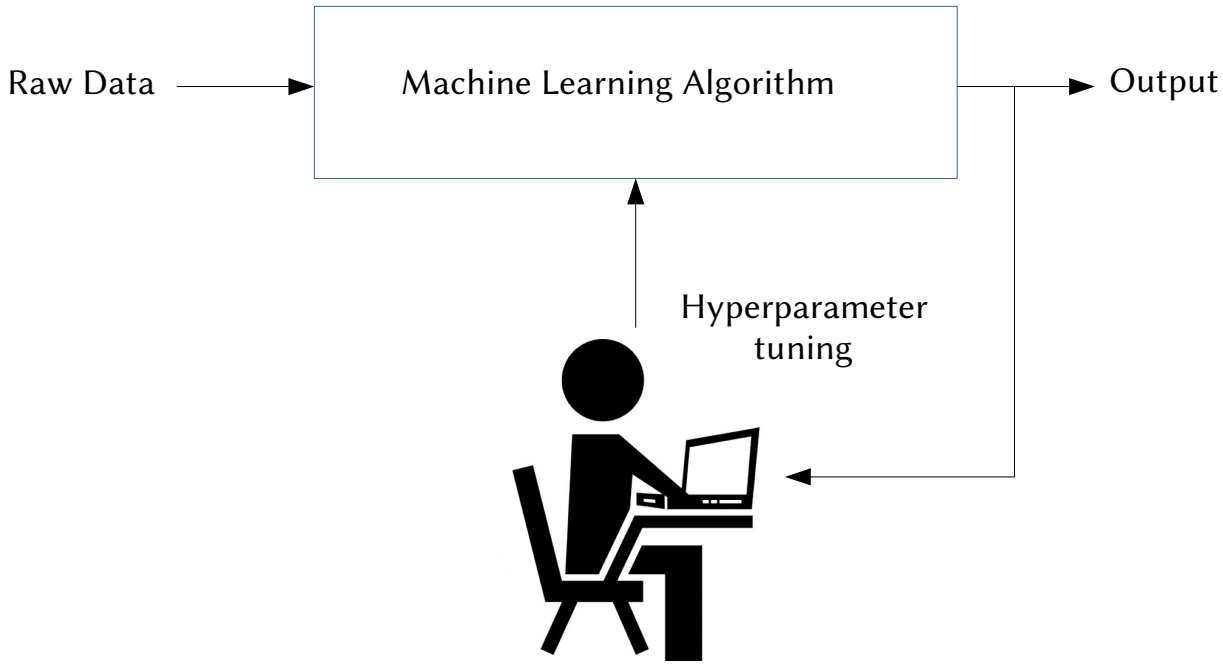


figure from
https://blog.easysol.net/building-ai-applications/

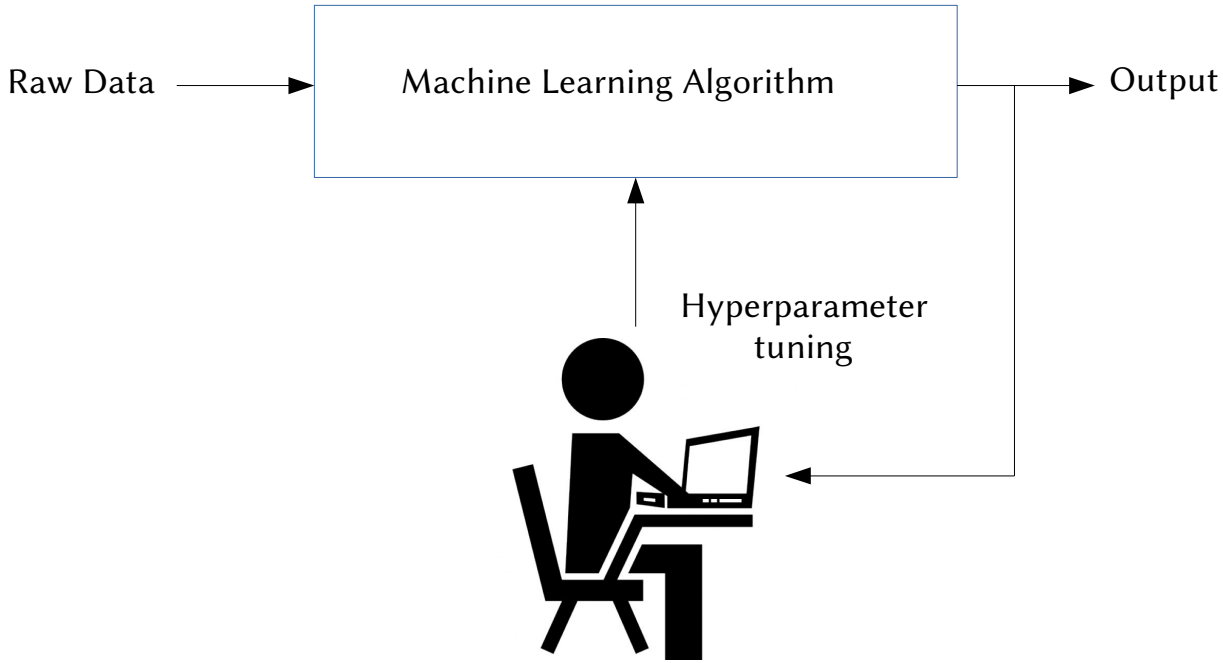
Towards End-to-End Learning



Towards End-to-End Learning



Towards End-to-End Learning



Grid search, random search, trial-and-error, graduate student descent,...

