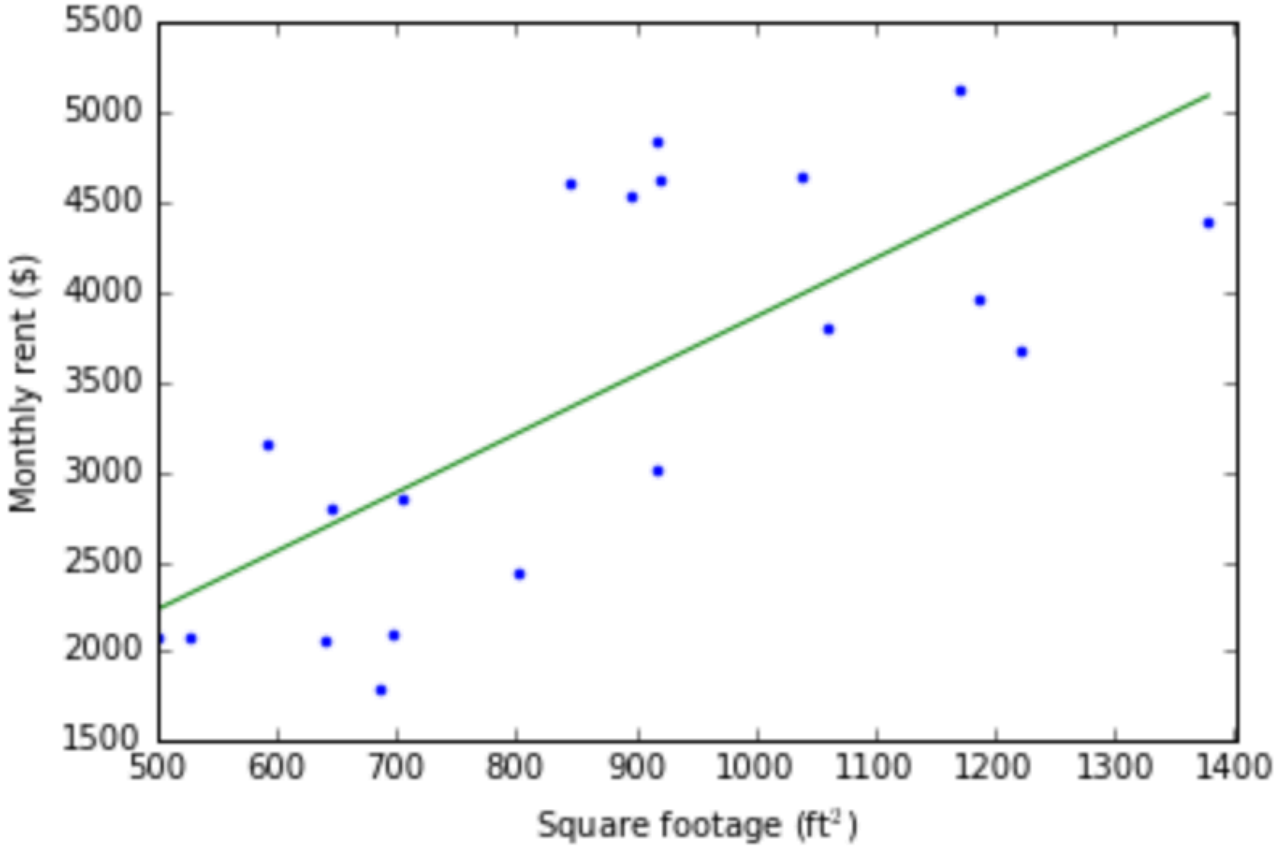
2018-01-17: Announcements

- HW #1 is uploaded to CCLE and due to Gradescope at 11:59pm on Monday, 22 Jan 2018.
- Please feel free to help answer questions on Piazza.
- Reminder that it is all okay to use Python 2.
- A/V system *ought* to be fine…
- Lecture notes continue to be updated and are labeled with the last date of update on CCLE.

2018-01-17: RECAP from 2018-01-10

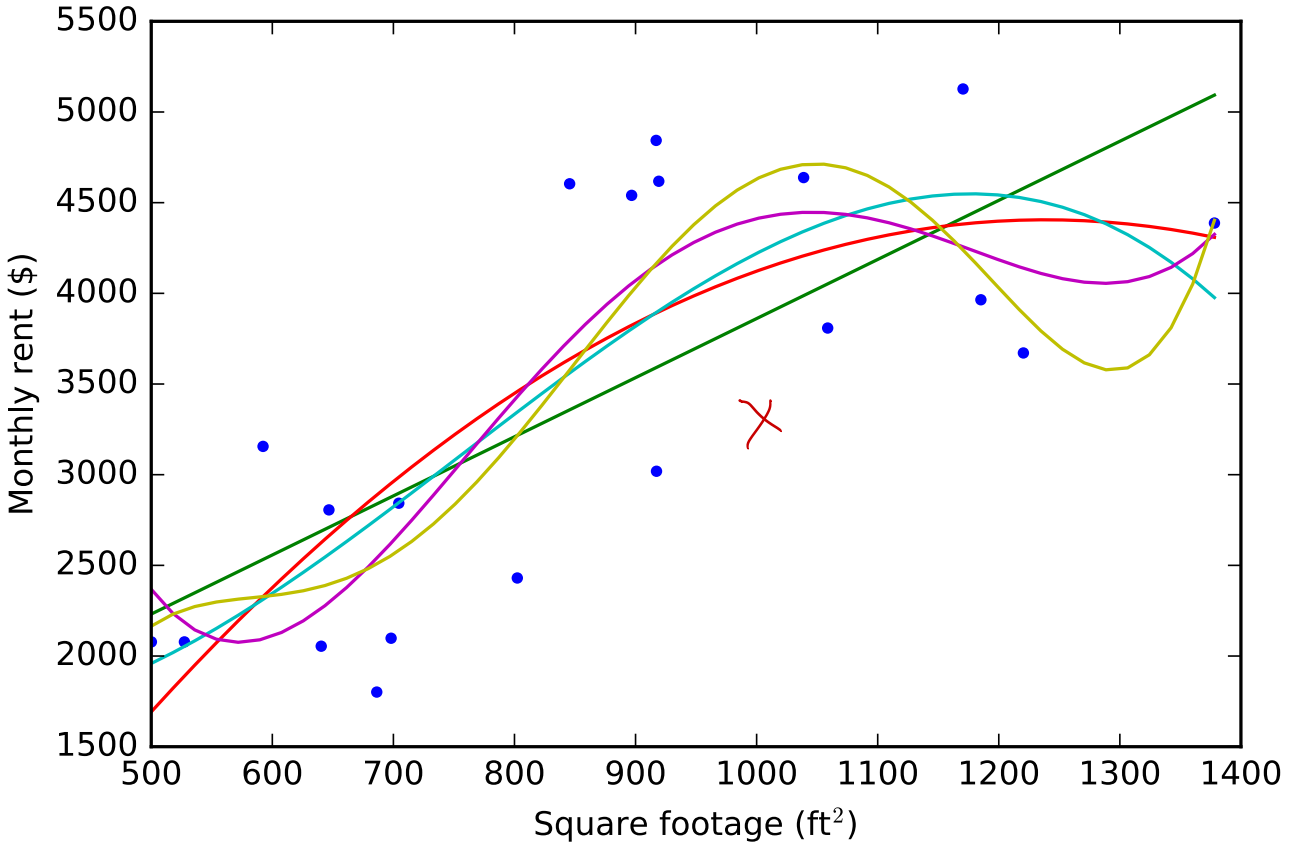


$$\mathcal{L} = \frac{1}{2}\sum_{i=1}^{N}(y^{(i)} - \theta^T \hat{\mathbf{x}}^{(i)})^2$$

$$y = \theta^T \hat{x}$$

$$\theta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T Y$$

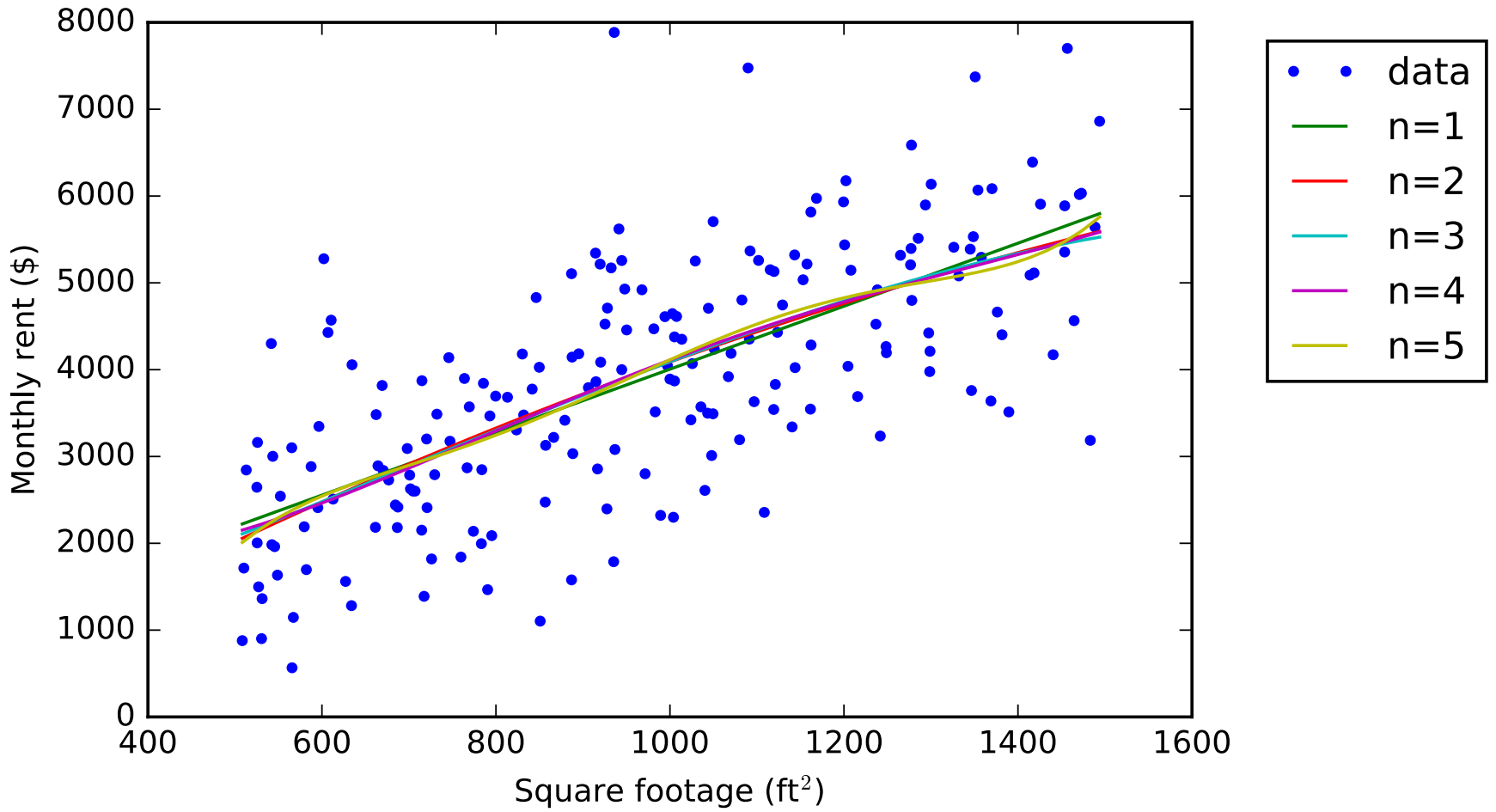
2018-01-17: RECAP from 2018-01-10



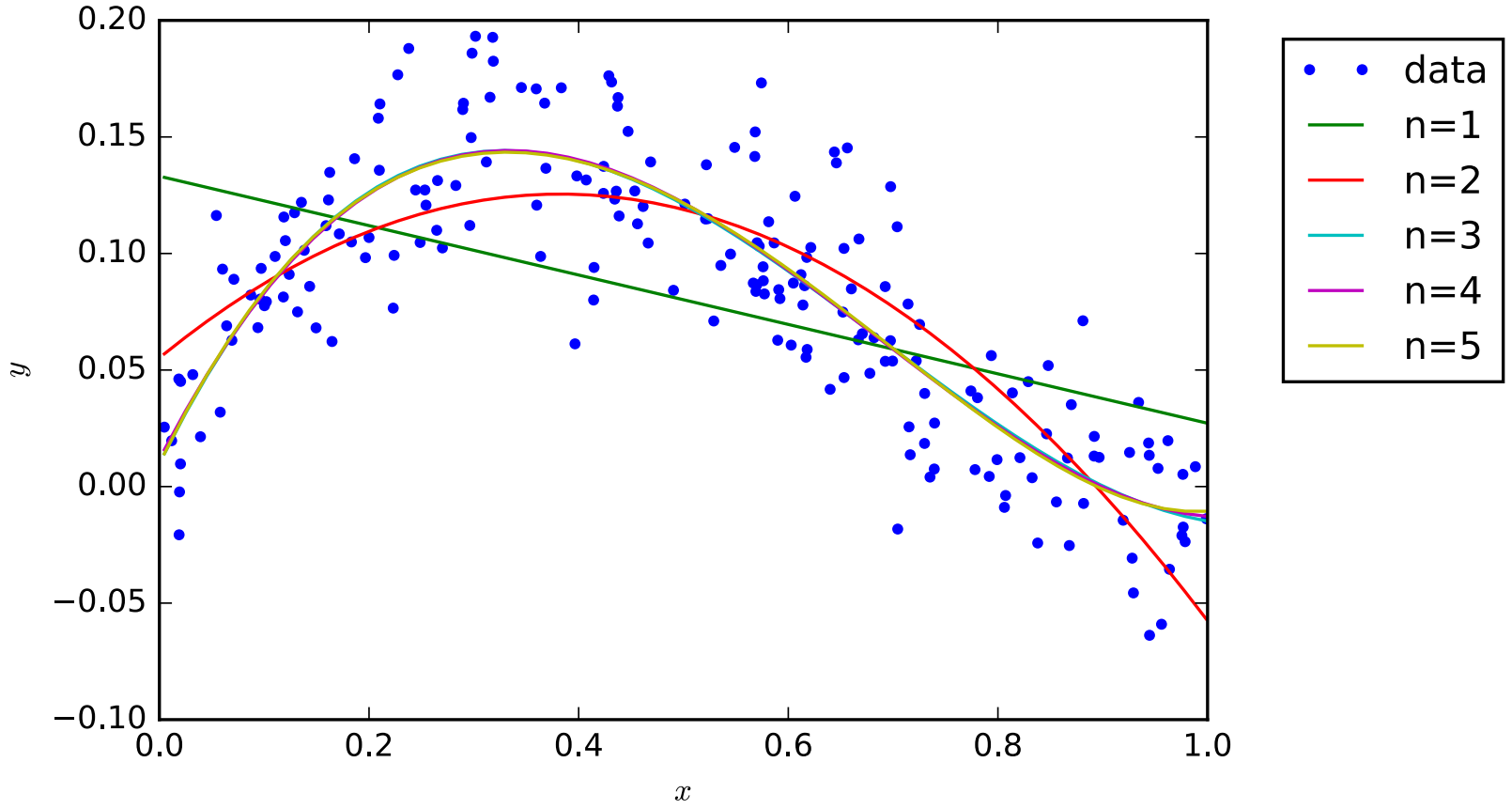
$y = ax + b$

$y = a_5 x^5 + a_4 x^4 + \ldots$

2018-01-17: RECAP from 2018-01-10



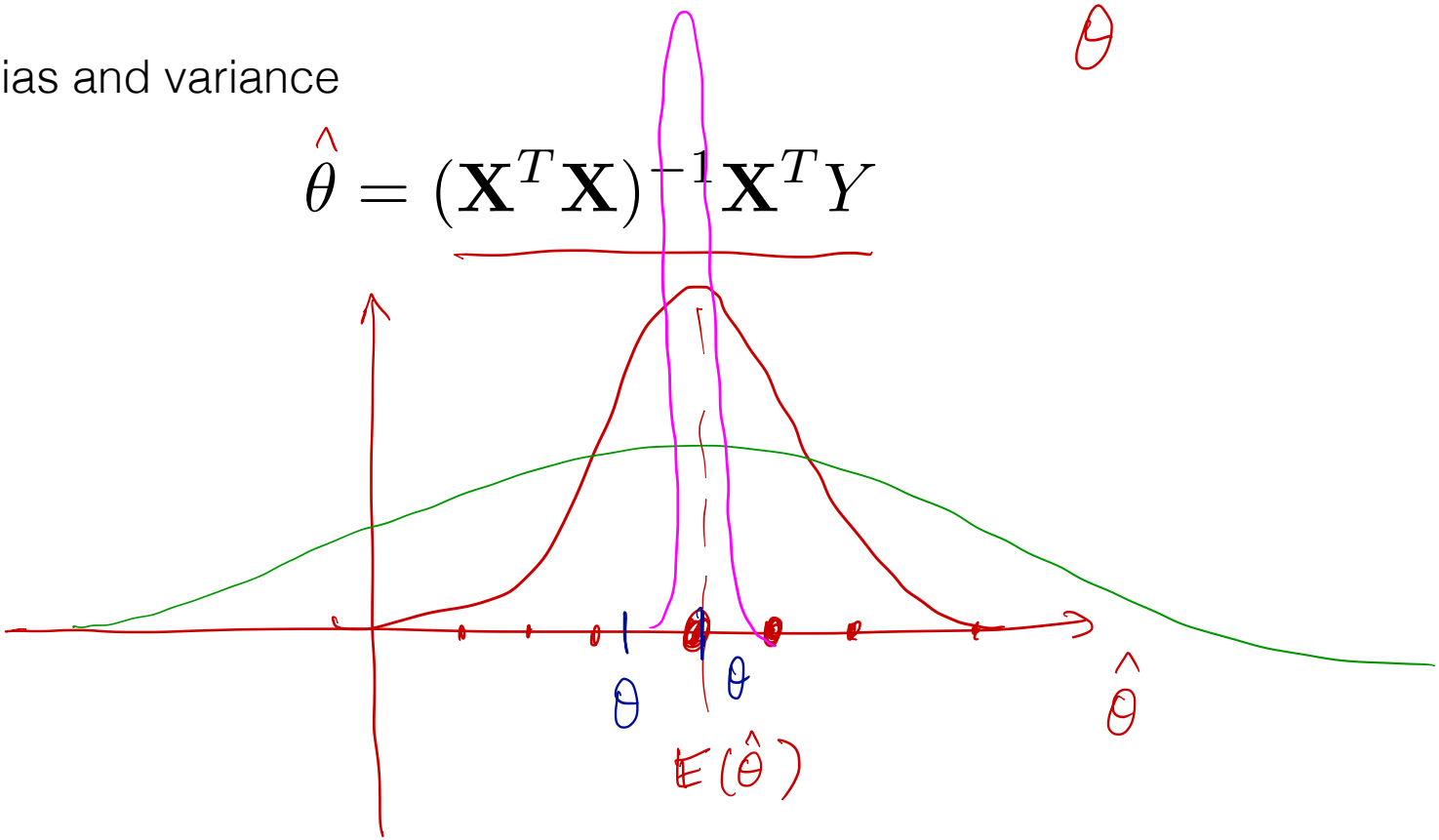
2018-01-17: RECAP from 2018-01-10



2018-01-17: RECAP from 2018-01-10

Estimator bias and variance

$$\hat{\theta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T Y$$



Estimator bias

The *bias* of an estimator, $\hat{\theta}_m$, is defined as:

$$\text{bias}(\hat{\theta}_m) = \mathbb{E}(\hat{\theta}_m) - \theta$$

Note:

- The expectation is taken over the data (where the randomness is introduced through samples $\mathbf{x}^{(i)}$).
- Informally, the bias measures how close $\hat{\theta}_m$ comes to estimating θ on average.
- An estimator is called *unbiased* if $\text{bias}(\hat{\theta}_m) = 0$.

Estimator bias example

$$x^{(i)} = \begin{cases} 0 & \text{w.p. } 1-\theta \\ 1 & \text{w.p. } \theta \end{cases}$$

Estimator bias (example)

- Let $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$ be iid samples from a Bernoulli distribution with mean θ.
- The *sample mean* estimator is given by:

$$\hat{\theta}_m = \frac{1}{m} \sum_{i=1}^{m} x^{(i)}$$

Intuitively, this ought to be a very reasonable estimator of the Bernoulli mean (or equivalently parameter).