Optimizing “black-box” functions

We are interested in optimizing a ’well behaved’ function $f : \Theta \to \mathbb{R}$ over some bounded domain of hyperparameters $\Theta \subset \mathbb{R}^d$, i.e. in solving

$$\theta_\star = \operatorname{argmin}_{\theta \in \Theta} f(\theta).$$

However, f is not known explicitly, i.e. it is a black-box function and we can only ever obtain noisy and expensive evaluations of f.
Goal: Find θ such that $f(\theta) \approx f(\theta_\star)$ while minimizing the number of evaluations of f.

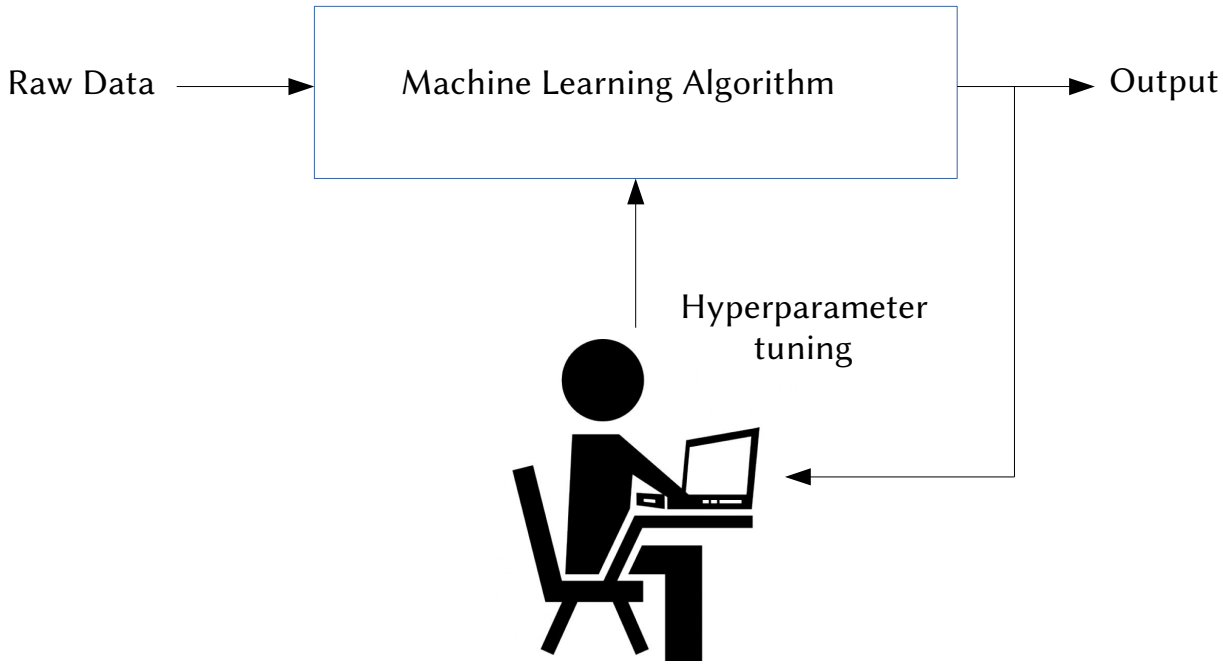
Probabilistic model for the objective f

Assuming that f is well behaved, we build a surrogate probabilistic model for it (typically a Gaussian Process, but other choices are possible).

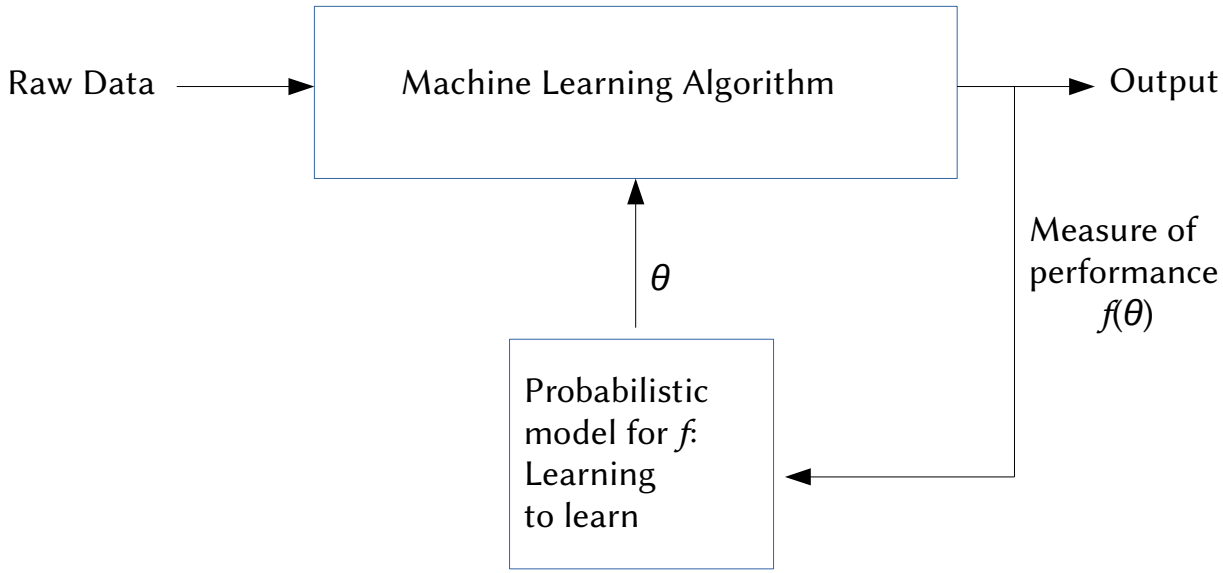
1. Compute the posterior predictive distribution of f using all evaluations so far.
2. Optimize a cheap proxy / acquisiton function instead of f which takes into account predicted values of f at new points as well as the **uncertainty in those predictions**: this proxy is typically much cheaper to evaluate than the actual objective f.
3. Evaluate the objective f at the optimum of the proxy and go to 1.