Estimator bias example

Then, the bias of this estimator is:

$$\text{bias}(\hat{\theta}_m) = \mathbb{E}\left(\frac{1}{m}\sum_{i=1}^{m} x^{(i)}\right) - \theta$$

$$= \left(\frac{1}{m}\sum_{i=1}^{m} \mathbb{E}\,x^{(i)}\right) - \theta$$

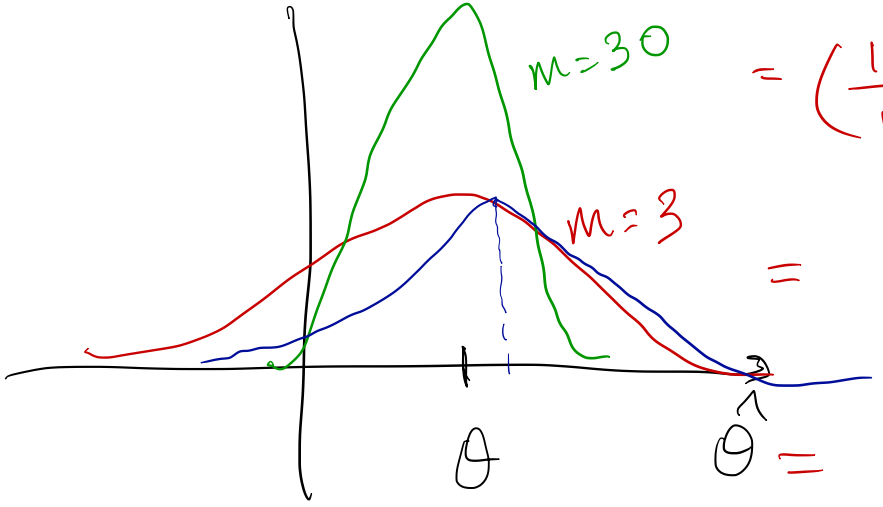
$$= \left(\frac{1}{m}\sum_{i=1}^{m} \theta\right) - \theta$$

$$= \frac{m\theta}{m} - \theta$$

$$= 0$$

$\mathbb{E}(aX) = a\mathbb{E}X$

$\mathbb{E}(X+Y) = \mathbb{E}X + \mathbb{E}Y$



Estimator variance

An unbiased estimator may on average estimate θ, but any single instance of $\hat{\theta}$ may deviate from θ. The variance of $\hat{\theta}$ in predicting is called the variance of the estimator, denoted $\mathrm{var}(\hat{\theta})$.

Some notes:

- This variability is due to the data samples that you get.
- The *standard error* of the estimator is $\sqrt{\mathrm{var}(\hat{\theta})}$ and is commonly denoted $\mathrm{SE}(\hat{\theta})$.

Estimator variance example

Estimator variance (example)

$p(X, Y) = P(X) P(Y)$

Let $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$ be iid samples from a Bernoulli distribution with mean θ. The *sample mean* estimator is given by:

$\mathrm{var}(aX) = a^2 \mathrm{var}(X)$

$$\hat{\theta}_m = \frac{1}{m} \sum_{i=1}^{m} x^{(i)}$$

$\mathrm{var}(X+Y) = \mathrm{var}(X) + \mathrm{var}(Y) + 2\,\mathrm{cov}(X,Y)$

Then, the variance of this estimator is:

$$\mathrm{var}(\hat{\theta}_m) = \mathrm{var}\left(\frac{1}{m} \sum_{i=1}^{m} x^{(i)}\right)$$

$$= \frac{1}{m^2} \sum_{i=1}^{m} \mathrm{var}(x^{(i)})$$

$$= \frac{1}{m^2} m(\theta(1-\theta))$$

$$= \frac{1}{m} \theta(1-\theta)$$

Estimator variance example

Estimator variance (example)

A common metric used in evaluating the average of the data is the standard error of the mean. Given iid samples $x^{(1)}, \ldots, x^{(m)}$, the standard error of the mean is given by:

$$\begin{aligned}
\mathrm{SE}(\hat{\mu}_m) &= \sqrt{\mathrm{var}\left(\frac{1}{m}\sum_{i=1}^{m} x^{(i)}\right)} \\
&= \sqrt{\frac{1}{m^2}\sum_{i=1}^{m} \mathrm{var}(x^{(i)})} \\
&= \sqrt{\frac{1}{m^2} m \mathrm{var}(x^{(i)})} \\
&= \frac{\sigma}{\sqrt{m}}
\end{aligned}$$

What does this mean intuitively?

Bias and mean-square error trade off

Trading off bias and variance

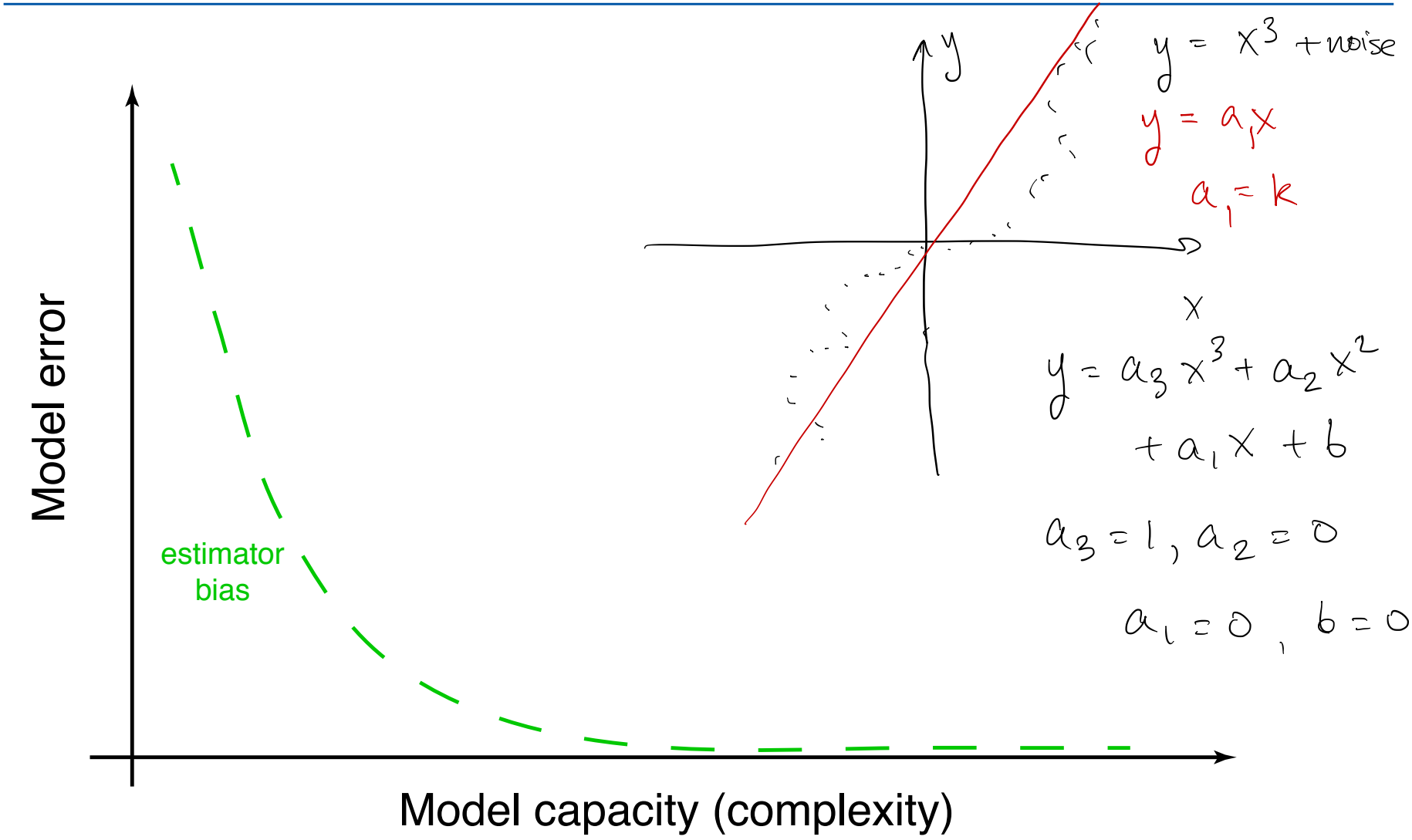
We may at times have to choose between a low-bias, high-variance estimator, or a high-bias, low-variance estimator.

- The mean squared error (MSE) naturally trades these two off.

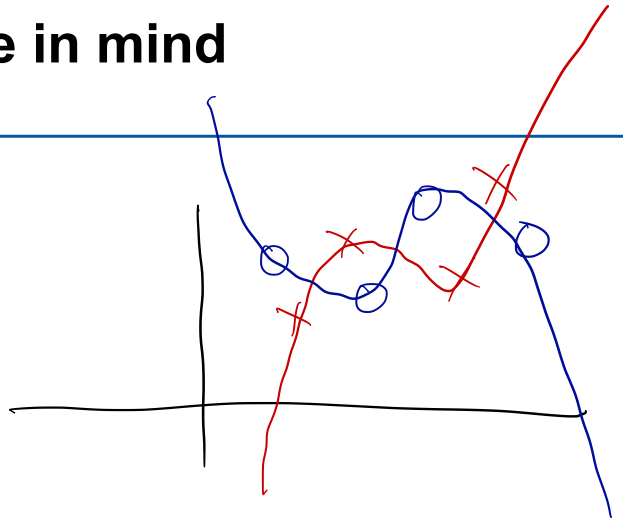
$$
\begin{aligned}
\text{MSE} &= \mathbb{E}(\hat{\theta}_m - \theta)^2 \\
&= \mathbb{E}(\hat{\theta}_m - \mathbb{E}\hat{\theta}_m + \mathbb{E}\hat{\theta}_m - \theta)^2 \\
&= \mathbb{E}(\hat{\theta}_m - \mathbb{E}\hat{\theta}_m)^2 + \mathbb{E}\left[2(\hat{\theta}_m - \mathbb{E}\hat{\theta}_m)(\mathbb{E}\hat{\theta}_m - \theta)\right] + \mathbb{E}(\mathbb{E}\hat{\theta}_m - \theta)^2 \\
&= \text{var}(\hat{\theta}_m) + 2\text{bias}(\hat{\theta}_m)\mathbb{E}(\hat{\theta}_m - \mathbb{E}\hat{\theta}_m) + \mathbb{E}\text{bias}(\hat{\theta}_m)^2 \\
&= \text{var}(\hat{\theta}_m) + \text{bias}(\hat{\theta}_m)^2
\end{aligned}
$$

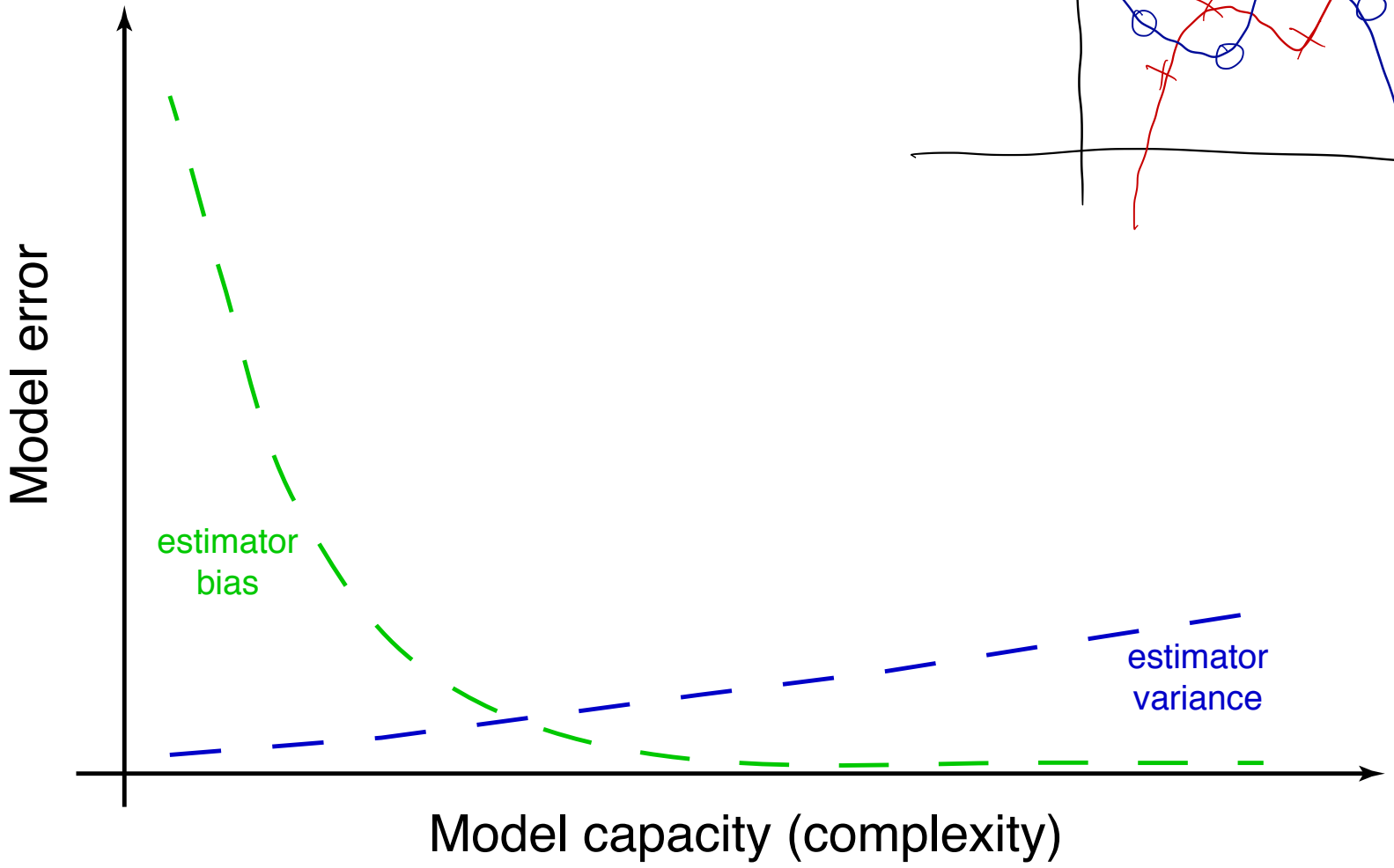
- Thus, minimizing MSE will simultaneously minimize the bias and variance of the estimator.

A picture to then have in mind

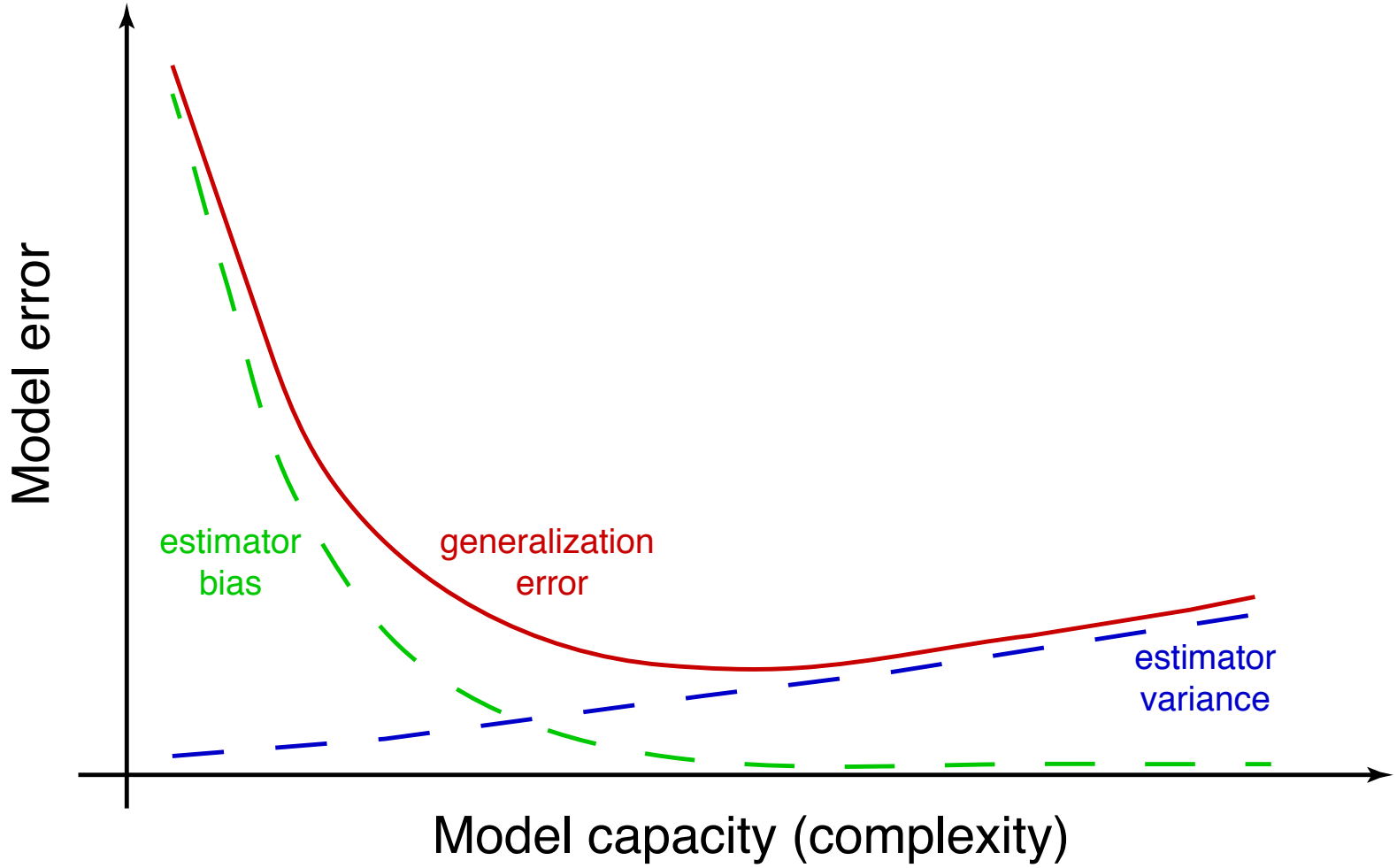


A picture to then have in mind

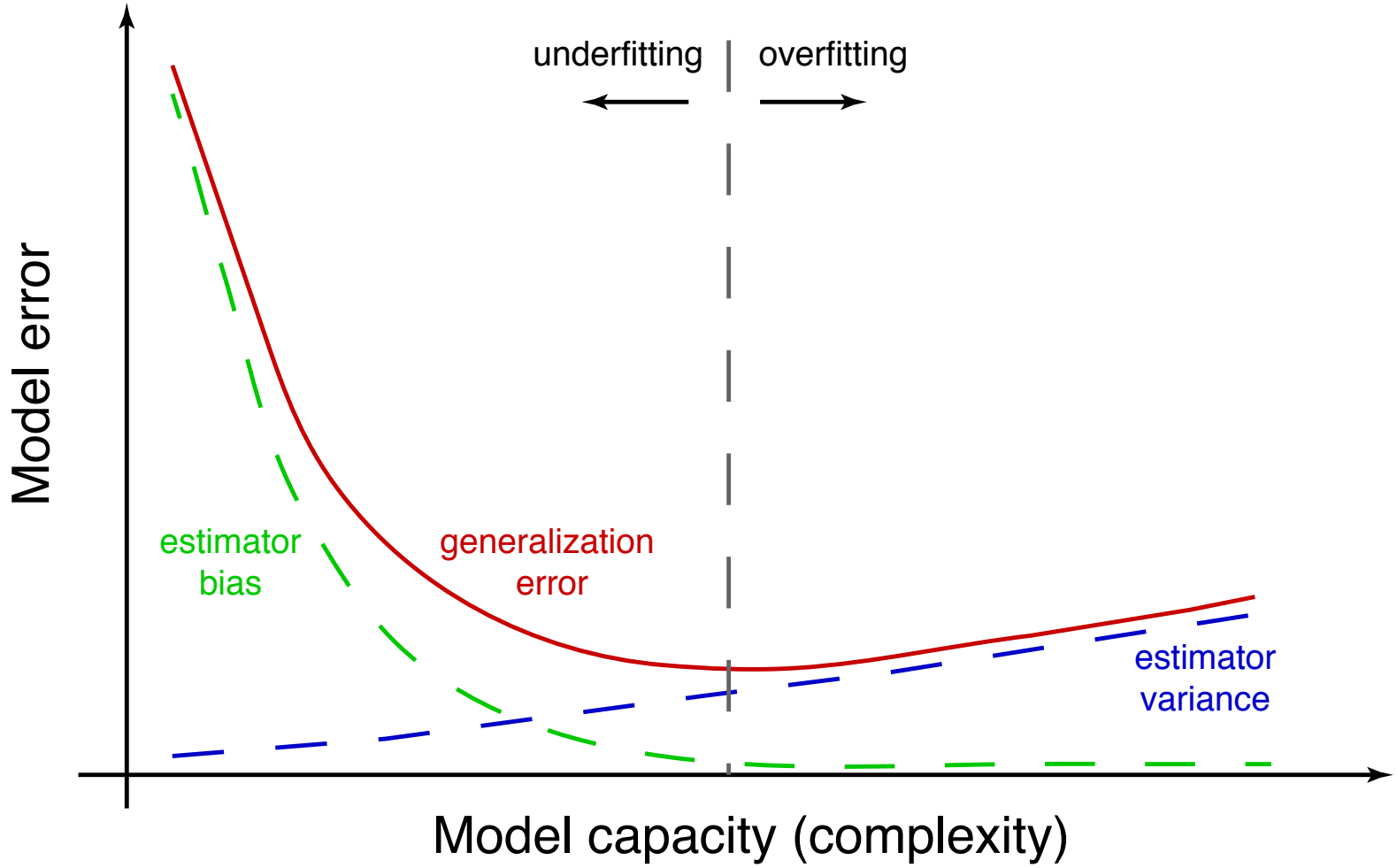




A picture to then have in mind



A picture to then have in mind



Picking a best model

From this picture, a reasonable way to pick a model is to assess its generalization error, and pick a setting of the parameters that results in minimal value.

$y = a_1 x + b$

$y = a_2 x^2 + a_1 x + b$

A brief aside:

Sometimes there may be scenarios where dataset size is so limited that it is not possible to have a reasonably good test dataset. In these cases, we may utilize *model selection*, which is itself an entire field.

At a high level, the principles of model selection is to find a way to *penalize* the model for being overly complex. There is an art to designing some penalty terms.

Criterion for selecting models include:

- Bayes information criterion
- Akaike information criterion
- Deviance information criterion

My recommendation: if you have enough data to make a test set, do it.