The proxy / acquisiton function should balance exploration against exploitation.

Towards End-to-End Learning



Towards End-to-End Learning



Optimizing “black-box” functions

Machine learning models often have a number of hyperparameters which need to be tuned:

- **kernel methods**: bandwidth in a Gaussian kernel, regularization parameters
- **neural networks**: number of layers, regularization parameters, layer size, batch size, learning rate

Define an objective function: a measure of generalization performance for a given set of hyperparameters obtained e.g. using cross-validation.

- Grid search, random search, trial-and-error...

Optimizing “black-box” functions

We are interested in optimizing a ’well behaved’ function $f : \mathcal{X} \to \mathbb{R}$ over some bounded domain $\mathcal{X} \subset \mathbb{R}^d$, i.e. in solving

$$x_\star = \operatorname{argmin}_{x \in \mathcal{X}} f(x).$$

However, f is not known explicitly, i.e. it is a **black-box** function and we can only ever obtain noisy (and potentially expensive as they may correspond to training of a large machine learning model or even running a complex physical experiment) evaluations of f.

Probabilistic model for the objective f

- Assuming that f is well behaved, we build a surrogate probabilistic model for it (Gaussian Process).
- Compute the posterior predictive distribution of f
- Optimize a cheap proxy / acquisiton function instead of f which takes into account predicted values of f at new points as well as the uncertainty in those predictions: this model is typically much cheaper to evaluate than the actual objective f.
- Evaluate the objective f at the optimum of the proxy.

The proxy / acquisiton function should balance exploration against exploitation.

Surrogate GP model

Assume that the noise ϵ_i in the evaluations of the black-box function is i.i.d. $\mathcal{N}\left(0, \delta^2\right)$:

$$\mathbf{f} \sim \mathcal{N}(0, \mathbf{K})$$
$$\mathbf{y}|\mathbf{f} \sim \mathcal{N}(\mathbf{f}, \delta^2 I).$$

Gives a closed form expression for the **posterior predictive mean** $\mu\left(x\right)$ and the **posterior predictive marginal standard deviation** $\sigma\left(x\right) = \sqrt{\kappa\left(x, x\right)}$ at any new location x, i.e.

$$f\left(x\right)|\mathcal{D} \sim \mathcal{N}\left(\mu\left(x\right), \kappa\left(x, x\right)\right),$$

where

$$\begin{aligned} \mu\left(x\right) &= \mathbf{k}_{x\mathbf{x}}(\mathbf{K} + \delta^2 I)^{-1}\mathbf{y}, \\ \kappa\left(x, x\right) &= k(x, x) - \mathbf{k}_{x\mathbf{x}}(\mathbf{K} + \delta^2 I)^{-1}\mathbf{k}_{\mathbf{x}x} \end{aligned}$$

- Exploitation: seeking locations with low posterior mean $\mu\left(x\right)$,
- Exploration: seeking locations with high posterior variance $\kappa\left(x, x\right)$.

Acquisition functions

- **GP-LCB**. "optimism in the phase of uncertainty"; minimize the lower $(1-\alpha)$-credible bound of the posterior of the unknown function values $f(x)$, i.e.