Evaluating generalization error

$y = \theta^T x$

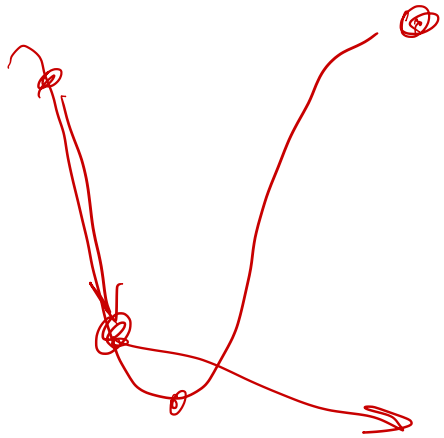
Training, validation, and testing data

The standard for training, evaluating, and choosing models is to use different datasets for each step.

- **Training data** is data that is used to learn the parameters of your model.
- **Validation data** is data that is used to optimize the hyperparameters of your model. This avoids the potential of overfitting to nuances in the testing dataset.
- **Testing data** is data that is used to score your model.

$n = 4$ $y = a_4 x^4 + \ldots$ $n = 3$

Note, in many cases, testing and validation datasets are used interchangeably as datasets that are used to evaluate the model. In this scenario, hyperparameters would also be optimized using training data.



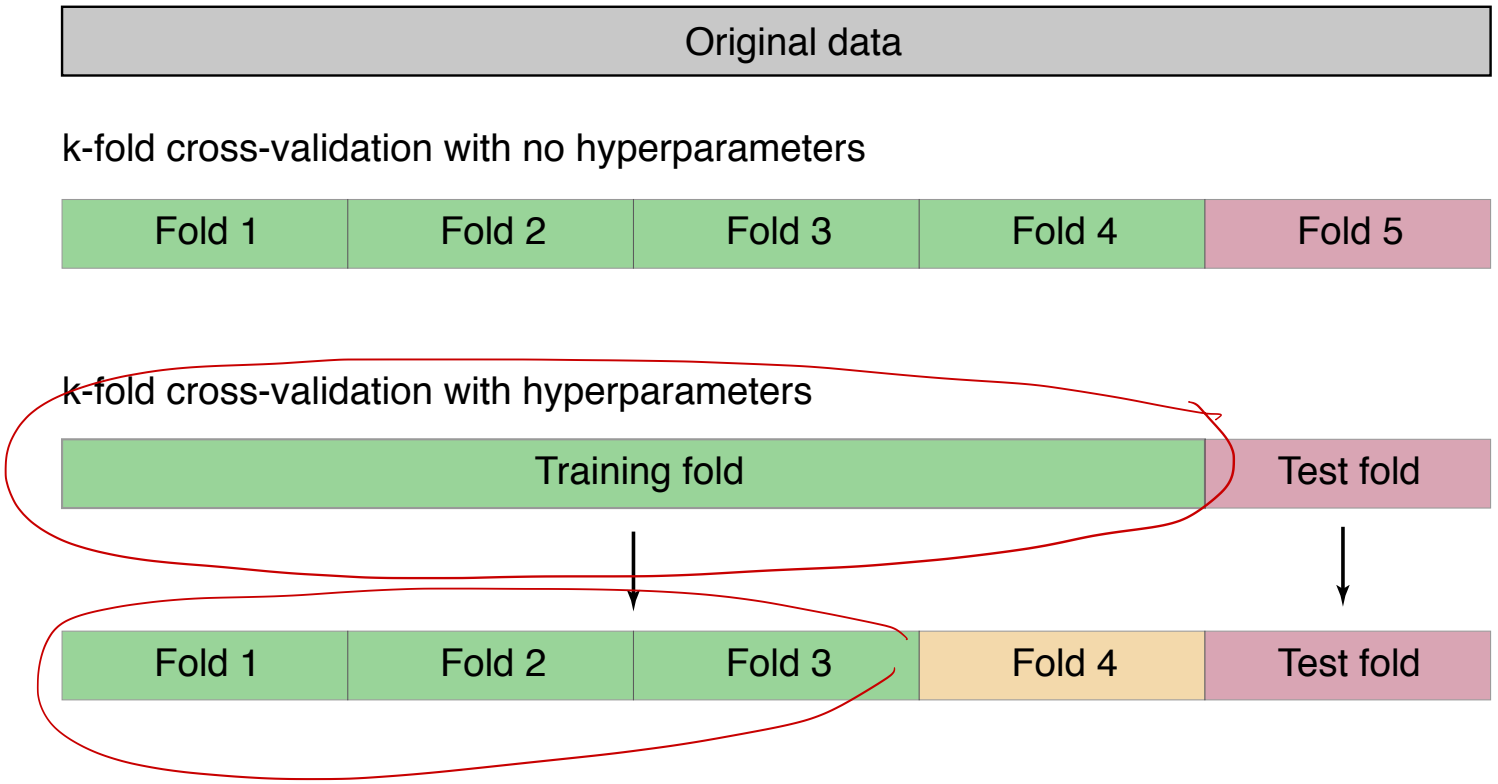
Evaluating generalization error

k-fold cross validation

In a common scenario, you will be given a training dataset and a testing dataset. To train a model using this dataset, one common approach is k-fold cross validation. Then the procedure looks as follows:

- Let the training dataset contain N examples. 1000
- Then, split the data into k equal sets, each of N/k examples. Each of these sets is called a "fold." K=5 200 800 200
- $k-1$ of the folds are datasets that are used to train the model parameters.
- The remaining fold is a testing dataset used to evaluate the model.
- You may repeatedly train the model by choosing which folds comprise the training folds and the testing fold.

Evaluating generalization error



k-fold cross validation from housing example

Routines for training models and testing data.

```
def train_models(x, y):
    # Fit a polynomial model up to N=5
    N = 5
    thetas = []
    for i in np.arange(N):
        xhat = np.vstack((x, np.ones_like(x))) if i == 0 else np.vstack((x**(i+1), xhat))
        thetas.append(np.linalg.inv(xhat.dot(xhat.T)).dot(xhat.dot(y)))

    return thetas

def test_models(x, y, thetas):
    N = 5
    errors = []
    for i in np.arange(N):
        xhat = np.vstack((x, np.ones_like(x))) if i == 0 else np.vstack((x**(i+1), xhat))
        errors.append(np.sqrt(np.sum((y - thetas[i].dot(xhat))**2) / len(y)))
    return errors
```

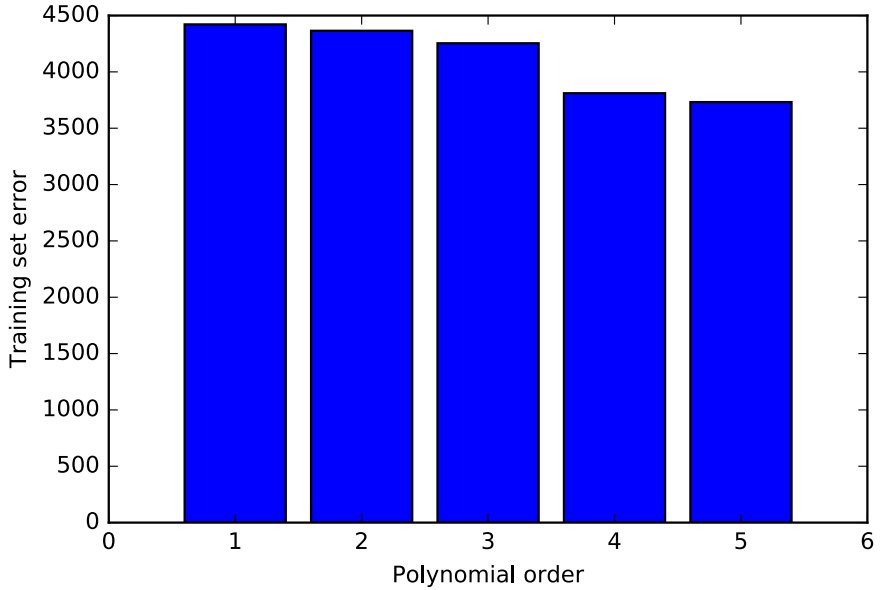
k-fold cross validation from housing example

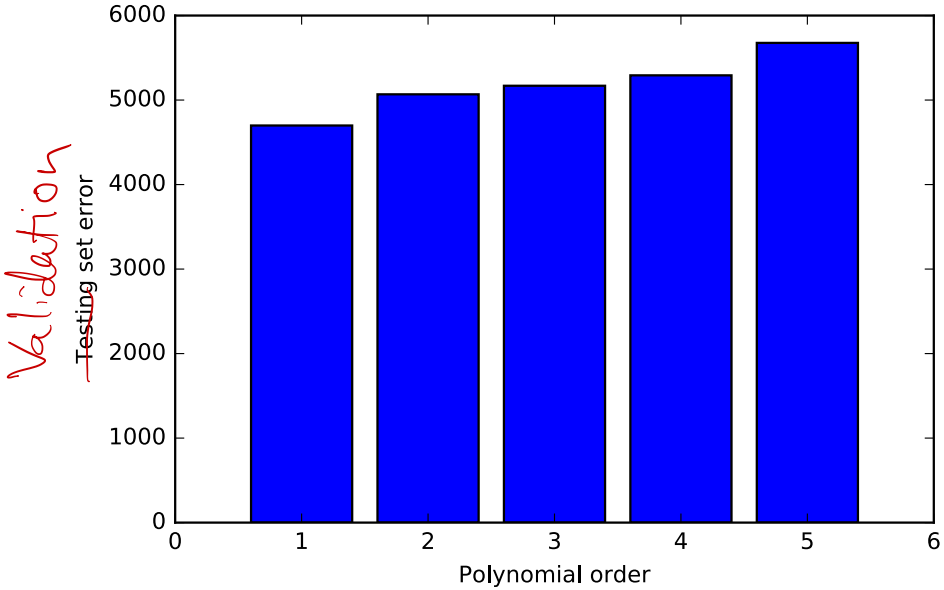
```
# (x,y) are the training data pairs from the supervised housing rent example
# num_train = len(x) is the number of examples
num_folds = 5
cv_idx = np.arange(num_train)
np.random.shuffle(cv_idx)
fold_size = num_train // 5
train_error, test_error = (np.zeros(5), np.zeros(5))
for i in np.arange(num_folds):
    test_idx = cv_idx[i*fold_size:(i+1)*fold_size]
    train_idx = np.concatenate((cv_idx[:i*fold_size], cv_idx[(i+1)*fold_size:]))
    x_test = x[test_idx]
    y_test = y[test_idx]
    x_train = x[train_idx]
    y_train = y[train_idx]
    thetas = train_models(x_train, y_train)
    train_error += test_models(x_train, y_train, thetas)
    test_error += test_models(x_test, y_test, thetas)

print train_error
print test_error
f = plt.figure()
ax = f.gca()
ax.bar(np.arange(5) + 1, train_error, width=0.8, align='center')
ax.set_xlabel('Polynomial order')
ax.set_ylabel('Training set error')
f.savefig('ml-basics_cv-train.pdf')
f = plt.figure()
ax = f.gca()
ax.bar(np.arange(5) + 1, test_error, width=0.8, align='center')
ax.set_xlabel('Polynomial order')
ax.set_ylabel('Testing set error')
f.savefig('ml-basics_cv-test.pdf')
```

```
[ 4421.38242236  4364.65797472  4254.01787959  3810.5847096   3731.95920537]
[ 4698.83840383  5067.8036503   5169.2079499   5292.40392825  5676.26492547]
```

k-fold cross validation from housing example





Other types of optimization

We've talked about examples where we want to *minimize* a mean-square error or distance metric.