$$\alpha_{LCB}(x) = \mu(x) - z_{1-\alpha}\sigma(x),$$

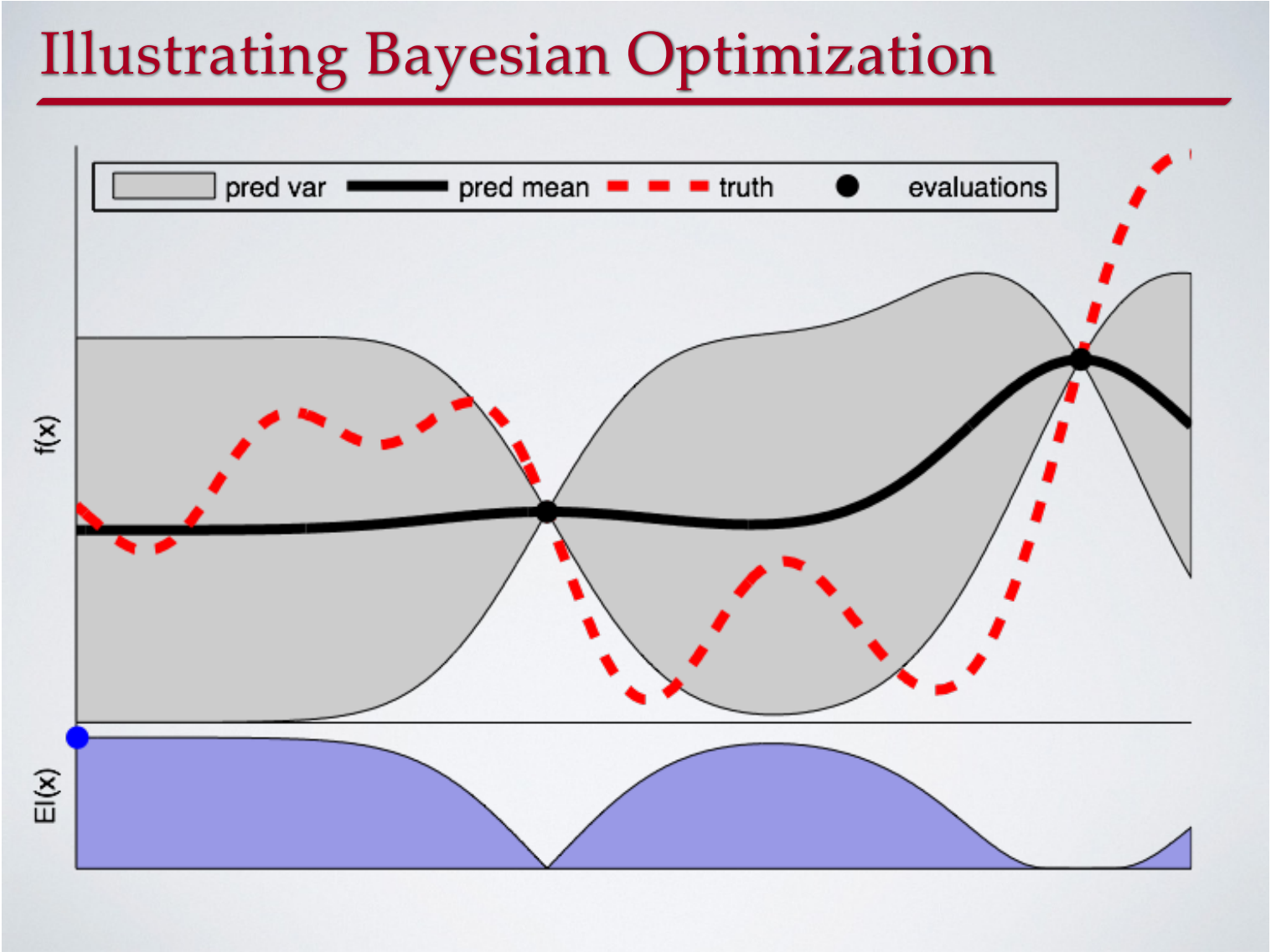
  where $z_{1-\alpha} = \Phi^{-1}(1-\alpha)$ is the desired quantile of the standard normal distibution.

- **PI** (probability of improvement). $\tilde{x}$: the optimal location so far, $\tilde{y}$: the observed minimum. Let $u(x) = \mathbb{1}\{f(x) < \tilde{y}\}$,

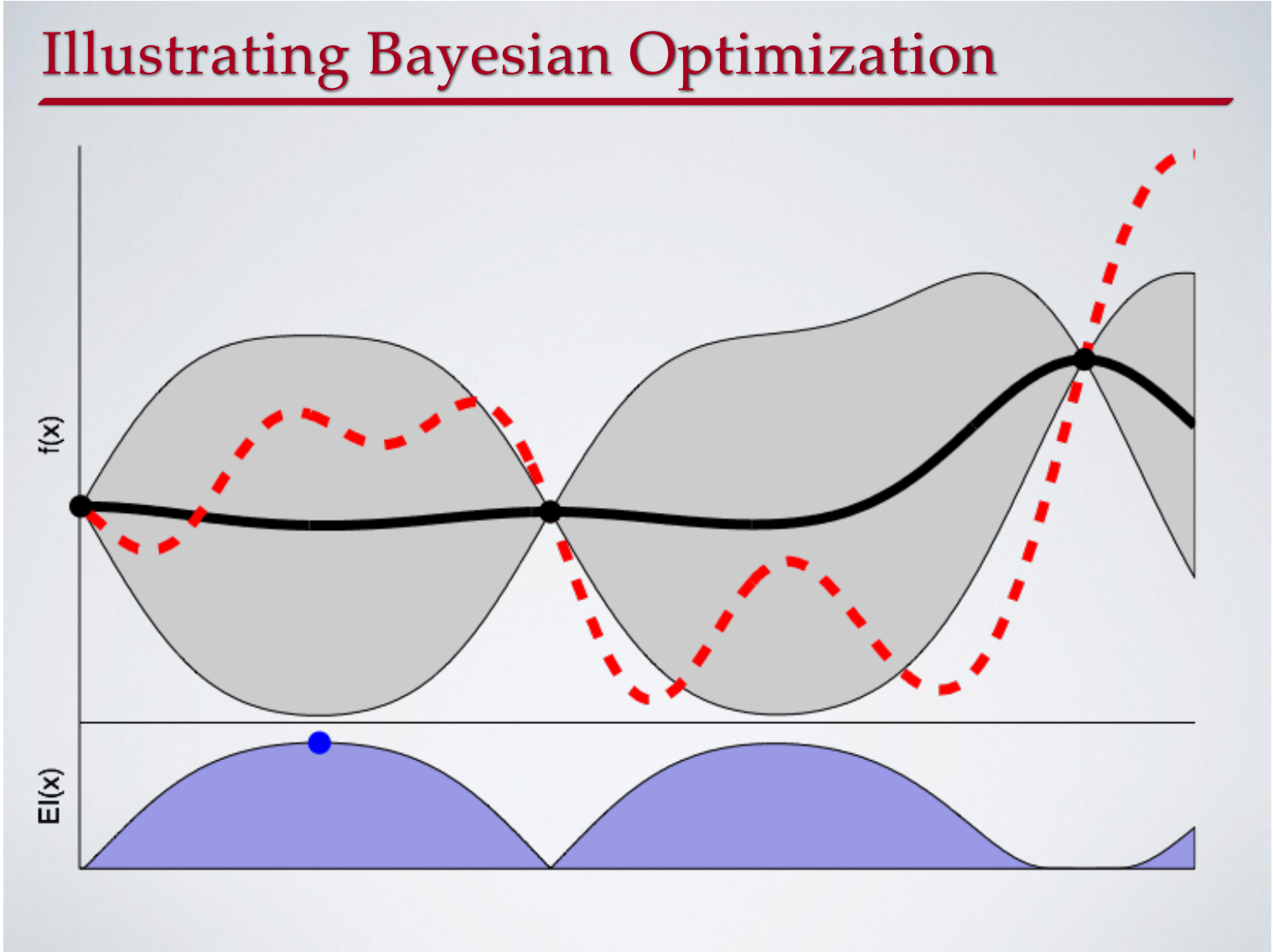
$$\alpha_{PI}(x) = \mathbb{E}[u(x)|\mathcal{D}] = \Phi(\gamma(x)), \quad \gamma(x) = \frac{\tilde{y} - \mu(x)}{\sigma(x)}$$

- **EI** (expected improvement). Let $u(x) = \max(0, \tilde{y} - f(x))$

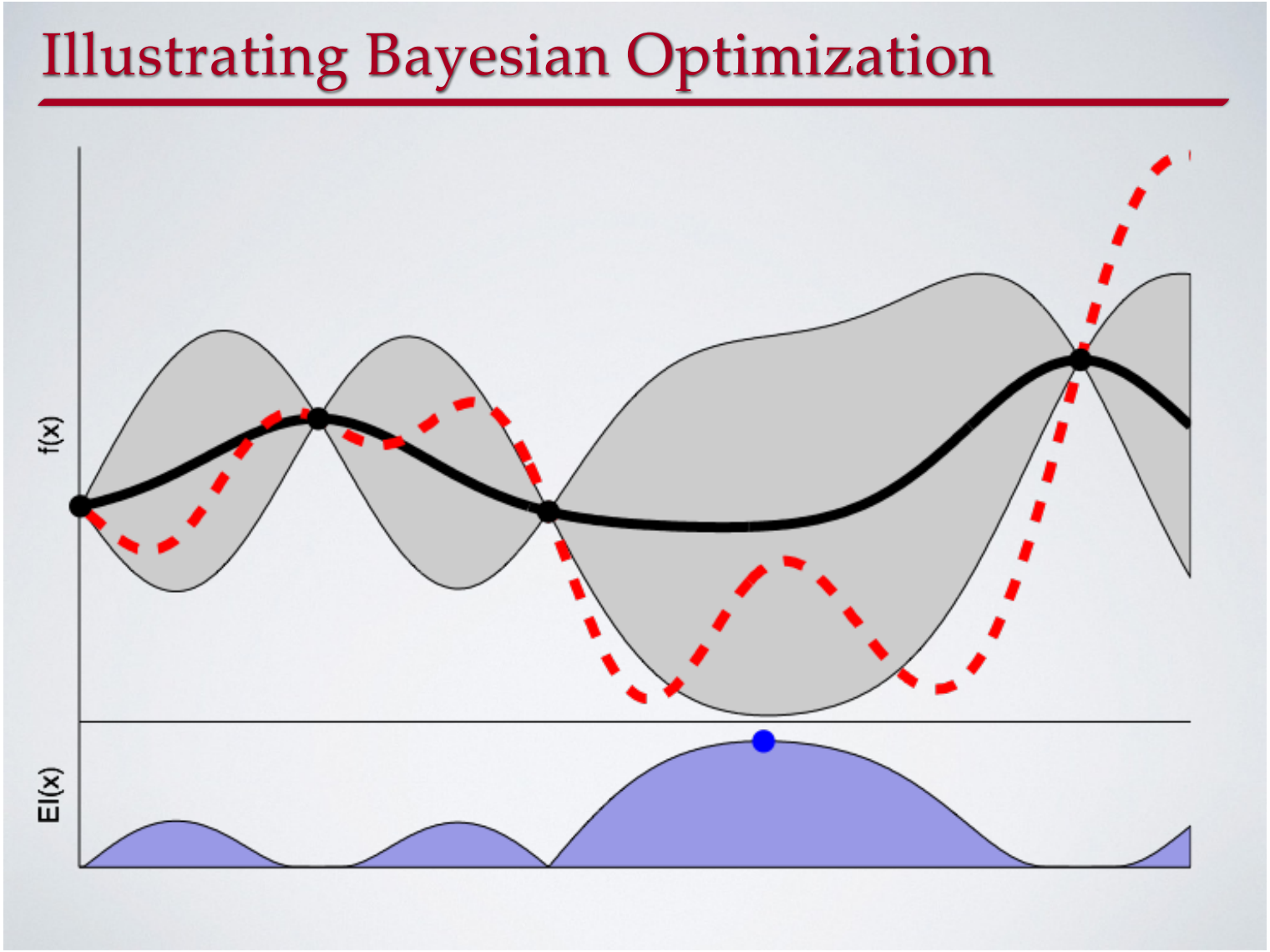
$$\alpha_{EI}(x) = \mathbb{E}[u(x)|\mathcal{D}] = \sigma(x)(\gamma(x)\Phi(\gamma(x)) + \phi(\gamma(x))).$$