Another metric that we may want to minimize is the *probability of having observed the data*. In this framework, the data is modeled to have some distribution with parameters. We choose the parameters to maximize the probability of having observed our training data.

H T H H T T H T

$x^{(i)} = \begin{cases} 0 & \text{w.p. } 1-\theta \\ 1 & \text{w.p. } \theta \end{cases}$

Model 1: $\theta = 1$

Model 2: $\theta = 0.5$

Model 3: $\theta = 0.75$

Model 1: $1^4 \cdot 0^4 = 0$

Model 2: $(0.5)^8 = 0.0039$

Model 3: $(0.75)^4 (0.25)^4 = 0.00124$

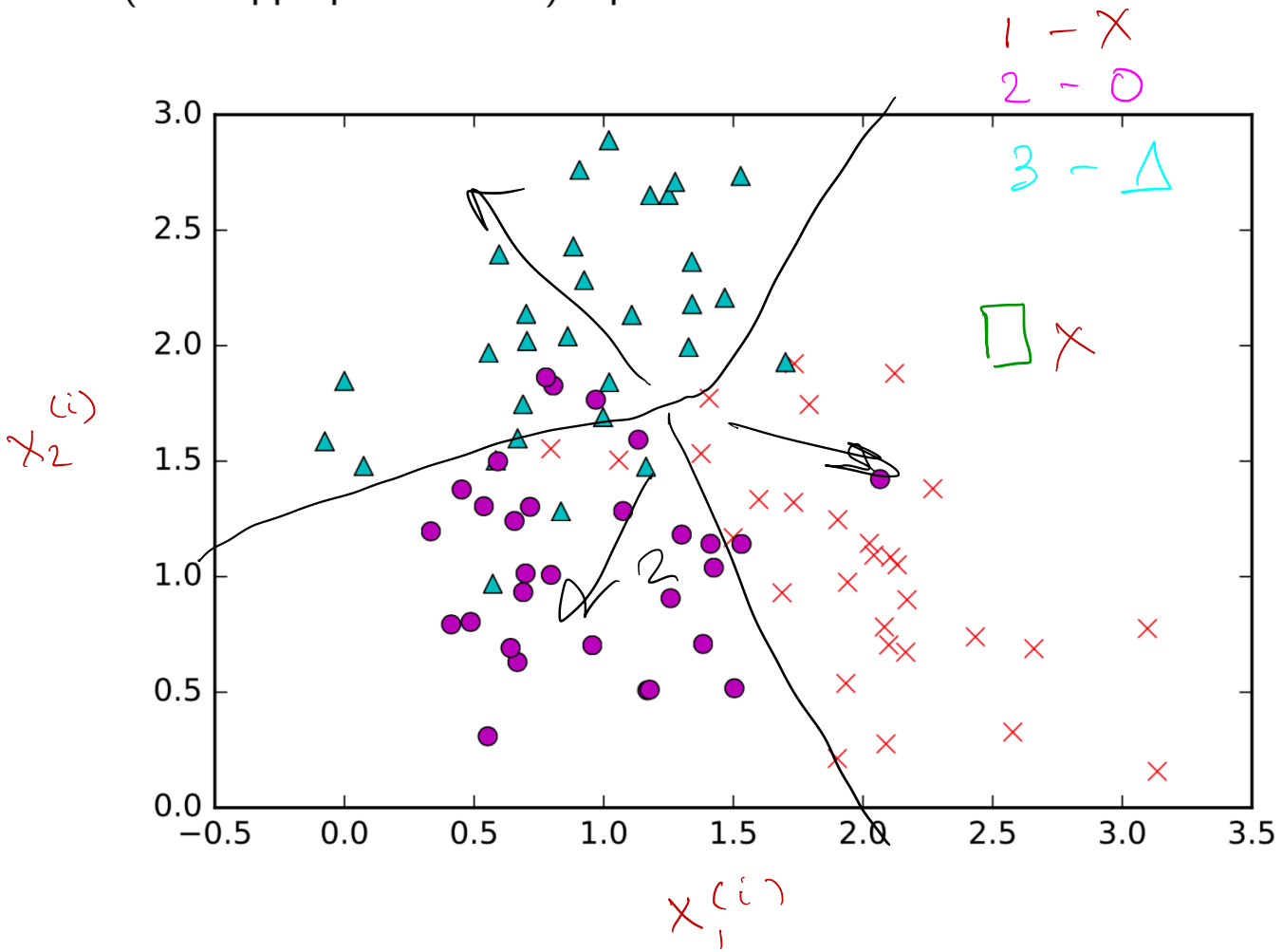
Maximum-likelihood introduction

H H H H H H

Pr ( ) under Model 1 : $1^5 \cdot (0)^0 = 1$

Example of ML estimation

Say we receive paired data $\{\mathbf{x}^{(i)}, y^{(i)}\}$ where $\mathbf{x}^{(i)} \in \mathbb{R}^2$ is a data point that belongs to one of three classes, $y^{(i)} \in \{1, 2, 3\}$. Classes $1, 2, 3$ denote the three possible classes (or labels) that a data point could belong to. The drawing below (with appropriate labels) represents this:



What is our goal in modeling?

What is our goal in modeling?

Chain rule for probability

$$\Pr(A=a) = p_A(a) = p(a)$$

$$\Pr(B=b) = p_B(b) = p(b)$$

$$\Pr\{A=a \text{ and } B=b\} = p_{A,B}(a,b) = p(a,b)$$

$$\Pr\{A=a \text{ and } B=b\} = \Pr\{A=a\}\Pr\{B=b \text{ given } A=a\}$$

$$= p(a)\,p(b|a)$$

$$= p(b)\,p(a|b)$$

$$p(a,b,c) = p(c)\,p(b|c)\,p(a|b,c)$$

$$= p(b)\,p(a|b)\,p(c|a,b)$$

$$= p(a)\,p(c|a)\,p(b|a,c)$$

Chain rule for probability

$$p(b,c \mid d,e) = \frac{p(b,c,d,e)}{p(d)\,p(e \mid d)}$$

$$p(e,d)$$

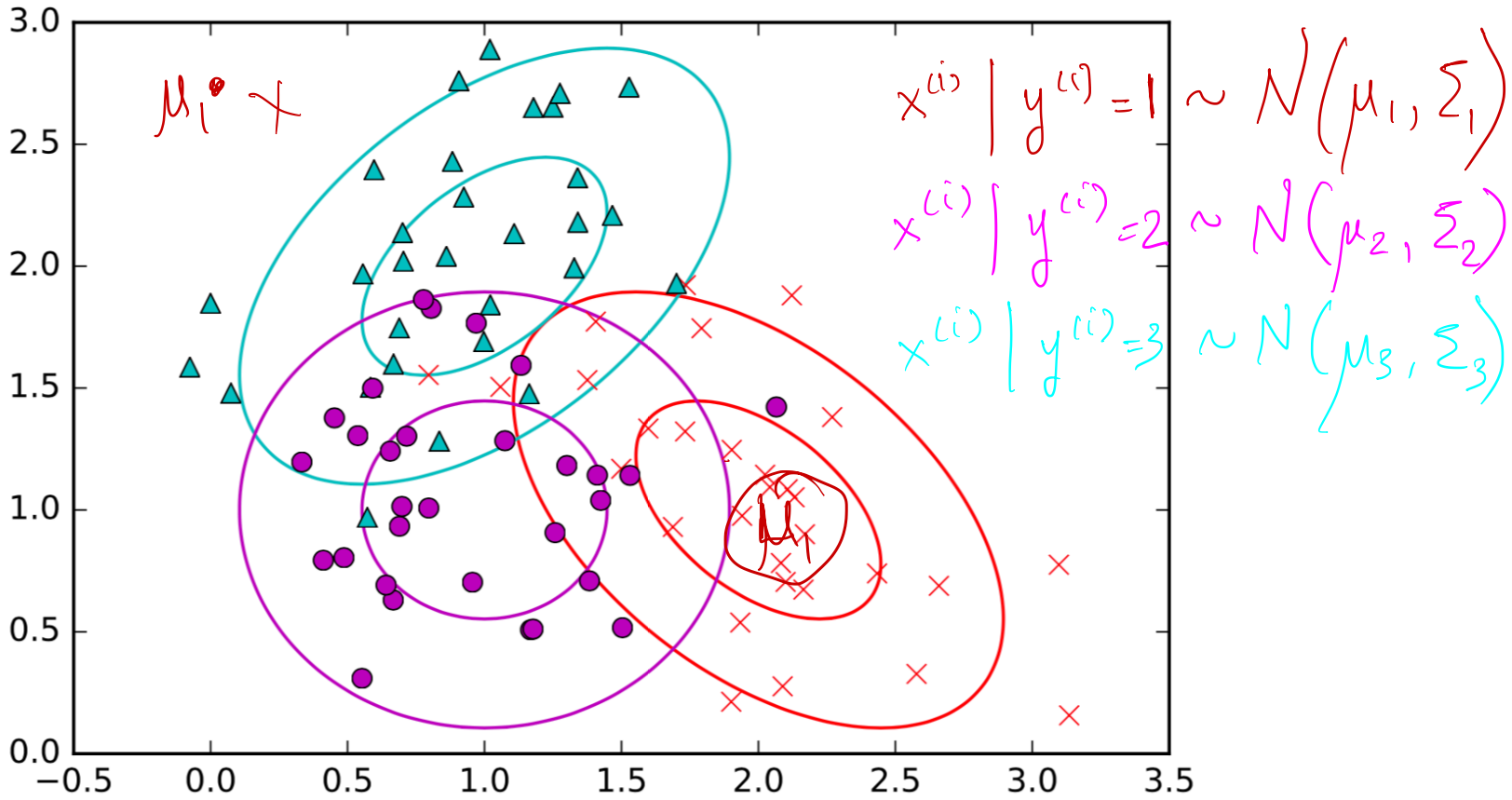
$$p(b,c,d,e) = p(e,d)\,p(b,c \mid d,e)$$

$$= p(e,d)\,p(b \mid d,e)$$

$$p(c \mid b,d,e)$$

Example of ML estimation (cont.)

We may want to model each cluster as being *generated* by a multivariate Gaussian distribution. Concretely, in this example, there are three Gaussian clusters, each one describing the distribution of points for that class.