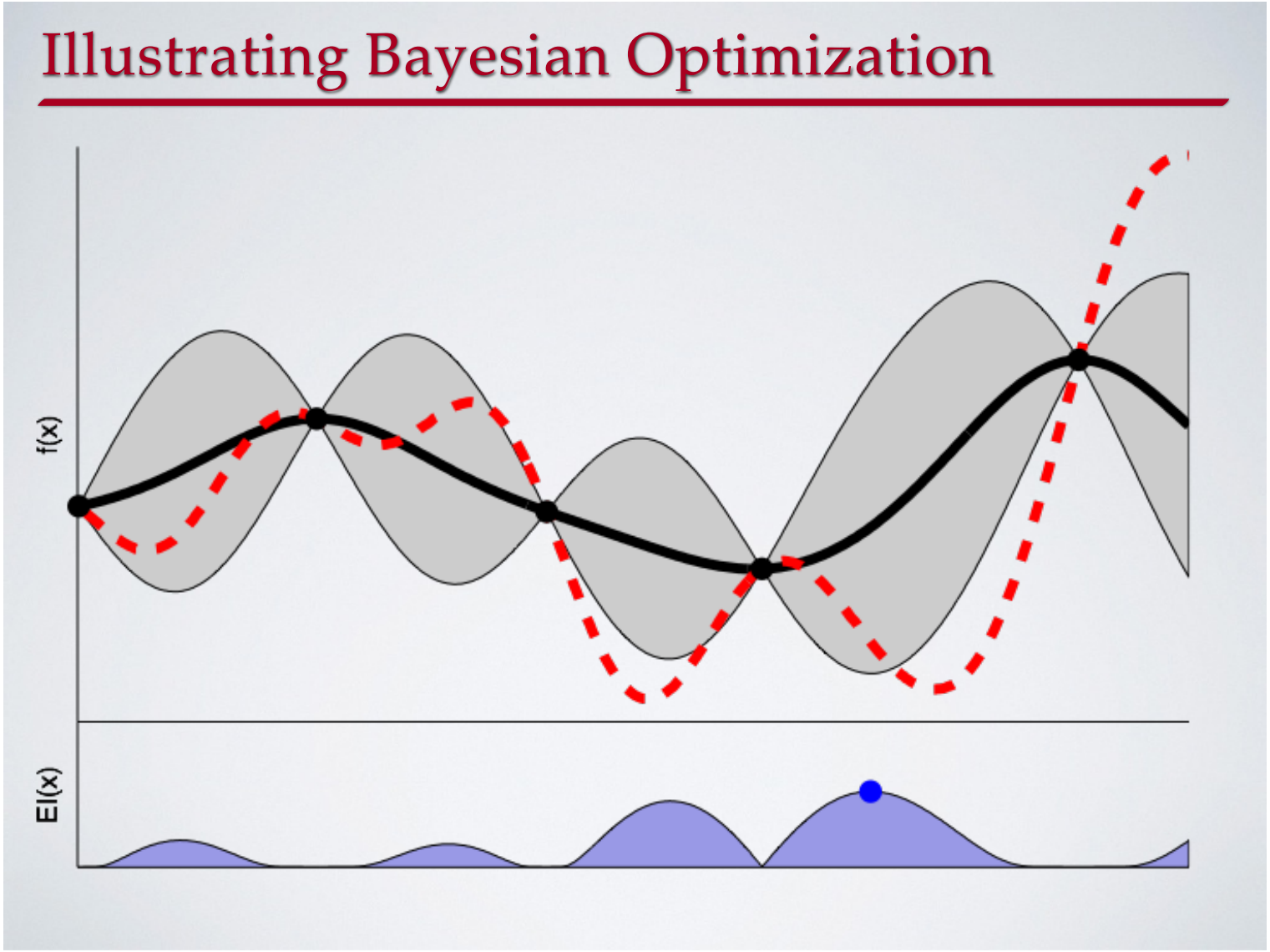
slides from **A Tutorial on Bayesian Optimization for Machine Learning** by Ryan Adams



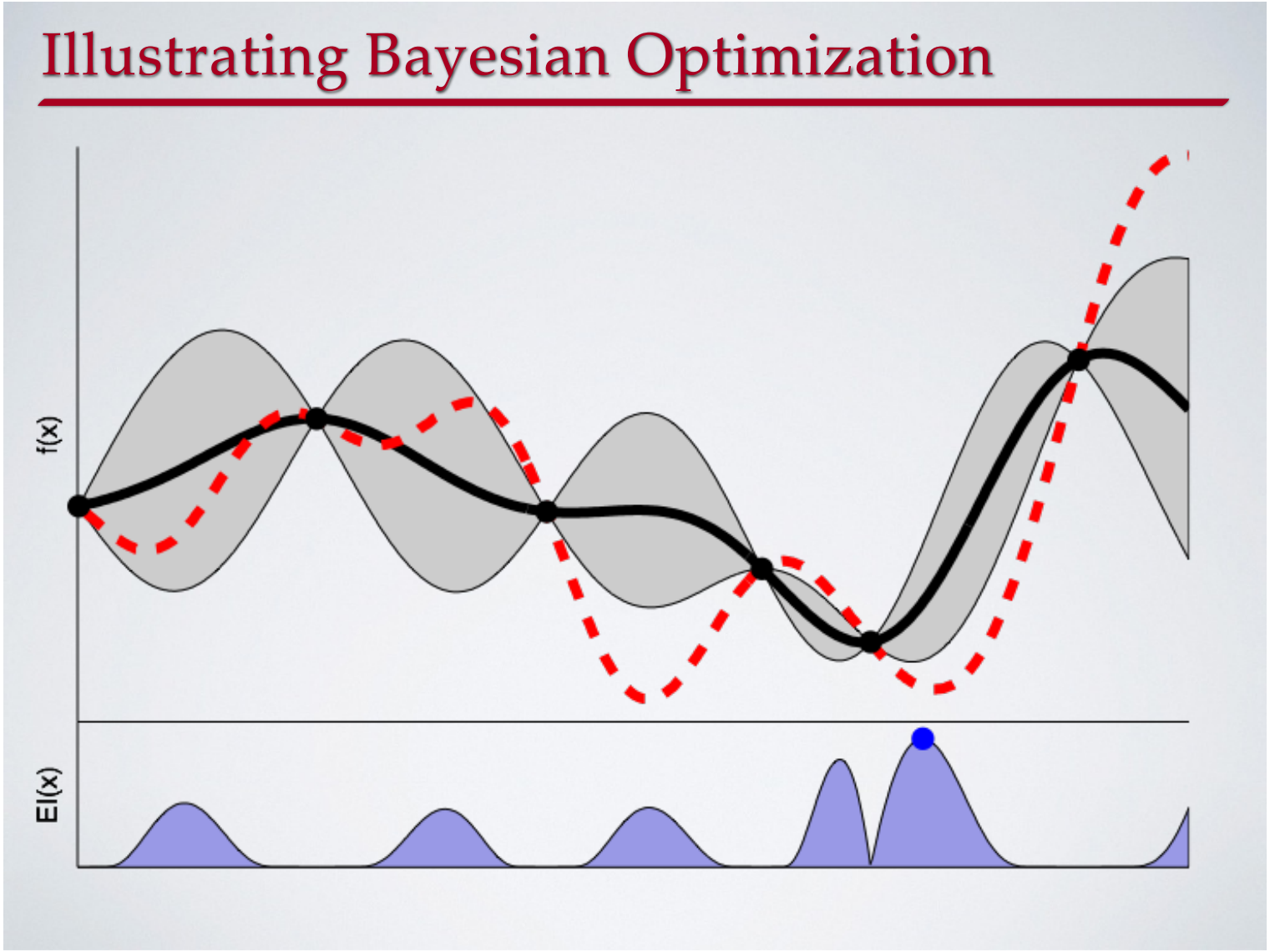
slides from **A Tutorial on Bayesian Optimization for Machine Learning** by Ryan Adams