Example of ML estimation (cont.)

The model setup is as follows:

- Each data point $\mathbf{x}^{(i)}$ belongs to class $y^{(i)}$.
- Each class y_i is parametrized according to a distribution.

$$\begin{aligned} \mathbf{x}^{(i)}|y^{(i)}=1 &\sim \mathcal{N}(\mu_1,\Sigma_1) \\ \mathbf{x}^{(i)}|y^{(i)}=2 &\sim \mathcal{N}(\mu_2,\Sigma_2) \\ \mathbf{x}^{(i)}|y^{(i)}=3 &\sim \mathcal{N}(\mu_3,\Sigma_3) \end{aligned}$$

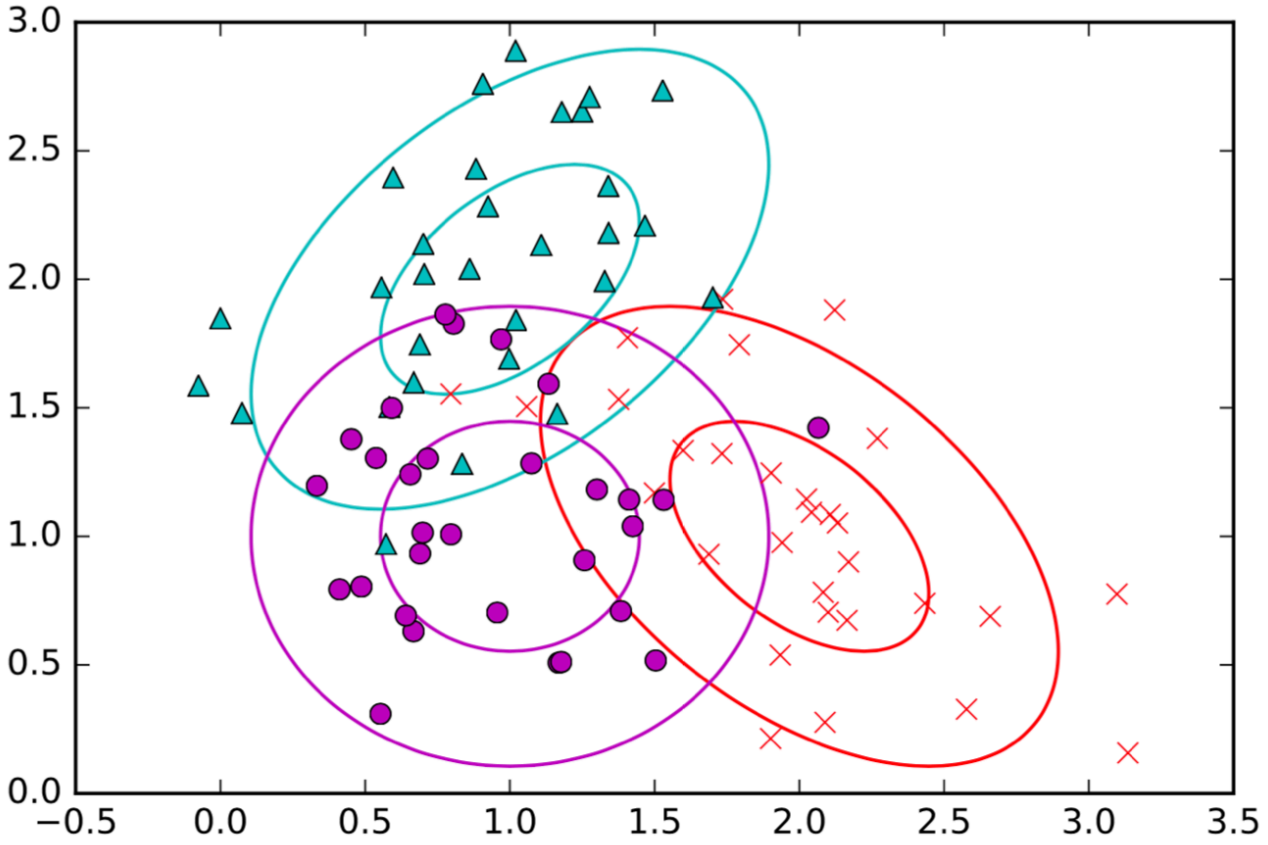
Thus, the parameters we can choose to optimize our model are $\theta=\{\mu_1,\Sigma_1,\mu_2,\Sigma_2,\mu_3,\Sigma_3\}$.

$p_\theta(y^{(i)}=j)=\frac{1}{3}$

- We'll assume all classes are equally probable a priori (so that maximum likelihood estimation and maximum a posteriori estimation are equivalent).
- Finally, we'll assume each data point is independent, so that we can easily write out probabilities, i.e.,

$$p\left(\{\mathbf{x}^{(i)},y^{(i)}\},\{\mathbf{x}^{(j)},y^{(j)}\}\right)=p\left(\mathbf{x}^{(i)},y^{(i)}\right)p\left(\mathbf{x}^{(j)},y^{(j)}\right)$$

Maximum-likelihood estimation



Maximum-likelihood estimation

Example of ML estimation (cont.)

$p(x^{(2)}, y^{(2)} \mid x^{(1)}, y^{(1)}) = p(x^{(2)}, y^{(2)})$

We'd like to maximize the likelihood of having seen the dataset. This can be written as

$$\mathcal{L} = p\left(\{\mathbf{x}^{(1)}, y^{(1)}\}, \{\mathbf{x}^{(2)}, y^{(2)}\}, \ldots, \{\mathbf{x}^{(N)}, y^{(N)}\}\right)$$

$$= \prod_{i=1}^{N} p(x^{(i)}, y^{(i)})$$

$$= \prod_{i=1}^{N} p(x^{(i)} \mid y^{(i)})\, p(y^{(i)})$$

$$\log \mathcal{L} = \sum_{i=1}^{N} \left[\log p(y^{(i)}) + \log p(x^{(i)} \mid y^{(i)})\right]$$

$$= k + \sum_{i=1}^{N} \log p(x^{(i)} \mid y^{(i)})$$

Maximum-likelihood estimation

Example of ML estimation (cont.)

Now, to simplify the probabilities, we evaluate $\log p(\mathbf{x}_i|y_i)$. We first observe:

$$\log p(\mathbf{x}^{(i)}|y^{(i)}=j) = \log \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma_j|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x}^{(i)}-\mu_j)^T\Sigma_j^{-1}(\mathbf{x}^{(i)}-\mu_j)\right)$$

$$= (2\pi)^{-n/2}|\Sigma_j|^{-1/2}\exp(\quad\quad)$$

$$= -\frac{n}{2}\log 2\pi - \frac{1}{2}\log|\Sigma_j| - \frac{1}{2}(x^{(i)}-\mu_j)^T\Sigma_j^{-1}(x^{(i)}-\mu_j)$$

$$= K_2 - \frac{1}{2}\log|\Sigma_j| - \frac{1}{2}(x^{(i)}-\mu_j)^T\Sigma_j^{-1}(x_i-\mu_j)$$

Maximum-likelihood estimation

$$\log \mathcal{L} = k_3 + \sum_{i=1}^{N} -\frac{1}{2}\log\left|\Sigma_{y^{(i)}}\right| - \frac{1}{2}\left(x^{(i)} - \mu_{y^{(i)}}\right)^T \Sigma_{y^{(i)}}^{-1} \left(x^{(i)} - \mu_{y^{(i)}}\right)$$

$$k_3 = N k_2 + N \log \frac{1}{3}$$

Maximum-likelihood estimation

$$\frac{\partial \log \mathcal{L}}{\partial \mu_j} = \frac{\partial}{\partial \mu_j}\left[\sum_{i:y^{(i)}=j} -\frac{1}{2}(\mathbf{x}^{(i)}-\mu_j)^T\Sigma_j^{-1}(\mathbf{x}^{(i)}-\mu_j)\right]$$

$$= -\frac{1}{2}\frac{\partial}{\partial \mu_j}\left[\sum_{i:y^{(i)}=j} x^{(i)T}\Sigma_j^{-1}x^{(i)} - x^{(i)T}\Sigma_j^{-1}\mu_j - \mu_j^T\Sigma_j^{-1}x^{(i)} + \mu_j\Sigma_j^{-1}\mu_j\right]$$

$$\frac{\partial}{\partial \mu_j}\mu_j^T a = a$$

$$\frac{\partial}{\partial x}x^TAx = (A+A^T)x$$

$$= -\frac{1}{2}\frac{\partial}{\partial \mu_j}\left[\sum_{i:y^{(i)}=j} x^{(i)T}\Sigma_j^{-1}x^{(i)} - 2\mu_j^T\Sigma_j^{-1}x^{(i)} + \mu_j\Sigma_j^{-1}\mu_j\right]$$

$$= -\frac{1}{2}\sum\left[0 - 2\Sigma_j^{-1}x^{(i)} + 2\Sigma_j^{-1}\mu_j\right]$$

$$= 0$$

Maximum-likelihood estimation

$$\sum_{i:y^{(i)}=j} \left(\Sigma_j^{-1} x^{(i)} \right) = \sum_{i:y^{(i)}=j} \left(\Sigma_j^{-1} \mu_j \right)$$

$$\sum_{i:y^{(i)}=j} x^{(i)} = N_j \mu_j$$

$$N_j = \sum_{i:y^{(i)}=j} 1$$

$$\mu_j = \frac{1}{N_j} \sum_{i:y^{(i)}=j} x^{(i)}$$

Examle of ML estimation (cont.)

We next take the derivative with respect to Σ, using the following facts (for more information, see "Tools" lecture notes on derivatives w.r.t. matrices):

$$\begin{aligned}\frac{\partial}{\partial \Sigma}\mathrm{tr}(\Sigma^{-1}\mathbf{A}) &= -\Sigma^{-T}\mathbf{A}^T\Sigma^{-T} \\ \frac{\partial}{\partial \Sigma}\log|\Sigma| &= \Sigma^{-T}\end{aligned}$$

Now, differentiating:

$$\frac{\partial \log \mathcal{L}}{\partial \Sigma_j} = \frac{\partial}{\partial \Sigma_j}\left[\sum_{i:y^{(i)}=j} -\frac{1}{2}\log|\Sigma_j| - \frac{1}{2}(\mathbf{x}^{(i)}-\mu_j)^T\Sigma_j^{-1}(\mathbf{x}^{(i)}-\mu_j)\right]$$

Maximum-likelihood estimation

Example of ML estimation (cont.)

Solving for the optimal parameters μ_j and Σ_j for all classes, we arrive at the following solution:

```
# x and y are the data and their corresponding labels
# x is 2 x num_examples of the data points
# y is 1 x num_examples of the data labels

class_idxs = [np.where(y == j)[0] for j in np.unique(y)]
means = []
covs = []

for this_class in class_idxs:
    means.append(np.mean(x[:, this_class], axis=1))
    covs.append(np.cov(x[:, this_class]))
```

Example of ML estimation (cont.)