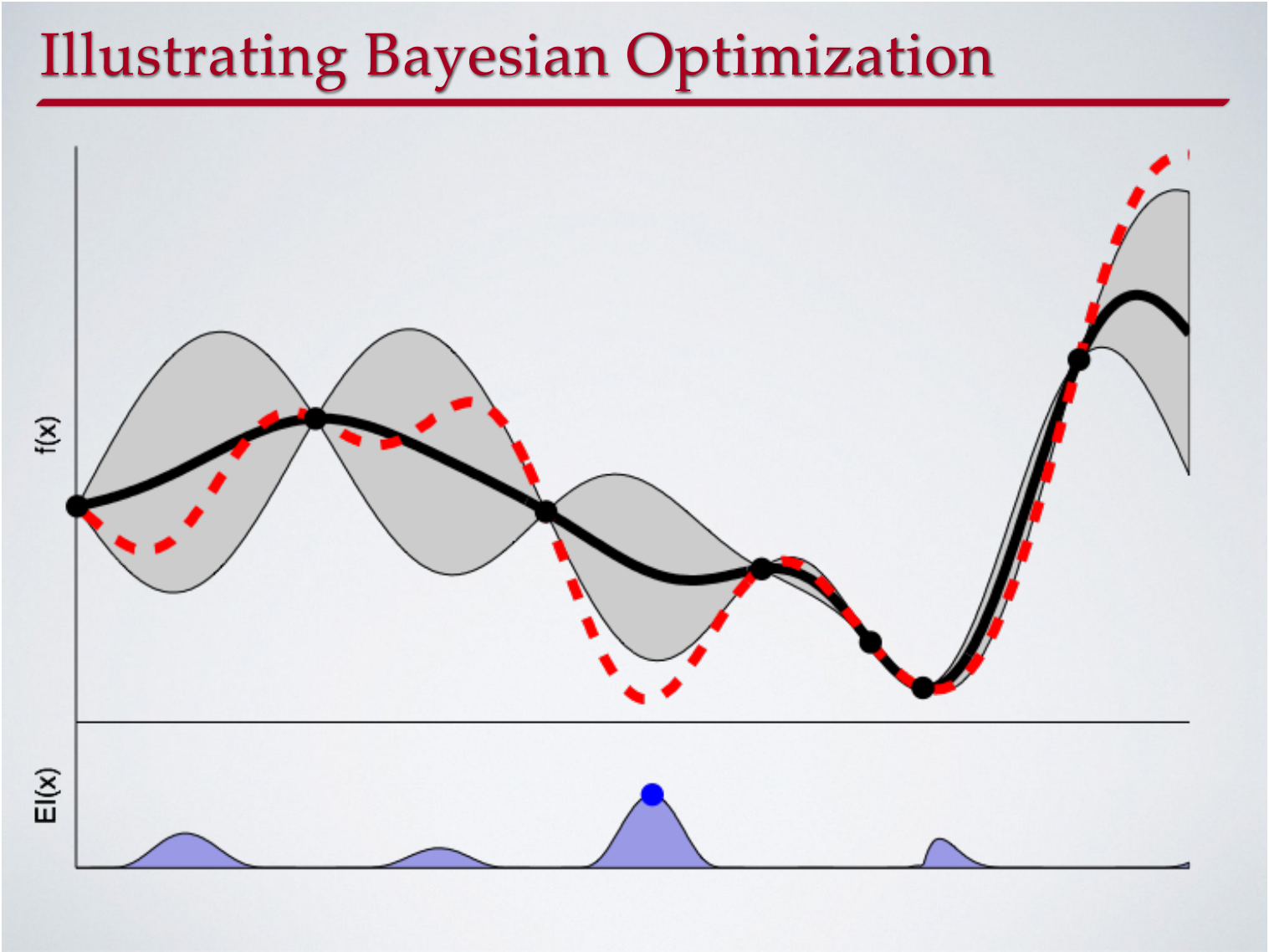
slides from **A Tutorial on Bayesian Optimization for Machine Learning** by Ryan Adams



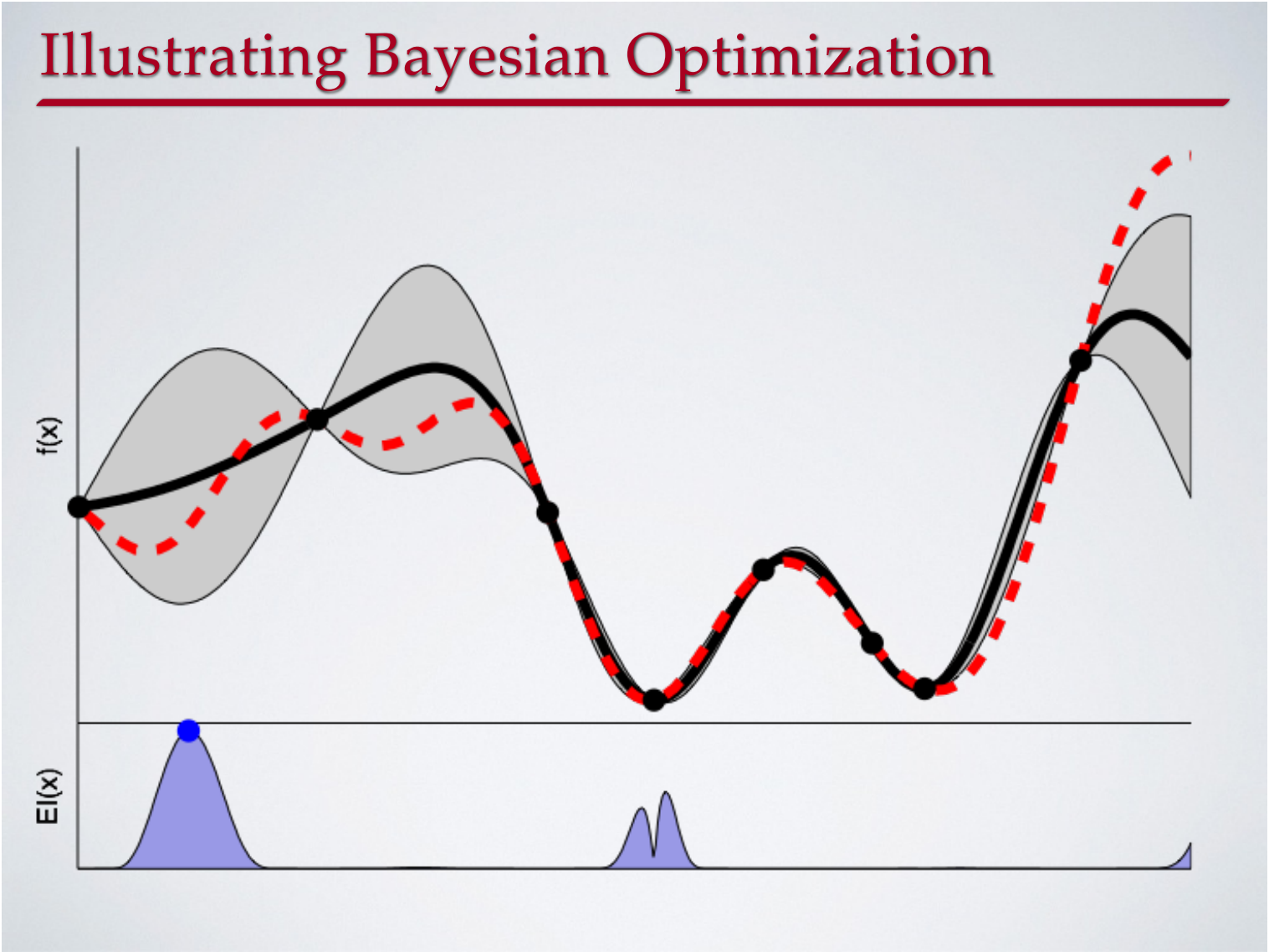
slides from **A Tutorial on Bayesian Optimization for Machine Learning** by Ryan Adams



slides from **A Tutorial on Bayesian Optimization for Machine Learning** by Ryan Adams



slides from **A Tutorial on Bayesian Optimization for Machine Learning** by Ryan Adams



slides from **A Tutorial on Bayesian Optimization for Machine Learning** by Ryan Adams

We considered a selection of topics in statistical machine learning, but there is much more!

- **Topics we did not cover**: multitask learning, online learning, reinforcement learning, message passing algorithms, generative adversarial networks, ensemble methods, boosting, causality, interpretability, robustness, fairness, differential privacy...
- **Further resources**:
  - Bishop, Pattern Recognition and Machine Learning, Springer.
  - Murphy, Machine Learning: A Probabilistic Perspective, MIT Press.
  - Shalev-Shwartz and Ben-David, Understanding Machine Learning: From Theory to Algorithms, Cambridge University Press.
  - Schölkopf and Smola, Learning with Kernels, MIT Press.
  - Rasmussen and Williams, Gaussian Processes for Machine Learning, MIT Press.
  - Goodfellow, Bengio and Courville, Deep Learning, MIT Press.
  - Machine Learning Summer Schools, videolectures.net.
  - Conferences: NeurIPS, ICML, AISTATS, UAI.