Solving for the optimal parameters μ_j and Σ_j for all classes, we arrive at the following solution:

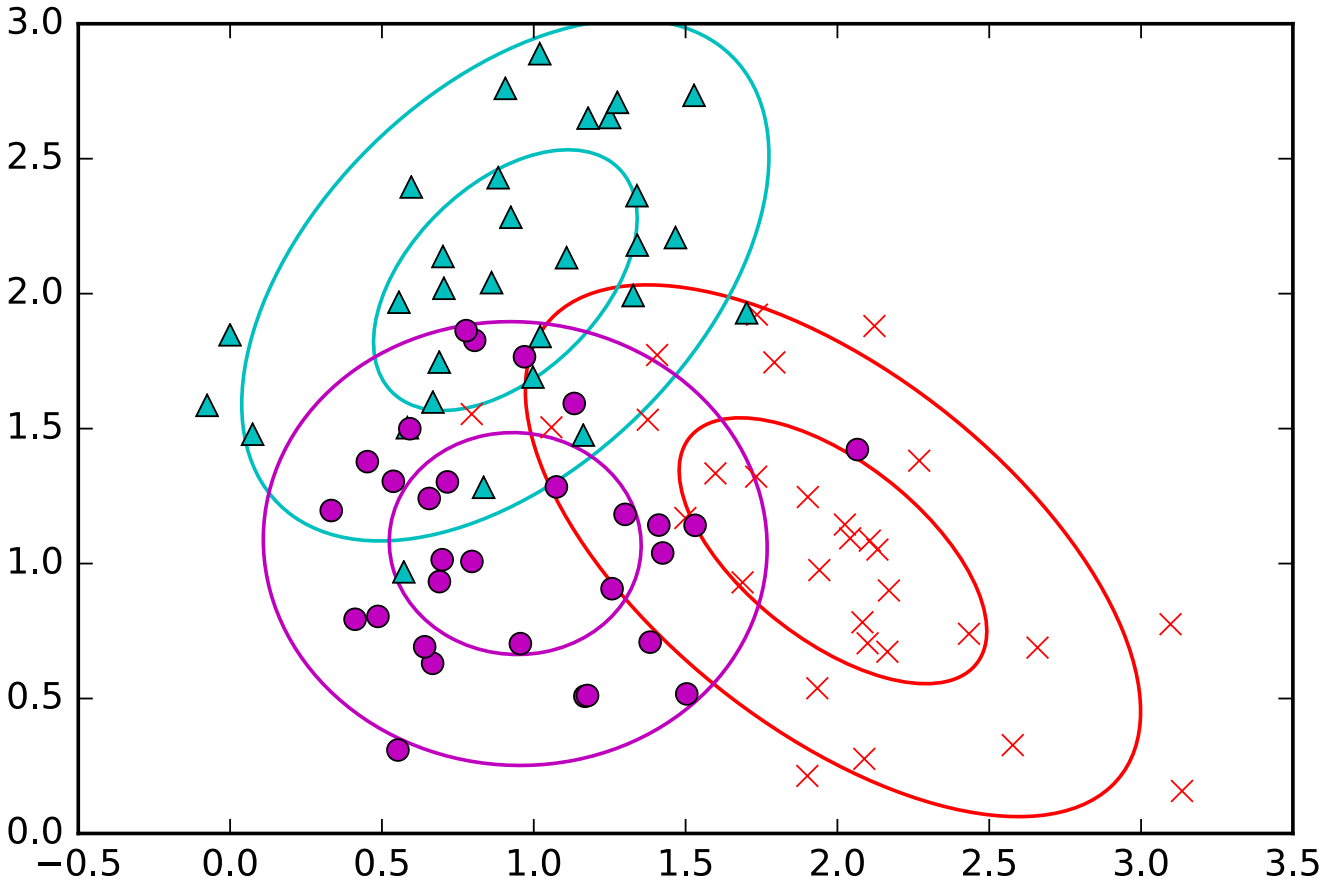
```
f = plt.figure()
ax = f.gca()
ax.plot(data1[0,:], data1[1,:], marker='x', linestyle='', color=class_colors[0])
ax.plot(data2[0,:], data2[1,:], marker='^', linestyle='', color=class_colors[1])
ax.plot(data3[0,:], data3[1,:], marker='o', linestyle='', color=class_colors[2])


for i in np.arange(len(class_idxs)):
    vals, vecs = eigsorted(covs[i])
    theta = np.degrees(np.arctan2(*vecs[:,0][::-1]))
    for nstd in np.arange(3):
        w, h = 2 * nstd * np.sqrt(vals)
        e = Ellipse(xy=(means[i][0], means[i][1]),
                    width=w, height=h,
                    angle=theta, color=class_colors[i])
        e.set_facecolor('none')
        ax.add_artist(e)

f.savefig('ml-basics_maxlik-data_fit-ellipsoids.pdf')
```

Example of ML estimation (cont.)

Solving for the optimal parameters μ_j and Σ_j for all classes, we arrive at the following solution:



Example of ML estimation (cont.)

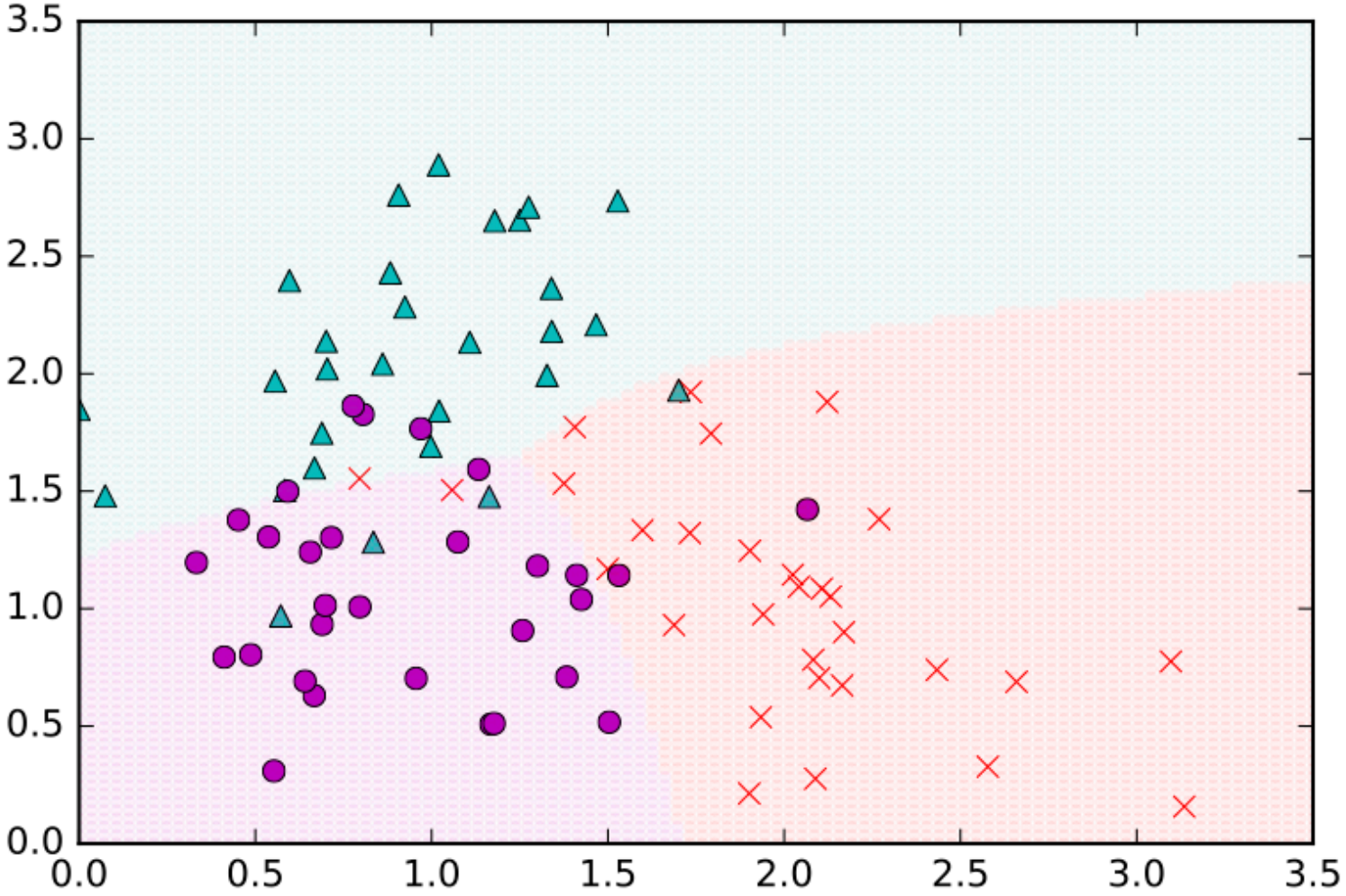
Imagine now a new point $\mathbf{x}$ comes in that we'd like to classify as being in one of three classes. To do so, we'd like to calculate: $\Pr(y = j|\mathbf{x})$ and pick the j that maximizes this probability. We'll denote this probability $p(j|\mathbf{x})$. We'll also assume, as earlier, that the classes are equally probable, i.e., $\Pr(y = j)$ is the same for all j.

Maximum-likelihood estimation

```
f = plt.figure()
ax = f.gca()
ax.plot(data1[0,:], data1[1,:], marker='x', linestyle='', color=class_colors[0])
ax.plot(data2[0,:], data2[1,:], marker='^', linestyle='', color=class_colors[1])
ax.plot(data3[0,:], data3[1,:], marker='o', linestyle='', color=class_colors[2])

for (i,xx) in enumerate(np.linspace(0, 3.5, 100)):
    for yy in np.linspace(0, 3.5, 100):
        xn = np.array((xx,yy))
        pmax = -np.Inf
        decoded_class = np.nan
        for c in np.arange(len(class_idxs)):
            class_prob = -np.log(np.linalg.det(covs[c])) - (xn-means[c]).T.dot(np.linalg.inv(covs[c])).dot(xn-means[c])
            if class_prob > pmax:
                pmax = class_prob
                decoded_class = c
        ax.plot(xx, yy, marker='.', color=class_colors[decoded_class], alpha=0.05)
ax.set_xlim(0,3.5)
f.savefig('ml-basics_maxlik-data_fit-ellipsoids_colorized.pdf')
```

Maximum-likelihood estimation



Even more cost functions

There are even more cost functions that we could use. We'll encounter some others in this class. Others include:

- MAP estimation.
- KL divergence.
- Maximize an approximation of the distribution.

In machine learning, it is important to arrive at an appropriate model and cost function. After that, it's important to know how to optimize it. In our examples here, the models were simple enough that we could differentiate and set the derivative equal to zero. In future lectures, we'll discuss more general ways to learn parameters of models when they are not so simple.

EXTRA SLIDES